Opponent Transformer: Modeling Opponent Policies as a Sequence Problem

Anonymous authors
Paper under double-blind review

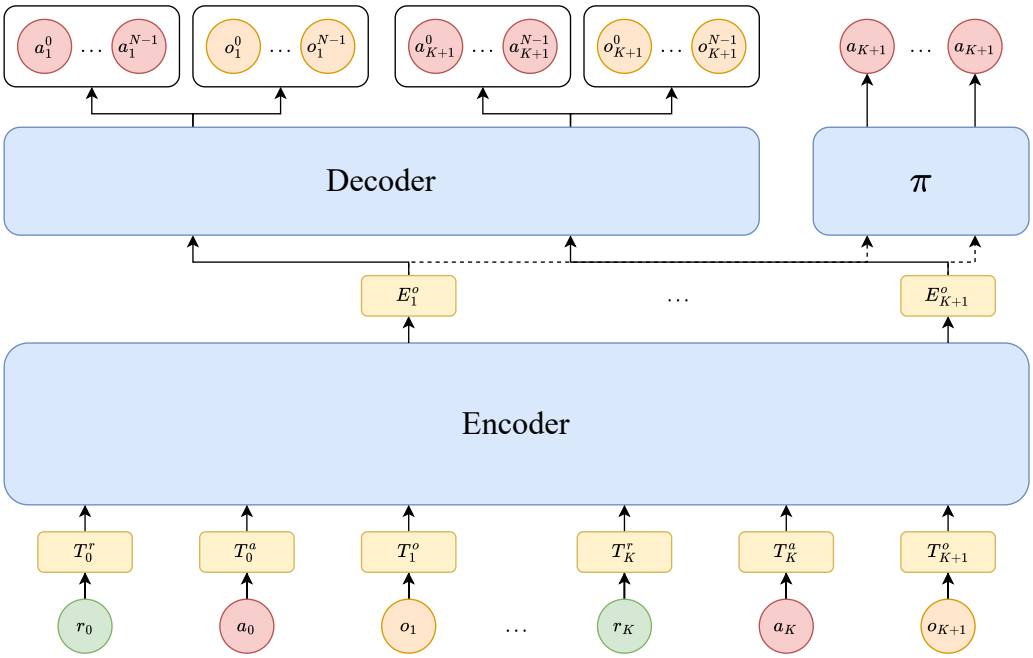


Figure 1: Opponent Transformer Architecture. We embed the controlled agent's previous reward, previous action, and current observation into embedding tokens, $T_t^{(r,a,o)}$, and transform them into an output sequence of embedding vectors, $E_t^{(r,a,o)}$. The embedding vectors are used to both condition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

Abstract

The ability of an agent to understand the intentions of others in a multi-agent system, also called opponent modeling, is critical for the design of effective local control policies. One main challenge is the unavailability of other agents' episodic trajectories at execution. To address the challenge, we propose a new approach that explicitly models the episodic trajectories of others. In particular, the proposed approach is to cast the opponent modeling problem as a sequence modeling problem via conditioning a transformer model on the sequence of the agent's local trajectory and predicting each opponent agent's trajectory. To evaluate the effectiveness of the proposed approach, we conduct experiments using a set of multi-agent environments that capture both cooperative and competitive payoff structures. The results show that the proposed method can provide better opponent modeling capabilities while achieving competitive or superior episodic returns.

1 Introduction

Multi-agent systems have seen remarkable progress in recent years in applications such as games (Han et al., 2019), traffic control (Chen et al., 2020), autonomous driving (Pal et al., 2021), etc. One of the core challenges in such systems is that the actions of all agents contribute to the transition of the system. Hence, it becomes crucial to model the actions of the other agents to reason about the optimal action to take in response. Opponent modeling (Albrecht & Stone, 2018) is the study of modeling concealed opponent information such that it can be used for conditioning the controlled agent's policy.

In opponent modeling, the primary challenge is to develop an agent capable of adapting to various opponent policies, relying solely on the available information during execution. This can become particularly challenging in settings where there is no direct information available about the opponents. In this case, the agent must learn to infer opponent behaviors as a function of its local information. Additionally, opponent policies may appear similar given a single transition. This ambiguity can be difficult to address without taking into account the temporal context. Therefore, an appropriate opponent modeling approach should learn a good representation of the opponent policy while taking into account the effect of the opponent policy over time.

There have been many recently proposed approaches for learning opponent models using deep learning (He & Boyd-Graber, 2016; Grover et al., 2018; Papoudakis et al., 2021). However, these approaches suffer some key drawbacks, namely, (1) access to opponent trajectories and (2) lack of considering the sequence of the controlled agent as a meaningful source of information. Motivated by the recent success of decision transformer (Chen et al., 2021) and its multi-agent variant (Wen et al., 2022), we propose to pose opponent modeling as a sequence modeling problem using a transformer model. Transformers have been incorporated in many ways in RL architectures in recent years, from feature extraction models to end-to-end policies (Agarwal et al., 2023; Ni et al., 2023; Gallici et al., 2023). In particular, we propose a transformer model that encodes the controlled agent's local trajectory into an embedding space that represents the effect of opponent policies. We train this model to reconstruct opponent trajectories conditioned on the local trajectory embedding. By doing so, we can train the model using opponent trajectories as supervised targets and at execution time. Consequently, an RL policy can condition on the embedding space purely as a function of the controlled agent's local trajectory. Our contributions are summarized as follows:

- **Opponent Modeling from Local Information:** To alleviate the requirement for access to opponent information at inference time, we only use local information to learn a latent representation as a proxy for the true opponent policy.
- **Local Trajectory as a Sequence Modeling Task:** Representing the controlled agent's local trajectory as a sequence yields more useful features over a fixed time horizon. Using a self-attention mechanism allows the opponent model to learn precisely which parts of the local trajectory correlate to the current opponent policy.
- **Online Joint Opponent Model and Policy Training:** We train the opponent model and the agent policy jointly in an online fashion. Because transformers notoriously require large amounts of data to converge properly, we believe online training provides a diverse enough dataset to achieve superior opponent modeling performance.

We also evaluate the proposed approach on a set of two multi-agent RL tasks from the Multi-agent Particle Environments (MPE) (Mordatch & Abbeel, 2017) representing cooperative and competitive payoff structures. We provide a comparison between the proposed approach and some baseline methods based on agent's episodic returns as well as the accuracy of the opponent model on the reconstruction of opponent trajectories. The comparison shows that the proposed method yields better opponent modeling capabilities while achieving comparable or better episodic returns.

2 Related Work

2.1 Opponent Modeling

When operating in a decentralized multi-agent system, it is important to incorporate information about other agents to determine a best response to a given state. In traditional centralized training with decentralized execution (CTDE) approaches such as MADDPG (Lowe et al., 2017) and MAPPO (Yu et al., 2022), this information is used by training a centralized critic that conditions on the joint observations of all agents which is then implicitly distilled into the actor policy. Opponent modeling is an alternative approach that explicitly learns to model opponent information. There is a large body of work on opponent modeling in multi-agent settings (Albrecht & Stone, 2018). He & Boyd-Graber (2016) 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 an encoder-decoder architecture using imitation learning and a contrastive triplet loss to both learn to accurately reconstruct opponent policies and correctly identify the opponent policy within the embedding space. Building off of the work of Grover et al. (2018), Papoudakis et al. (2021) also used an encoder-decoder architecture for reconstructing opponent policies. However, they model this reconstruction using the controlled agent's local trajectory only. In this sense, the work in Papoudakis et al. (2021) serves as the most direct inspiration for this work. Zhang et al. (2023) introduced an approach that adapts to changing policies, similar to our problem setting, however the opponents in this work can switch policies within an episode, so the model must learn to quickly adapt. Xing et al. (2023) studied ad hoc teamwork wherein 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 it's teammates as well as the distribution of policy types to generalize to unseen teammate sets. Finally, Ma et al. (2024) learned an opponent 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 sub-field of machine learning including computer vision Dosovitskiy et al. (2021) and more recently for reinforcement learning (Agarwal et al., 2023) since inception. 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 as well as 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) posed 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 re-imagined as a sequence-to-sequence task (Wen et al., 2022) where the model maps input sequences of observations to output sequences of actions. Similar to our problem setting, Jing et al. (2024) introduced a transformer architecture for learning opponent policy representations from offline datasets. In this paper, we are interested in learning latent representations of the controlled agent's local trajectory as a function of opponent policies.

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 $\mathcal{A} = \mathcal{A}^0 \times \ldots \times \mathcal{A}^N$ representing 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: $\mathcal{A} \times \mathcal{S} \times \mathcal{O}^i \to [0, 1]$ given by

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

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

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

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

Agent i uses a policy $\pi^i(a^i|o^i)$, which is a probability distribution over the set of actions $\mathcal{A}^i$ to select an action $a^i \in \mathcal{A}^i$ given an observation $o^i \in \mathcal{O}^i$. 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 compute 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 (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 $\{T_t, \ldots, T_{t+K}\}$ with context length K and transforms them into representation embedding vectors $\{E_t, \ldots, E_{t+K}\}$. 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 an opposing set of agents which can utilize one of several fixed policies. To be specific, we assume that each individual opponent agent i adopts a policy $\pi^{i,m}$, the collection of which forms the joint opponent policy $\pi^{-1,m}$. These policies can be heuristic or can be learned using RL. In this work, we consider the set of M joint opponent policies $\Pi = \{\pi^{-1,m} | m = 1, \ldots, M\}$ that can be either heuristic or pretrained using

RL. For simplicity, from here on we refer to the controlled agent with no superscript and all of the opponents with superscript -1. Thus the agent has an action space $\mathcal{A}$ and an observation space $\mathcal{O}$. Similarly, the opponents 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 opponent 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 opponent policy type m is concealed from the 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 opponent 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 later and introduce a transformer-based approach to modeling such opponent policies.

4 Method

4.1 Opponent Transformer

We format opponent 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 agent's previous reward, $a_{t-1} \sim \mathcal{A}$ is the agent's previous action, and $o_t \sim \mathcal{O}$ is the agent's current observation. The agent's local episodic trajectory 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, individual opponent 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 opponent modeling is to learn a representation of the joint opponent policy $\pi^{-1,m}$ such that this representation can be included as an inductive bias for the agent policy. Inspired by the recent success of transformers in such problems, we build a transformer encoder model, which we refer to as the Opponent Transformer, 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 local agent's 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 the observation embeddings E_{t+K}^o for downstream tasks. This embedding vector E_{t+K}^o in addition to the 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 opponent policy.

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

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

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

4.2 Policy Training

The goal of the controlled agent is to learn a policy that adapts to different joint opponent policies $\pi^{-1,m}$. We train the Opponent Transformer such that the embedding vector E_t^o is a good proxy for the true opponent information. By incorporating this vector into the agent policy, it allows the policy to better adapt to varying opponent 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) algorithm (Mnih et al., 2016). Thus, the RL objective is given by

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

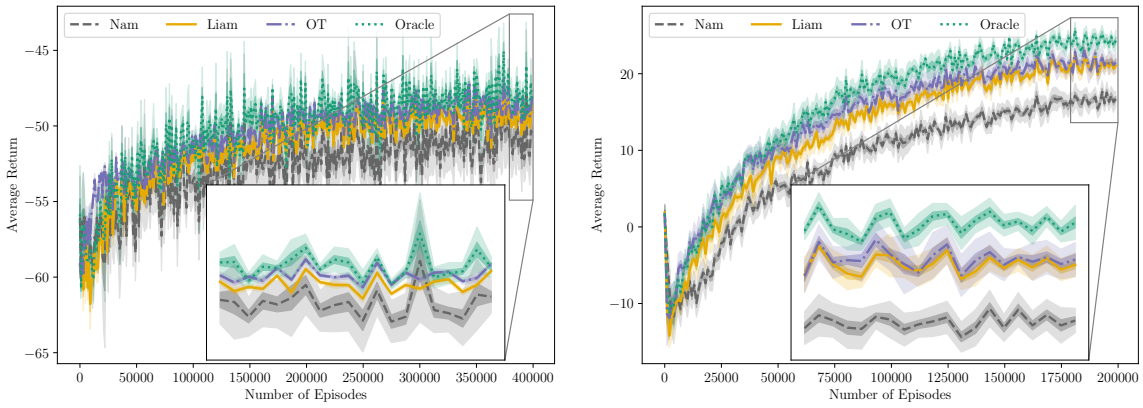
where B is a batch of transitions, π_θ is the policy parameterized by θ, V_ϕ is the value function parameterized by ϕ, A^π is the advantage function under policy π, and H is the entropy function weighted by the entropy coefficient β. We optimize (6) and (7) jointly, sampling the set of opponent policies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

5 Experiments

5.1 Experimental Setup

To validate our approach, we performed experiments in two Multi-Agent Particle Environments (MPEs). Specifically, we utilized a cooperative MPE scenario from (Mordatch & Abbeel, 2017) and an adapted competitive MPE scenario (Boehmer et al., 2020). Each experiment showcases a unique scenario where cooperativeness or competitiveness plays a vital role and must be modeled appropriately. Through rigorous analysis, we assessed the performance of our approach, both in terms of modeling opponent behavior and solving the final task. In all 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. Additionally, we utilized a transformer encoder that is comprised of four transformer blocks with four attention heads and a hidden dimension of 128. We trained policies for 10 million time steps and evaluated every 100 episodes. To ensure the reproducibility of the results, we run ten different training runs with different random seeds and plot the average of the results to provide reliable evidence of our approach's performance.

We compare our proposed method against several key baselines that showcase an array of solutions in this space. Some baselines employ an explicit opponent model, while others are implicit. These baselines can be categorized based on the amount of information available to the controlled agent about the opponents: (i) **No Agent Modelling (NAM):** This baseline only has access to the controlled agent's current observation and last action. With no information of the opponent agents, this baseline serves as the lower baseline of performance. (ii) **Local Information Agent Modelling (LIAM):** This baseline from Papoudakis et al. (2021) uses an encoder-decoder architecture to encode the controlled agent's local information into an embedding space, similar to our approach, and then decode the embedding vector to reconstruct the set of opponent observations and actions. Only the encoder is used during inference thus only giving direct access to the controlled agent's information. (iii) **Oracle:** This baseline assumes knowledge of the opponent trajectories at all times, including opponent observations and actions. The controlled agent conditions on the joint vector of its local observation, its last action, and the set of observations and actions from the opponents. Since there is no ambiguity about the intents and strategies of the opponent agents, we expect this baseline to serve as the upper baseline of performance.



(a) Spread returns.

(b) Tag returns.

Figure 2: Average episodic returns and 95% confidence intervals for the two experimental scenarios across ten random seeds with a zoom-in view of the last 20,000 episodes.

5.2 Experimental Environments

5.2.1 Cooperative Navigation (Spread)

We adopt the original implementation of the cooperative navigation scenario from Mordatch & Abbeel (2017). The scenario involves three agents and three landmarks, all initialized in random starting positions. Agents are tasked to cooperate to navigate and cover all landmarks while avoiding any collisions with each other. All agents are rewarded collectively based on the sum of the distances between each landmark and the closest agent. Agents are penalized if they collide with each other.

To generate the set of opponent policies, we pretrain three sets of opponent policies using MAPPO and IPPO, each with different random seeds. This results in eight total pretrained opponent policies. Additionally, we use five heuristic policies: three policies to cover the first, second, and third landmarks, respectively, and two policies to cover the farthest and closest landmarks. These 17 policies can be sampled to generate new joint opponent policies.

We report the average evaluation returns across the ten training runs with the 95% confidence interval in Figure 2(a). As expected, the NAM and Oracle baselines serve as the lower and upper bounds of performance, respectively. LIAM performs well, nearly matching the upper level of performance set by the Oracle agent. Notably, our proposed Opponent Transformer (OT) achieves the highest return, with the exception of the Oracle agent. Moreover, OT tends to converge to an optimal average return more quickly than other models, including the Oracle baseline.

The opponent modeling results are listed in Table 1. OT achieves higher action reconstruction accuracy and lower observation reconstruction error during training and testing.

Table 1: **Joint opponent reconstruction performance on spread scenario.** We report the accuracy of the opponent action reconstruction, as well as the MSE of the joint opponent observation reconstruction, averaged across ten randomly seeded runs. The best results are shown in bold.

Opponent Model	Train Action Accuracy	Train Obs. MSE	Test Action Accuracy	Test Obs. MSE
LIAM	63.0 ± 2.61	0.0450 ± 0.00923	65.6 ± 2.50	0.0404 ± 0.00427
Ours	$\mathbf{85.3 \pm 0.59}$	$\mathbf{0.00492 \pm 0.00085}$	$\mathbf{69.4 \pm 0.63}$	$\mathbf{0.00312 \pm 0.00014}$

5.2.2 Predator-Prey (Tag)

We use a modified predator-prey environment proposed in Boehmer et al. (2020) which was additionally used in the evaluation for Papoudakis et al. (2021). The scenario has two large landmarks,

three adversarial agents, and one good agent which we control. The good agent is faster than the adversaries. In this modified version of the classic predator-prey task, the good agent is given a reward of $+1$ if it collides with just one of the adversaries and then all of the adversaries are rewarded with -1. However, if the agent collides with more than one adversary, it is rewarded with -1 and the adversaries are rewarded with $+1$. Additionally, if the agent travels to the boundary of the environment it is penalized with -10.

For this task, we use the ten heuristic and pretrained policies from Papoudakis et al. (2021). This includes four heuristic policies: (i) going after the prey, (ii) going after one of the predators, (iii) going after the agent (predator or prey) that is closest, (iv) going after the predator that is closest. The remaining six policies were trained using MADDPG and IA2C.

The average evaluation returns are shown in Figure 2(b). Again we see that NAM and Oracle perform as expected. LIAM performs closer to the upper baseline than the lower. OT achieves an average return matching or exceeding the Oracle agent and once again converges quicker.

The opponent modeling results, listed in Table 2, show that OT again outperforms LIAM during training for both opponent action and observation reconstruction. However, OT underperforms in opponent action reconstruction during evaluation.

Table 2: **Joint opponent reconstruction performance on tag scenario.** We report the accuracy of the opponent action reconstruction, as well as the MSE of the joint opponent observation reconstruction, averaged across ten randomly seeded runs. The best results are shown in bold.

Opponent Model	Train Action Accuracy	Train Obs. MSE	Test Action Accuracy	Test Obs. MSE
LIAM	72.2 ± 2.91	0.716 ± 0.371	$\mathbf{71.3 \pm 0.43}$	1.74 ± 0.075
Ours	$\mathbf{78.3 \pm 1.16}$	$\mathbf{0.0793 \pm 0.00193}$	66.9 ± 0.63	$\mathbf{0.109 \pm 0.0080}$

6 Conclusion and Future Work

In this paper, we proposed a new opponent modeling architecture called Opponent Transformer. This approach does not require access to opponent information at execution time, making the controlled agent fully decentralized. This architecture makes use of a transformer which allows the model to exploit and extract features present in the sequence of the controlled agent's episodic trajectory. We presented evidence of the model's efficacy using the Predator-Prey and Cooperative Navigation scenarios from the multi-agent particle environments.

One future research direction is to explore regularization techniques for balancing the opponent modeling and policy optimization tasks. Another direction is to explore the limitations of current opponent modeling techniques for larger multi-agent systems.

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.

Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward A thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 3414–3421. AAAI Press, 2020. doi: 10.1609/AAAI.V34I04.5744. URL https://doi.org/10.1609/aaai.v34i04.5744.

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.

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.

Matteo Gallici, Mario Martin, and Ivan Masmitja. Transfqmix: Transformers for leveraging the graph structure of multi-agent reinforcement learning problems. In Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (eds.), *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pp. 1679–1687. ACM, 2023. doi: 10.5555/3545946.3598825. URL https://dl.acm.org/doi/10.5555/3545946.3598825.

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.

Lei Han, Peng Sun, Yali Du, Jiechao Xiong, Qing Wang, Xinghai Sun, Han Liu, and Tong Zhang. Grid-wise control for multi-agent reinforcement learning in video game AI. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2576–2585. PMLR, 2019. URL http://proceedings.mlr.press/v97/han19a.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.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/9dc5accb1e4f4a9798eae145f2e4869b-Abstract-Conference.html.

Avik Pal, Jonah Philion, Yuan-Hong Liao, and Sanja Fidler. Emergent road rules in multi-agent driving environments. 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=d8Q1mt2Ghw.

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.

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.