Local Information Agent Modelling in Partially-Observable Environments

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modelling the behaviours of other agents is essential for understanding how agents interact and making effective decisions. Existing methods for agent modelling commonly assume knowledge of the local observations and chosen actions of the modelled agents during execution. To eliminate this assumption, we extract representations from the local information of the controlled agent using encoder-decoder architectures. Using the observations and actions of the modelled agents during training, our models learn to extract representations about the modelled agents conditioned only on the local observations of the controlled agent. The representations are used to augment the controlled agent's decision policy which is trained via deep reinforcement learning; thus, during execution, the policy does not require access to other agents' information. We provide a comprehensive evaluation and ablations studies in cooperative, competitive and mixed multi-agent environments, showing that our method achieves significantly higher returns than baseline methods which do not use the learned representations.

1 Introduction

An important aspect of autonomous decision-making agents is the ability to reason about the unknown intentions and behaviours of other agents. Much research has been devoted to this *agent modelling* problem [Albrecht and Stone, 2018], with recent works focused on learning informative representations about another agent's policy using deep learning architectures for agent modelling and reinforcement learning (RL) [He et al., 2016, Raileanu et al., 2018, Grover et al., 2018, Rabinowitz et al., 2018, Zintgraf et al., 2021].

A common assumption in existing methods is that the modelling agent has access to the local trajectory of the modelled agents during execution [Albrecht and Stone, 2018], which may include their local observations of the environment state and their past actions. While it is certainly desirable to be able to observe the agents' local contexts in order to reason about their past and future decisions, in practice such an assumption may be too restrictive. Agents may only have a limited view of their surroundings, communication with other agents may be infeasible or unreliable [Stone et al., 2010], and knowledge of the perception system of other agents may be unavailable [Gmytrasiewicz and Doshi, 2005]. In such cases, an agent must reason with only locally available information.

We consider the following question: *Can effective agent modelling be achieved using only the locally available information of the controlled agent during execution?* A strength of deep learning techniques is their ability to identify informative features in data. Here, we use deep learning techniques to extract informative features about the trajectory of the modelled agent from a stream of local observations for the purpose of agent modelling. Specifically, we consider a multi-agent setting in which we control one agent which must learn to interact with a set of other agents. We assume a given set of possible policies for the non-learning agents and that these policies are fixed.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

We propose *Local Information Agent Modelling* (LIAM), which can be seen as the general idea of learning the relationship between the trajectory of the controlled agent and the trajectory of the modelled agent. In this work we propose one instantiation of this idea: an encoder-decoder agent modelling method that can extract a compact yet informative representation of the modelled agents given only the local information of the controlled agent (its local state observations, and past actions). The model is trained to replicate the observations and actions of the modelled agents from the local information only. During training, the modelled agent's observations and actions are utilised as reconstruction targets for the decoder; after training, only the encoding component is retained which generates representations using local observations of the controlled agent. The learned representation conditions the policy of the controlled agent in addition to its local observation, and the policy and model are optimised concurrently during the RL learning process.

We evaluate LIAM in three different multi-agent environments: double speaker-listener [Mordatch and Abbeel, 2017, Lowe et al., 2017], level-based foraging (LBF) [Albrecht and Stone, 2017, Papoudakis et al., 2020], and a modified version of predator-prey proposed by [Böhmer et al., 2020]. Our results support the idea that effective agent modelling can be achieved using only local information during execution: the same RL algorithm generally achieved higher average returns when combined with representations generated by LIAM than without, and in some cases the average returns are comparable to those achieved by an ideal baseline which has access to the modelled agent's trajectory. We also provide detailed evaluations of the learned encoder and decoder of LIAM as well as comparison with different instantiations of LIAM.

2 Related Work

Learning Agent Models: We are interested in agent modelling methods that use neural networks to learn representations of the other agents. He et al. [2016] proposed a method which learns a modelling network to reconstruct the modelled agent's actions given its observations. Raileanu et al. [2018] developed an algorithm for learning to infer an agent's intentions using the policy of the controlled agent. Grover et al. [2018] proposed an encoder-decoder method for modelling the agent's policy. The encoder learns a point-based representation of different agent trajectories, and the decoder learns to reconstruct the modelled agent's policy. Rabinowitz et al. [2018] proposed the Theory of mind Network (TomNet), which learns embedding-based representations of modelled agents for meta-learning. Tacchetti et al. [2019] proposed relational forward models to model agents using graph neural networks. [Zintgraf et al., 2021] uses a VAE for agent modelling for fully-observable tasks. All these aforementioned works either assume that the controlled agent has direct access to the trajectories of the modelled agent during execution or that the evaluation environments are fully-observable. Xie et al. [2020] learned latent representations from local information to influence the modelled agents, however, in contrast to our work, they did not utilise the modelled agent's trajectories. Finally, agent modelling from local information has been researched under the I-POMDP model [Gmytrasiewicz and Doshi, 2005] and in the Poker domain research. In contrast to our work, I-POMDPs utilise recursive reasoning [Albrecht and Stone, 2018] which assumes knowledge of the observation models of the modelled agents (which is unavailable in our setting). In the Poker domain, Johanson et al. [2008], Bard et al. [2013] created a mixture-of-expert counter-strategies during training against several type of opponent policies, while during execution they cast the selection of the best counter-strategy against a specific opponent type as a bandit problem. The main difference between LIAM and these works is that the latter adapt (to select the best strategy) over a number of hands (we consider that a hand is equivalent to an episode) against each opponent. In contrast, in our work we use a single episode for adaptation.

Representation Learning in Reinforcement Learning: Another related topic which has received significant attention is representation learning in RL. Using unsupervised learning techniques to learn low-dimensional representations of the environment state has led to significant improvements both in the returns as well as the sample efficiency of the RL. Recent works on representation learning focus on learning world models and use them to train the RL algorithm [Ha and Schmidhuber, 2018, Hafner et al., 2020] or learning state representations for improving the sample sample efficiency of RL algorithms using reconstruction-based [Corneil et al., 2018, Igl et al., 2018, Gregor et al., 2019, Kurutach et al., 2018, Gelada et al., 2019] or non-reconstruction-based methods [Laskin et al., 2020, Zhang et al., 2020], using contrastive learning or bi-simulation metrics. Works on meta-RL and transfer learning focus on learning representations over tasks and use them to solve

new and previously unseen tasks [Doshi-Velez and Konidaris, 2016, Hausman et al., 2018, Zhang et al., 2018, Rakelly et al., 2019, Zintgraf et al., 2019]. In contrast to these work, LIAM focuses on learning representations about the relationship between the trajectories of the controlled agent and the trajectories of the modelled agent.

Multi-agent Reinforcement Learning (MARL): MARL algorithms are designed to train multiple agents to solve tasks in a shared environment [Hernandez-Leal et al., 2017, Papoudakis et al., 2019]. At the beginning of the training, a number of agents are untrained and typically initialised with random policies. A common paradigm in MARL is Centralised Training with Decentralised Execution (CTDE), in which the trajectories of all agents are utilised during training, while during execution each agent conditions its individual policy only on its local trajectory. The information of all agents can be utilised during training for various reasons, such as computing a joint value function [Lowe et al., 2017, Foerster et al., 2018, Sunehag et al., 2018, Rashid et al., 2018], or generating intrinsic rewards [Jaques et al., 2019]. During execution, only the local information of each agent is used for selecting actions in the environment. Our work relates to CTDE in that we train LIAM in a centralised fashion using the trajectories of all the existing agents in the environment. However, during execution, the learned model uses only the local trajectory of the controlled agent.

3 Approach

3.1 Problem Formulation

We control a single agent which must learn to interact with other agents that use one of a fixed number of policies. We model the problem as a Partially-Observable Stochastic Game (POSG) [Shapley, 1953, Hansen et al., 2004] which consists of N agents $\mathbb{I} = \{1, 2, ..., N\}$, a state space $\mathcal{S}$, the joint action space $\mathcal{A} = \mathcal{A}^1 \times ... \times \mathcal{A}^N$, a transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ specifying the transition probabilities between states given a joint action, and for each agent $i \in \mathbb{I}$ a reward function $r^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. We consider partially-observable settings, where each agent i has access only to its local observation $o^i \in \mathcal{O}^i$, where $\mathcal{O}^i$ is the observation set of agent i. The observation function $\Omega^i : \mathcal{S} \times \mathcal{A} \times \mathcal{O}^i \rightarrow [0, 1]$ defines a probability distribution over the possible next observations of agent i given the previous state and the joint action of all agents.

We denote the agent under our control by 1, and the modelled agents by -1 where for notational convenience we will treat the modelled agents as a single "combined" agent with joint observations o^{-1} and actions a^{-1}. We assume a set of fixed policies, $\Pi_k = \{\pi^{-1,k} | k = 1, ..., K\}$, which may be defined manually (heuristic) or pretrained using RL. Each fixed policy determines the modelled agent's actions as a mapping $\pi^{-1,k}(o^{-1})$ from the modelled agent's local observation o^{-1} to a distribution over actions a^{-1}. Our goal is to find a policy π_θ parameterised by θ for agent 1 which maximises the average return against the fixed policies from the training set Π_k, assuming that each fixed policy is initially equally probable and fixed during an episode:

$$\arg\max_{\theta} \mathbb{E}_{\pi_\theta, \pi^{-1,k} \sim \mathcal{U}(\Pi_k)} \left[\sum_{t=0}^{H-1} \gamma^t r^1_{t+1} \right] \tag{1}$$

where r^1_{t+1} is the reward received by agent 1 at time $t + 1$ after performing the action a^1_t, H is the episode length (horizon), and $\gamma \in (0, 1)$ is a discount factor.

3.2 Local Information Agent Modelling

We aim to learn the relationship between the trajectory of the controlled agent and the trajectory of the modelled agent. We denote by $\tau^{-1} = \{o^{-1}_t, a^{-1}_t\}^{t=H}_{t=0}$ the trajectory of the modelled agent where o^{-1}_t and a^{-1}_t are the modelled agent's observation and action at time step t in the trajectory, up to horizon H. These trajectories are generated from the fixed policies in Π_k. We assume the existence of some latent variables (or *embeddings*) in the space $\mathcal{Z}$, and at each time step t the latent variables z_t contain information both about the fixed policy that is used by the modelled agent as well as the dynamics of the environment as perceived by the modelled agent. To learn the relationship between the modelled agent's trajectory and the latent variables we can use a parametric decoder. The decoder is denoted as $f_{\boldsymbol{u}} : \mathcal{Z} \rightarrow \tau^{-1}$ and is the model that decodes the latent variables to the trajectory of the modelled agent.

The last step to learn the relationship between the local and the modelled agent's trajectories is to use a recurrent encoding model, that we denote as $f_{\boldsymbol{w}} : \tau^1 \rightarrow \mathcal{Z}$, with parameters $\boldsymbol{w}$, to learn the relationship between the local trajectory of the controlled agent and the latent variables. Specifically, we learn the function that relates the modelled agent's trajectory to the latent variables with an encoder that only depends on local information of the controlled agent. Since during execution only the encoder is required to generate the latent variables of the modelled agent, this approach removes the assumption that access to the modelled agent's observations and actions is available during execution.

At each time step t, the recurrent encoder network generates an embedding z_t, which is conditioned on the information of the agent under control $(o^1_{1:t}, a^1_{1:t-1})$, until time step t. At each time step t the parametric decoder learns to reconstruct the modelled agent's observation and action (o^{-1}_t and a^{-1}_t) conditioned on the embedding z_t. The modelled agent's observation depends on the dynamics of the environment that are encoded in z_t. The modelled agent's action at each time step depends on its observation and the modelled agent's policy, which is also encoded by z_t. Therefore, the decoder consists of two fully-connected feed-forward networks; the observation reconstruction network $f^o_{\boldsymbol{u}}$, and the policy reconstruction network $f^\pi_{\boldsymbol{u}}$ (see Figure 1). In each time step t, the observation reconstruction network receives as input embedding z_t and reconstructs the modelled agent's observation o^{-1}_t. The policy reconstruction network, receives as input the observation of the modelled agent o^{-1}_t concatenated together with the embedding z_t and outputs a categorical distribution over the modelled agent's action a^{-1}_t. The encoder-decoder loss can be written as:

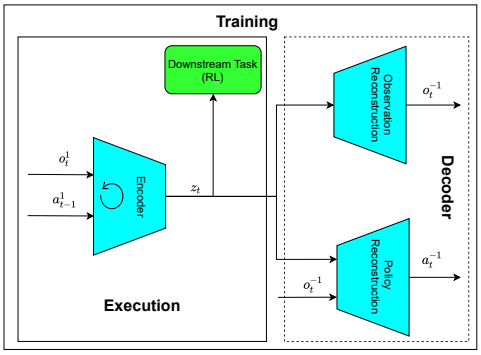


Figure 1: Diagram of LIAM architecture. The two solid-line rectangles show the components of LIAM that are used during training and during execution respectively.

$$\mathcal{L}_{ED} = \frac{1}{H}\sum_{t=1}^{H}[(f^o_{\boldsymbol{u}}(z_t) - o^{-1}_t)^2 - \log f^\pi_{\boldsymbol{u}}(a^{-1}_t | o^{-1}_t, z_t)] \quad \text{where} \;\; z_t = f_{\boldsymbol{w}}(o^1_{:t}, a^1_{:t-1}) \tag{2}$$

We refer to this method as **LIAM** (**L**ocal **I**nformation **A**gent **M**odelling). LIAM uses the information of both the controlled agent and the modelled agent during training, but during execution only the information of the controlled agent is used.

3.3 Reinforcement Learning Training

The latent variables z augmented with the controlled agent's observation can be used to condition the RL optimised policy. Consider the augmented space $\mathcal{O}^1_{aug} = \mathcal{O}^1 \times \mathcal{Z}$, where $\mathcal{O}^1$ is the original observation space of the controlled agent in the POSG, and $\mathcal{Z}$ is the representation space about the agent's models. The advantage of learning the policy on $\mathcal{O}^1_{aug}$ compared to $\mathcal{O}^1$ is that the policy can specialise for different $z \in \mathcal{Z}$. In our experiments we optimised the policy of the controlled agent using A2C, however, other RL algorithms could be used in its place. The input to the actor and critic are the local observation and the generated representation. We do not back-propagate the gradient from the actor-critic loss to the parameters of the encoder. We use different learning rates for optimising the parameters of the networks of RL and the encoder-decoder. We empirically observed that LIAM exhibits high stability during learning, allowing us to use larger learning rate compared to RL. Additionally, we subtract the policy entropy from the policy gradient loss to encourage exploration [Mnih et al., 2016]. Given a batch B of trajectories in the environment, the objective of A2C is:

$$\begin{aligned}\mathcal{L}_{A2C} = \mathbb{E}_{(o_t, a_t, r_{t+1}, o_{t+1}) \sim B}[&\frac{1}{2}(r^1_{t+1} + \gamma V_{\boldsymbol{\phi}}(o^1_{t+1}, z_{t+1}) - V_{\boldsymbol{\phi}}(o^1_t, z_t))^2 \\ &- \hat{A} \log \pi_\theta(a^1_t | o^1_t, z_t) - \beta H(\pi_\theta(a^1_t | o^1_t, z_t))]\end{aligned} \tag{3}$$

The pseudocode of LIAM is given in Appendix A and the implementation details in Appendix E. Intuitively, at the beginning of each episode, LIAM starts with uninformative embeddings "average" over the possible agent policies. At each time step, the controlled agent interacts with the environment and the modelled agent, and updates the embeddings based on the local trajectory that it perceives.

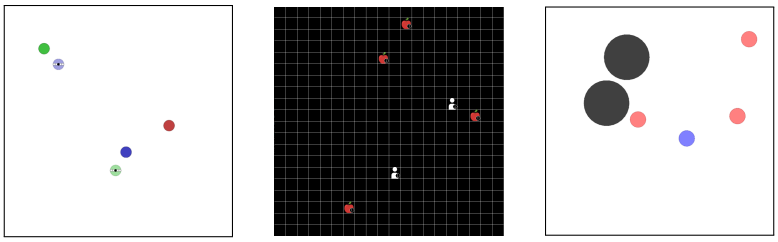


Figure 2: The three evaluation environments. Double speaker-listener (left), level-based foraging (middle), predator prey (right).

4 Experiments

4.1 Multi-Agent Environments

We evaluate the proposed method in three multi-agent environments (one cooperative, one mixed, one competitive): double speaker-listener [Mordatch and Abbeel, 2017], level-based foraging [Albrecht and Stone, 2017, Papoudakis et al., 2020], and a version of predator-prey proposed by [Böhmer et al., 2020]. For each environment, we create ten different policies which are used for training (set Π_k). More details about the process of generating the fixed policies are presented in Appendix B.

Double speaker-listener (DSL): The environment consists of two agents and three designated landmarks. At the start of each episode, the agents and landmarks are generated in random positions, and are randomly assigned one of three possible colours - red, green, or blue. The task of each agent is to navigate to the landmark that has the same colour. However, each agent cannot perceive its colour. The colour of each agent can only be observed by the other agents. Each agent must learn both to communicate a five-dimensional message to the other agent as well as navigate to the correct landmark. The agent's observation includes the relative positions of all landmarks and the other agents as well as the communication message from the previous time step and the colour of the other agent. The cooperative reward at each time step is the negative average Euclidean distance between two agents and their corresponding correct landmark.

Level-based foraging (LBF): The environment is a 20×20 grid-world, consisting of two agents and four food locations. The agents and the foods are assigned random levels and positions at the beginning of an episode. The goal is for the agents to collect all foods. Agents can either move in one of the four directions or attempt to pick up a food. A group of one or more agents successfully pick a food if the agents are positioned in the adjacent cells to the food and if the sum of the agents' levels is at least as high as the food's level. The controlled agent has to learn to cooperate to load foods with a high level and at the same time act greedily for foods that have lower levels. The environment has sparse rewards, representing the contribution of the agent in the gathering all foods in the environment. For example, if the agent receives a food with level 2, and there are another three foods with levels 1, 2 and 3 respectively, the reward of the agent is $2/(1+2+2+3) = 0.25$. The maximum cumulative reward that all agents can achieve is normalised to 1. The environment is partially-observable, and each agent observes up to four grid cells in every direction.

Predator Prey (PP): The environment consists of four agents and two large obstacles. Three of the agents are predators and the other agent is the prey. Each agent has five navigation actions. If only one of the predators captures the prey than all the predators receive reward -1 and the prey reward 1. If two or more predators capture the prey, then the predators receive reward 1 and the prey reward -1. In the experiments, we control the prey, while the predators sample their policies from the set of fixed policies. The environment is partially-observable where the prey can only observe other agents and obstacles that are within its receptive field. The episode terminates after 50 time steps.

4.2 Baselines

Our problem of learning to adapt to different fixed policies can be viewed either as an agent modelling problem or as a single-agent task-adaptation RL problem. We control one agent which has to learn to adapt to multiple tasks, where each one of the fixed policies defines a different task. Thus, task adaptation algorithms can be used as baselines to address our problem. We compare our method

against five baselines, two of which are indicative of the upper and the lower performance of LIAM when it is evaluated against the policies from Π_k. All baselines are trained using the A2C algorithm.

Full Information Agent Model (FIAM): This baseline is indicative of the upper performance of LIAM when it is evaluated against the fixed policies from the set Π_k. FIAM utilises the trajectories of the modelled agent during training as well as during execution and it is trained similarly to LIAM. However, the encoder is conditioned on the actual trajectory of the modelled agent, and therefore requires access to the modelled agent's trajectory during execution.

No Agent Model (NAM): This baseline does not use an explicit agent model, and it is indicative of the lower performance of LIAM. It does, however, use a recurrent policy network which receives as input the observation-action-reward triplet of the controlled agent. By using a recurrent network we can compare returns achieved by implicit agent modelling (that is done by the hidden states of the recurrent network) with the returns achieved by explicit agent modelling that is done by LIAM. NAM is similar to the single-agent task-adaptation algorithm RL^2 [Wang et al., 2016, Duan et al., 2016]. In contrast to the original implementation of RL^2, we reset the hidden state of the recurrent network at the beginning of each episode to ensure consistency among the tested algorithms.

VariBad [Zintgraf et al., 2019]: This baseline trains a VAE to learn latent representations of the observation and reward functions, as they are perceived by the controlled agent, conditioned on the observation, action, reward triplet of the controlled agent. We include VariBad for two reasons: it is an algorithm that learns to adapt to different tasks, and also it uses a recurrent encoder similarly to LIAM (with the exception that the encoder of LIAM does not condition on the reward), but does not utilise the trajectory of the modelled agent. As a result, we can observe whether utilising the modelled agent's trajectory during training, as is done in LIAM, results in higher evaluation returns.

Classification-Based Agent Modelling (CBAM): This baseline learns to reconstruct the identity of the policy that is used by the modelled agent. The policy of the controlled agent is conditioned on the reconstructed identity concatenated with the local observation of the controlled agent. CBAM consists of a recurrent network that receives as input the observation-action sequence of the controlled agent. The output has a softmax activation function, and we train it to maximise the log-likelihood of the identities of the policies that are used by the modelled agent. CBAM uses the fixed policy identities during training (that LIAM does not), but does not use the trajectories of the modelled agent (that LIAM does).

Constrastive Agent Representation Learning (CARL): Finally, we evaluate a non-reconstruction baseline based on contrastive learning [Oord et al., 2018]. CARL utilises the modelled agent's trajectories during training but during execution only the trajectories of the controlled agent are used. Details of the implementation of this baseline are included in Appendix C. This baseline was included because recent works [Laskin et al., 2020, Zhang et al., 2020] in state representation found that non-reconstruction methods tend to perform better than reconstruction methods, especially in domains that have pixel-based observations.

4.3 Evaluation of Returns

Figure 3 shows the average evaluation returns of all methods during training in the three multi-agent environments. The returns of every evaluated method are averaged over five runs with different initial seeds, and the shadowed part represents a 95% confidence interval. We evaluate the policy, learned by each method, every 1000 training episodes for 100 episodes. During the evaluation, the agent follows the stochastic policy that is outputted from the policy network. We found that sampling the action from the policy distribution leads to significantly higher returns compared to following the greedy policy.

First of all, we observe that in all three evaluation environment the returns of LIAM are closer to the upper baseline (FIAM) than to the lower baseline (NAM). The difference in the returns between LIAM and VariBad can be attributed to the fact that LIAM utilises the modelled agent's trajectories during training, leading to better representations for RL. VariBad only learns how the current observation of the controlled agent is related to the next observation and reward, while LIAM learns how the local observation is related to the trajectory of the modelled agent. CARL performs worse compared to LIAM because the generated embeddings contain less information compared to the embeddings generated from LIAM. For example, in the level-based foraging environment, a specific time step t the control agent observes that the modelled agent is at a specific cell. The observation of the modelled

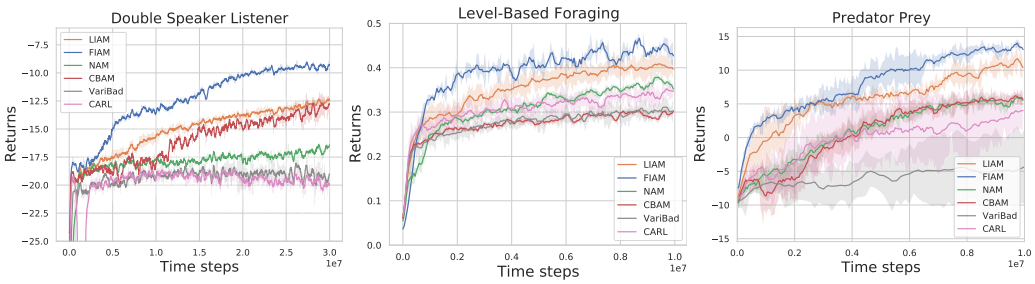


Figure 3: Episodic evaluation returns and 95% confidence interval of the five evaluated methods during training, against policies from Π_k.

agent also contains this type of information, and as a result, the contrastive task can easily relate that these two embeddings are the two parts of a positive pair. Therefore the representation is trained to only encode such information. On the other hand, LIAM encodes into z all information that has been gathered so far in the interaction to be able to reconstruct the modelled agent's trajectory. We observe that LIAM achieves higher return than CBAM because identifying the agent identity is not always an informative representation. LIAM utilises the modelled agent's trajectory during training and learns how the local trajectory corresponds to the trajectory of the modelled agent. In the double-speaker listener environment, CBAM's returns are higher compared to the rest of the baselines and close to the returns of LIAM, because the different fixed policies are discretely different (different communication messages correspond to different colours) and the classifier learns to accurately identify the fixed policy identity. However, in the other environments the classifier is unable to separate different fixed policies, which results in lower average returns.

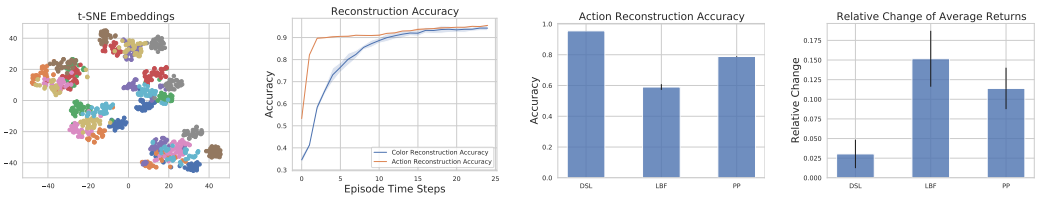
4.4 Model Evaluation

After comparing LIAM against five baselines with respect to the average evaluation returns, we now evaluate the encoder and the decoder of LIAM. First, we visualise the embedding space of LIAM and we evaluate how fast the encoder learns to generate informative embeddings. Finally, we evaluate whether the decoder of LIAM learns to accurately imitate the modelled agent.

We analyse the embeddings learned by LIAM's encoder. Figure 4a presents the two-dimensional projection, using the t-SNE method [Van der Maaten and Hinton, 2008], of the generated embeddings at the 20th time step of the episode for all policies in Π_k in the double speaker-listener environment. We observe that the embeddings belong in five clusters, where each cluster represents the communication message that the modelled agent's policy is following at each episode. Importantly, we observe that embeddings from each agent policy belong to exactly three clusters. This represents the three underlying colours that are observed by the modelled agent. The embedding space presents the 15 different combinations of the three colours and the five possible messages. Note that the embeddings are not explicitly clustered based on the fixed policy identities but based on the selected actions and private observations, which means that different policies may be embedded in the same region.

Figure 4b presents how the action reconstruction accuracy, and the accuracy of the controlled agent identifying its own colour (which is not observed by the controlled agent) in the double speaker-listener environment are changing through time. The colour of the controlled agent is represented as three-dimensional one-hot vector in the observation of the modelled agent, and the controlled agent does not have direct access to this information during their interaction. At each time step t we use the embedding z_t to reconstruct the observation and the action of the modelled agent. In the reconstructed observation we find the three dimensional vector that corresponds to the colour of the controlled agent, and we consider that the identified colour corresponds to the element of the three-dimensional vector that is closer to 1. Then, we compute the percentage of the times where the reconstructed colour matches the colour of that is truly observed by the modelled agent. LIAM learns to identify the modelled agent's action with accuracy greater than 90% after just five time steps, while it learns to identify its own colour with accuracy greater than 90% after 15 time steps.

The decoder of the controlled agent receives as input an embedding at each time step, and predicts the observations and actions of the modelled agent. We evaluate the decoder based on action prediction accuracy. At the 20th time step of the episode, we use the generated embeddings and the decoder to



(a) (b) (c) (d)

Figure 4: (a) t-SNE projection of the embeddings in the double speaker-listener environment, where different colours represent different fixed policies and different points represent different episodes. (b) Modelled agent's actions and controlled agent's colours reconstruction accuracy with respect to the episode time steps. (c) Reconstruction accuracy of the modelled agent's actions at the 20th time step of the episode in the three evaluation environments (the confidence interval is not visible in the double speaker-listener bar plot). (d) Relative change in average returns of the modelled agent using the decoder vs. the fixed policies to select their actions.

reconstruct the modelled agent's actions. Figure 4c presents the average action prediction accuracy and the 95% confidence interval in the three evaluation environments. The reduced action prediction accuracy in level-based foraging and the predator-prey compared to double speaker-listener can be explained by the fact that the controlled agent does not always observe the modelled agents' positions, and as a result it is harder to predict their actions.

Next, since the decoder of LIAM is trained to imitate the fixed agent policies, we can replace the fixed policies with the decoder and compare the returns achieved by the modelled agent. Let G^{-1} be the average return achieved by the modelled agent using the fixed policies, and G^{dec} be the return that the modelled agent achieves using the decoder. The absolute relative change in the returns using the decoder compared to using the fixed policies can be computed as: $c = |(G^{dec} - G^{-1})|/|G^{-1}|$ Figure 4d presents the absolute relative change of the average returns when using the decoder for choosing the modelled agent's actions compared to the returns achieved by the fixed policies in the three evaluation environments. We observe that the average returns of the modelled agent using the decoder are not altered significantly in any of the environments, demonstrating that the decoder learns to accurately imitate the modelled agent's policies.

4.5 Ablation Study

In this section we perform an ablation study to justify the design choices behind the architecture of LIAM. We design and evaluate four different instantiations of LIAM in the double speaker-listener environment. First we evaluate how different data inputs in the encoder of LIAM affect the returns in the double speaker-listener environment. We consider the model LIAM-No-Obs, where in the encoder of LIAM we only input the actions of the controlled agent, and the model LIAM-No-Act, where in the encoder of LIAM we only input the observations of the controlled agent. We also consider an encoder-decoder model that does not utilise the trajectory of the modelled agent during training. We call this model LIAM-Local, and at each time step t the decoder learns to reconstruct the next observation o^1_{t+1} and the action a^1_t conditioned on the embedding z_t. Finally, we evaluate a variational encoder-decoder model [Kingma and Welling, 2014] that performs inference to a Hidden Markov Model (HMM), and we refer to it as LIAM-VAE. LIAM-VAE learns a prior $(p(z_t|z_{t-1}))$ that express the temporal relationship of the latent distribution, similarly to Chung et al. [2015]. We discuss the implementation of this model in Appendix C.

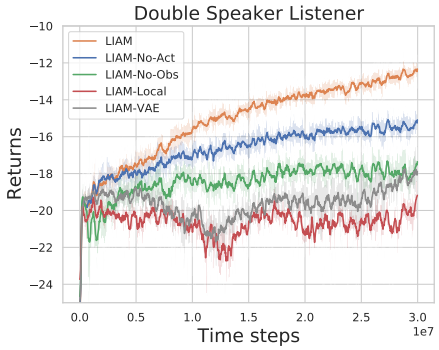
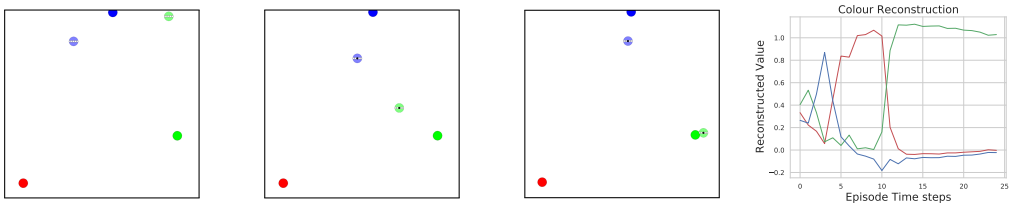


Figure 5: Average returns comparison of LIAM against four ablated versions of LIAM in the double speaker-listener environment.

Figure 5 presents the average returns of LIAM and the aforementioned ablation baselines in the double speaker-listener environment. First, we observe that the lack of either the observation or the



(a) $t = 0$ (b) $t = 7$ (c) $t = 13$ (d)

Figure 6: (a, b, c) Instances of one episode of the double speaker-listener environment at time steps 0, 7 and 13 respectively. (d) Decoder's reconstruction of the colour of the controlled agent.

action of the controlled agent from the encoder of LIAM results in significant lower returns compared to full LIAM. The reason is that the model observes how the modelled agent reacts (by observing the relative position of the modelled agent that is included in the controlled agent's observation) to different communication messages (the communication message is one of the actions of the controlled agent) that the controlled agent outputs, and learns to generate informative embeddings (see also Section 4.6). LIAM-Local achieves the lowest returns compared to the other baselines. The main reason behind this result, is that LIAM-Local does not learn the relationship between the local and the modelled agent's trajectory. Finally, we believe that the main reason that LIAM-VAE performs worse than LIAM is that the KL regularisation is too restrictive, which results in less informative representations compared to LIAM. To alleviate this issue, we can use a β coefficient to weigh the KL loss, similarly to Higgins et al. [2017] or use the MMD^2 distance [Gretton et al., 2007] for regularisation, similarly to info-VAE [Zhao et al., 2017]. In both cases the resulting objective is not a lower bound to the evidence. We did not find any important benefit of using a β coefficient or the MMD^2 distance compared to the simpler instantiation of LIAM that we present in this work.

4.6 Understanding LIAM

To better understand the working mechanisms of LIAM, we analyse how LIAM performs agent modelling in one episode of the double speaker-listener environment. In this episode, the controlled agent has colour green, while the modelled agent has colour blue. The reconstruction of the colour of the controlled agent can be seen as a proxy of the belief about its colour, that the controlled agent encodes into z at each time step. Note that the reconstructed values presented in Figure 6d are not a probability distribution (can be lower than 0 or larger than 1) due to unconstrained optimisation, but we consider that the value closest to 1 represents the belief of the controlled agent about its colour. At the time step 0, all reconstructed colours have similar value close to 0.3 meaning that the controlled agent does not have any information about its colour. The controlled agent outputs different messages and observes how the modelled agent reacts to them through the episode. We observe that at time step 7 the controlled agent believes that its colour is red and starts moving toward the red landmark. After the 11th interaction, the controlled agent has communicated several different messages to the modelled agent and by observing how the modelled agent reacts to them, it understands that its belief about its colour was wrong, and it starts moving toward the correct landmark.

5 Conclusion

We proposed LIAM, the idea of learning models about the trajectories of the modelled agent using the trajectories of the controlled agent. LIAM concurrently trains the model with a decision policy, such that the resulting agent model is conditioned on the local observations of the controlled agent. LIAM is agnostic to the type of interactions in the environment (cooperative, competitive, mixed) and can model an arbitrary number agents. Our results show that LIAM can significantly improve the episodic returns that the controlled agent achieves over baseline methods that do not use agent modelling. The returns that are achieved by A2C when combined with LIAM in some cases nearly match the returns that are achieved by the upper baseline FIAM which assumes access to the trajectories of the modelled agent during execution. In the future, we would like to explore notions of "safety" to handle agents that aim to deceive and exploit the agent model [Shoham et al., 2007, Ganzfried and Sandholm, 2011, 2015].

References

Stefano V. Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2017.

Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 2018.

Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pages 980–991. PMLR, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *International Conference on Machine Learning*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *arXiv preprint arXiv:1506.02216*, 2015.

Dane Corneil, Wulfram Gerstner, and Johanni Brea. Efficient model-based deep reinforcement learning with variational state tabulation. In *International Conference on Machine Learning*, 2018.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semi-parametric regression approach for discovering latent task parametrizations. In *International Joint Conference on Artificial Intelligence*, 2016.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

Sam Ganzfried and Tuomas Sandholm. Game theory-based opponent modeling in large imperfect-information games. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

Sam Ganzfried and Tuomas Sandholm. Safe opponent exploitation. *ACM Transactions on Economics and Computation (TEAC)*, 2015.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, 2019.

Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24, 2005.

Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. *International Conference on Learning Representations*, 2019.

Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, 2007.

Aditya Grover, Maruan Al-Shedivat, Jayesh K Gupta, Yura Burda, and Harrison Edwards. Learning policy representations in multiagent systems. *International Conference on Machine learning*, 2018.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *International Conference on Learning Representations*, 2020.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI Conference on Artificial Intelligence*, 2004.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. *International Conference on Learning Representations*, 2018.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for POMDPs. In *International Conference on Machine Learning*, pages 2122–2131, 2018.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *International Conference on Machine Learning*, 2019.

Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *Advances in neural information processing systems*, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. *arXiv preprint arXiv:1807.09341*, 2018.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

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

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Comparative evaluation of multi-agent deep reinforcement learning algorithms. *arXiv preprint arXiv:2006.07869*, 2020.

Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. *International Conference on Machine Learning*, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. *arXiv preprint arXiv:1802.09640*, 2018.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *International Conference on Machine Learning*, 2019.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *International Conference on Machine Learning*, 2018.

Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural Computation*, 1997.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953.

Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, 2007.

P. Stone, G.A. Kaminka, S. Kraus, and J.S. Rosenschein. Ad hoc autonomous agent teams: collaboration without pre-coordination. In *AAAI Conference on Artificial Intelligence*, 2010.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *International Conference on Autonomous Agents and Multi-Agent Systems*, 2018.

Andrea Tacchetti, H Francis Song, Pedro AM Mediano, Vinicius Zambaldi, Neil C Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W Battaglia. Relational forward models for multi-agent learning. *International Conference on Learning Representations*, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 2008.

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

Annie Xie, Dylan P Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. *Conference on Robot Learning*, 2020.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2019.

Luisa Zintgraf, Sam Devlin, Kamil Ciosek, Shimon Whiteson, and Katja Hofmann. Deep interactive bayesian reinforcement learning via meta-learning. *arXiv preprint arXiv:2101.03864*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Appendix D
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see Appendix E
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]