Neural Amortized Inference for Nested Multi-Agent Reasoning

Kunal Jha[1], Tuan Anh Le[2], Chuangyang Jin[3], Yen-Ling Kuo[4],
Joshua B. Tenenbaum[5], Tianmin Shu[5,6]
[1]Dartmouth College
[2]Google Research
[3]New York University
[4]University of Virginia
[5]Massachusetts Institute of Technology
[6]Johns Hopkins University
kunal.a.jha.24@dartmouth.edu, tuananhl@google.com, cj2133@nyu.edu, ylkuo@virginia.edu, jbt@mit.edu,
tianmin.shu@jhu.edu

Abstract

Multi-agent interactions, such as communication, teaching, and bluffing, often rely on higher-order social inference, i.e., understanding how others infer oneself. Such intricate reasoning can be effectively modeled through nested multi-agent reasoning. Nonetheless, the computational complexity escalates exponentially with each level of reasoning, posing a significant challenge. However, humans effortlessly perform complex social inferences as part of their daily lives. To bridge the gap between human-like inference capabilities and computational limitations, we propose a novel approach: leveraging neural networks to amortize high-order social inference, thereby expediting nested multi-agent reasoning. We evaluate our method in two challenging multi-agent interaction domains. The experimental results demonstrate that our method is computationally efficient while exhibiting minimal degradation in accuracy.

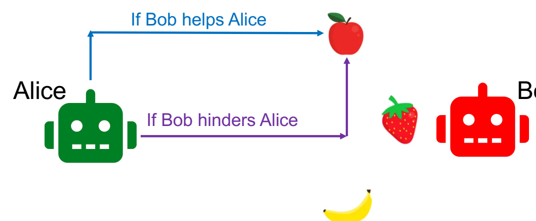

Figure 1: Illustration of nested reasoning between two agents. Alice wants to get the apple and Bob may help or hinder Alice. Agents do not know each other's goals. It is crucial that Alice successfully infers Bob's intent so that Alice can choose the optimal path to either cooperate with Bob or compete with Bob. Such inference requires nested reasoning – in this case, Alice infers how Bob infers Alice.

Introduction

We reason about and act in a world that contains not only inanimate objects but also agents. Reasoning about other agents is key to our everyday abilities to empathize, communicate, teach, and more. Imbuing AI systems with adequate social inference is key to enabling rich multi-agent interactions. For instance, as illustrated in Figure 1, to successfully interact with another agent, one would need to infer not only the agent's goal but also how the other agent infers the goal of oneself. Current multi-agent systems still cannot conduct nested multi-agent reasoning efficiently and flexibly.[1]

There have been prior works attempting to build agents that can conduct nested reasoning with each other to achieve sophisticated interactions. Most notably, Interactive Partially Observable Markov Decision Making Process (I-POMDP) (Gmytrasiewicz and Doshi 2005) provides a framework for modeling multi-agent systems with nested reasoning. While the framework is general and can be applied to a broad range of domains, we still cannot equip autonomous agents with human-like nested reasoning to date. One of the main challenges is that inference under the I-POMDP framework is recursive and hence slow. In complex domains in which there are large state, action, and goal spaces, solving I-POMDP becomes computationally intractable.

However, recursively reason effortlessly in our daily life despite the complexity of the inference. This work aims to develop an efficient inference method for nested multi-agent reasoning. We follow the I-POMDP framework and propose to amortize inference at different levels of the I-POMDP by training neural networks to sample data-driven proposals. This allows us to sample a small number of hypotheses to cover the ground truth, which drastically decreases the overall computation required without sacrificing accuracy.

We demonstrate our approach in two complex domains requiring higher-order social reasoning. In the first domain, we consider a two-player game in 2D grid worlds, Construction, in which one agent (Bob) needs to help or hinder another agent (Alice) but the two agents do not know each other's goal. To successfully infer the goals of Bob, one would need nested goal inference. Additionally, we evaluate our approach in a driving domain, in which drivers must infer each other's mental states (including goals and beliefs about the physical states) recursively to avoid collision and to signal danger to other drivers in anticipation of a crash.

[1]The code and the supplementary material are available at https://www.tshu.io/nested_reasoning.

To successfully predict drivers' actions, a model must conduct nested goal and belief inference. Our experimental results show that, in both domains, the amortized inference greatly reduces the total amount of compute while maintaining a similar inference accuracy compared to conventional approaches. It can also estimate the uncertainty in inference and generalize well to unseen scenarios.

Related Work

Nested Multi-agent Reasoning. There have been prior works formulating nested reasoning between multi-agents. The most general formulation is the interactive partially observable Markov decision process (I-POMDP), proposed by Gmytrasiewicz and Doshi (2005). Additionally, there have been models that address special cases in nested multi-agent reasoning. For instance, cooperative inverse reinforcement learning (CIRL) models human-robot cooperation in which a user needs to infer how a robot assistant infers the user's reward (Hadfield-Menell et al. 2016). Frank and Goodman (2012) model a type of nested reasoning between a speaker and a listener, pragmatic reasoning, with the Rational Speech Act (RSA) modeling framework. Tejwani et al. (2022) proposes a Social MDP framework to model complex social interactions that rely on nested reasoning about the reward functions of other agents. While these are all powerful frameworks, there has not been much work on developing efficient inference algorithms for these frameworks in complex domains with large hypothesis spaces on each level. Conventional methods (Rathnasabapathy, Doshi, and Gmytrasiewicz 2006; Doshi and Gmytrasiewicz 2009; Seaman, van de Meent, and Wingate 2018) can conduct explicit nested reasoning robustly but fail to scale to complex environments. Recently, Han and Gmytrasiewicz (2019) proposed an end-to-end model to approximate the nested belief update as hidden state updates in a neural network to train multi-agent policies. However, such end-to-end models cannot conduct explicit nested reasoning with explainable goals and belief representations (e.g., they can generate actions given agents' goals but cannot infer agents' goals based on observed actions). They also cannot estimate uncertainty in inference. This work aims to fill in the gap by proposing an efficient, explainable, and flexible inference method.

Theory of Mind Reasoning. More broadly, nested multi-agent reasoning is a type of Theory of Mind reasoning (Premack and Woodruff 1978), in which agents must infer each other's mental states based on observed actions. There are two main types of computational models for Theory of Mind: end-to-end methods based on neural networks, such as Machine Theory of Mind network (Rabinowitz et al. 2018; Chuang et al. 2020), and model-based methods relying on generative models of agents such as Bayesian Theory of Mind (Baker et al. 2017; Ullman et al. 2009; Shum et al. 2019; Netanyahu et al. 2021). Our approach combines both types to achieve fast (through neural networks) yet robust (through model-based reasoning) inference.

Neural Amortized Inference. There has been prior work on neural amortized inference for accelerating probabilistic inference in complex domains (Le, Baydin, and Wood 2017), such as computer graphics (Ritchie et al. 2016) and particle physics (Baydin et al. 2019). These works demonstrated that neural networks can sample data-driven proposals more likely to include the ground-truth hypothesis. Consequently, the inference algorithms would only need to maintain a small set of particles to achieve high accuracy. In this paper, we adopt the idea of neural amortized inference to accelerate the inference following I-POMDP formulation as a general solution for nested multi-agent reasoning.

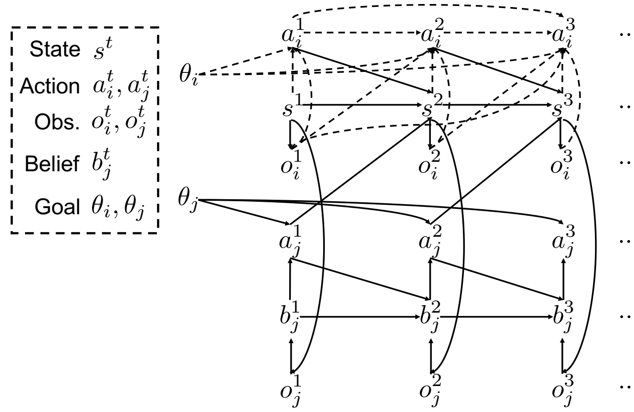

Figure 2: Graphical model for I-POMDP where an agent i plans its actions while modeling another agent j. Dashed arrows describe the *planner* that outputs the probabilities of i's actions. The solid arrows describe the dynamics of the environment and agent j that the planner is based on.

Background

I-POMDP

Interactive Partially Observable Markov Decision Process (I-POMDP) extends POMDP by recursively modeling other agents in the environment (Gmytrasiewicz and Doshi 2005). For rotational convenience and without loss of generality, we assume that there are two agents, i and j. Agent i is the ego agent, which models and interacts with agent j.

As illustrated in Figure 2, an I-POMDP model states $s^{1:T}$[2], state observations of the two agents $o_i^{1:T}, o_j^{1:T}$, actions of the two agents $a_i^{1:T}, a_j^{1:T}$; for agent j, we additionally model its beliefs $b_j^{1:T}$ and other information about its mind that is relevant to its behavior, θ_j. In this work, we define θ_i as agent i's goal. But this can be extended to other types of information such as preferences.

Following Doshi and Gmytrasiewicz (2009), we inductively define i's interactive state $is_{i,\ell}$ at level-ℓ as

Level 0: $is_{i,0} = s$

Level 1: $is_{i,1} = (s, b_{j,0}, \theta_j)$ where $b_{j,0}$ is a *distribution* over j's interactive state at level 0, $is_{j,0}$

...

Level ℓ: $is_{i,\ell} = (s, b_{j,\ell-1}, \theta_j)$ where $b_{j,\ell-1}$ is a *distribution* over j's interactive state at the previous level, $is_{j,\ell-1}$

This introduces a *generative* model for agents' behavior conditioned on their nested reasoning, $p(is_{i,\ell}^{1:T}, o_i^{1:T}, a_i^{1:T})$.

[2] T is the total number of steps.

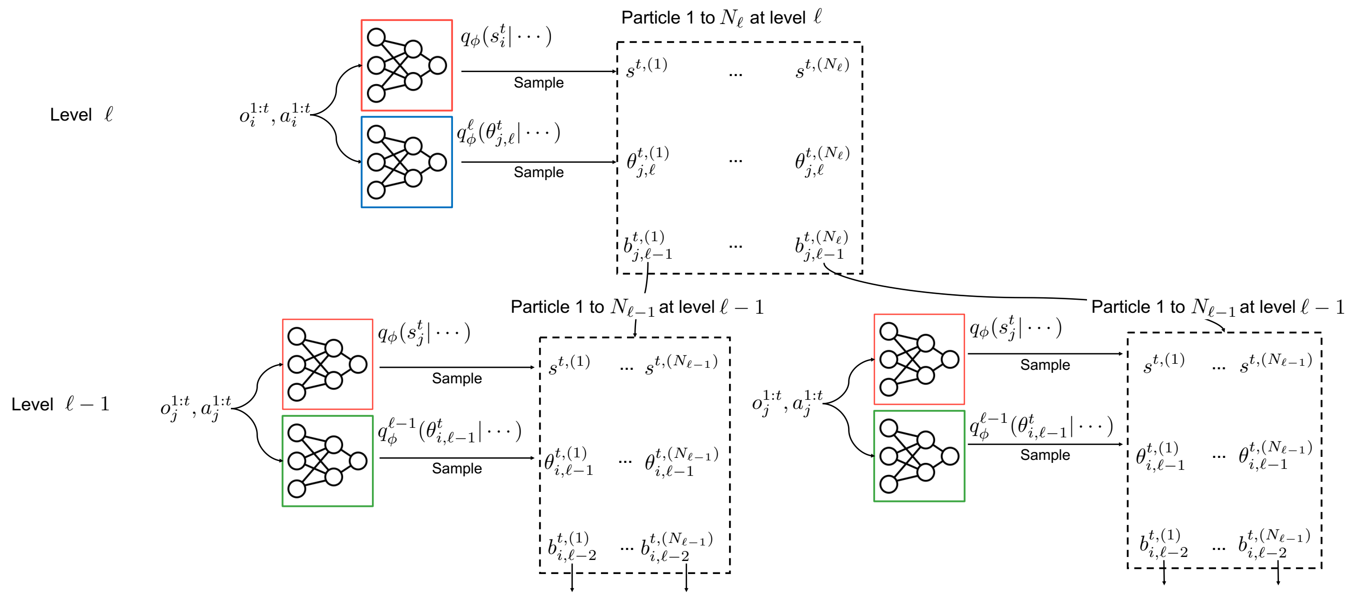

Figure 3: Illustration of particle sampling at each level using the recognition networks recursively. Networks with the same colors are the same. The two branches at level $\ell-1$ have the same design because each branch represents a sampled belief, $b_{j,\ell-1}^{t,(n)}$, which is a distribution of j's interactive state at level $\ell-1$. In each branch, we approximate $b_{j,\ell-1}^{t,(n)}$ by sampling a set of particles at level $\ell-1$ using the *same* recognition network for level $\ell-1$ (i.e., $q_\theta^{\ell-1}$).

Inference. The level-ℓ agent i performs inference to obtain the belief $b_{i,\ell}^t := p(is_{i,\ell}^t|o_i^{1:t}, a_i^{1:t-1})$. i's interactive state at level ℓ (i.e., $is_{i,\ell}^t = (s^t, b_{j,\ell-1}^t, \theta_j)$) contains j's belief at level $\ell-1$ (i.e., $b_{j,\ell-1}^t = p(is_{j,\ell-1}^t|o_j^{1:t}, a_j^{1:t-1})$). Thus, inference at level ℓ depends on inference at level $\ell-1$ which depends on inference at level $\ell-2$, and so on. This recursion terminates at level 0 in which the inference is the same as in a POMDP. That is, the belief is $b_{i,0} = p(s^t|o_i^{1:t}, a_i^{1:t-1})$, disregarding the other agent.

Planning. The approximate planner of a level-ℓ agent i determines its policy given its belief $b_{i,\ell}^t$ and i's goal θ_i. Inference in I-POMDP requires planning. Given observations $o_i^{1:t}$ and previous actions $a_i^{1:t-1}$, the full posterior under the generative model in Figure 2 contains not only the interactive state $is_{i,\ell} = (s, b_{j,\ell-1}, \theta_j)$ but also j's actions and observations, $a_{j,\ell-1}, o_{j,\ell-1}$, which need to be marginalized out. Marginalizing actions requires evaluating j's policy at level $\ell-1$, $\pi_{j,\ell-1}(a_{j,\ell-1}^t|b_{j,\ell-1}^t, \theta_j)$. As illustrated in Figure 2, planning also requires inference. For example, agent i needs to simulate how its own actions may change agent j's belief in order to interact with agent j optimally.

Amortized Inference for I-POMDPs

Since planning and inference at level ℓ depend on the nested inference of beliefs at lower levels, it can become prohibitively expensive to reason about agents with high ℓ. Hence, we propose to *amortize inference* at each level. At level ℓ, we learn a neural recognition model $q_\phi^\ell(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T})$ parameterized by ϕ to approximate $p(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T})$ by minimizing the KL divergence between exact inference and the recognition model on data sampled from the generative model $p(is_{i,\ell}^{1:T}, o_i^{1:T}, a_i^{1:T})$:

$$\mathcal{L}(\phi,\ell) = \mathbb{E}\left[\mathrm{KL}(p(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T})||q_\phi^\ell(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T}))\right] \tag{1}$$

Sampling at level ℓ requires inference at level $\ell-1$. To accelerate the sampling, we amortize the lower level inference using a previously trained recognition model at that level, $q_\phi^{\ell-1}(is_{j,\ell-1}^{1:T}|o_j^{1:T}, a_j^{1:T})$. At level-0, we learn a recognition model over states $q_\phi^0(s_{1:T}|o_i^{1:T}, a_i^{1:T})$.

The algorithm for training a set of recognition models, $q_\phi^0, \ldots, q_\phi^\ell$, is outlined as follows:

- Train a recognition model $q_\phi^0(s_{1:T}|o_{1:T}, a_{1:T})$ by generating data from $p(s^{1:T}, o_i^{1:T}, a_i^{1:T})$ and minimizing $\mathcal{L}(\phi, 0)$.
- for levels $\ell = 1, \ldots, L$
  - Train $q_\phi^\ell(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T})$ by generating data from $p(is_{i,\ell}^{1:T}, o_i^{1:T}, a_i^{1:T})$ and minimizing $\mathcal{L}(\phi,\ell)$.

We describe how we generate the training data in the supplementary material.

Factorization of the recognition model. We factorize the recognition model autoregressively:

$$q_\phi^\ell(is_{i,\ell}^{1:T}|o_i^{1:T}, a_i^{1:T}) = \prod_{t=1}^{T} q_\phi^\ell(is_{i,\ell}^t|is_{i,\ell}^{t-1}, o_i^{1:t}, a_i^{1:t-1}). \tag{2}$$

This allows us to approximate the belief at any step t, $b_{i,\ell}^t = p(is_{i,\ell}^t | o_i^{1:t}, a_i^{1:t-1})$. We further factorize the recognition model at each step over beliefs, states, and goals:

$$q_\phi^\ell(is_{i,\ell}^t | is_{i,\ell}^{t-1}, o_i^{1:t}, a_i^{1:t-1}) = q_\phi^\ell(b_{j,\ell-1}^t | is_{i,\ell}^{t-1}, o_i^{1:t}, a_i^{1:t-1}) \cdot q_\phi(s^t | is_{i,\ell}^{t-1}, o_i^{1:t}, a_i^{1:t-1}) \cdot q_\phi^\ell(\theta_j | is_{i,\ell-1}^{t-1}, o_i^{1:t}, a_i^{1:t-1}). \tag{3}$$

One remaining challenge is parameterizing the distribution over beliefs at the lower level $\ell - 1$, $b_{j,\ell-1}^t$. For this, we first examine the distribution over $b_{j,\ell-1}^t$ under the *prior*, $p(b_{j,\ell-1}^t | o_{j,\ell-1}^t, b_{j,\ell-1}^{t-1}, a_{j,\ell-1}^{t-1})$, following the factorization in Figure 2. Under the prior, this distribution is a *deterministic* belief-update function of the current observation, previous belief, and previous action. Since such belief updates are in general intractable, we represent beliefs as a set of N weighted samples (or particles), $\{(is_{j,\ell-1}^{t,(n)}, w_{j,\ell-1}^{t,(n)})\}_{n=1}^{N_\ell}$. We perform a particle update, $b_{j,\ell-1}^t = \text{ParticleUpdate}(o_{j,\ell-1}^t, b_{j,\ell-1}^{t-1}, a_{j,\ell-1}^{t-1})$, at each step. In this paper, we set the recognition distribution over $b_{j,\ell-1}^t$ to be *identical to the prior*. That is, to sample from the recognition model $q_\phi(b_{j,\ell-1}^t | o_{j,\ell-1}^t, b_{j,\ell-1}^{t-1}, a_{j,\ell-1}^{t-1})$, we perform the particle update above. By doing so, during importance sampling described below (Eq. (4)), the ratio $p(b_{j,\ell-1}^t | \cdots) / q_\phi(b_{j,\ell-1}^t | \cdots)$ conveniently becomes one.

In sum, we need to train a recognition model for state inference, $q_\phi(s^t | \cdots)$, shared by all levels; additionally, for *each* level, we train a recognition model for goal inference at that level, $q_\phi^\ell(\theta_j^t | \cdots)$.

Importance sampling. As Figure 3 illustrates, we sample N_ℓ particles based on the recognition networks at each level ℓ at time t. We can then compute the importance weight for a particle at level ℓ at time t as

$$w_t = \frac{p(is_{i,\ell}^{1:t}, o_i^{1:t} | a_i^{1:t-1})}{q_\phi^\ell(is_{i,\ell}^{1:t} | o_i^{1:t}, a_i^{1:t-1})} = \frac{\sum_{a_j^{1:t-1}} p(s^{1:t} | a_i^{1:t-1}, a_j^{1:t-1}) \pi_{j,\ell-1}(a_j^{1:t-1} | b_{j,\ell-1}^{1:t-1}, \theta_{j,\ell})}{q_\phi(s^{1:t} | o_i^{1:t}, a_i^{1:t-1}) q_\phi^\ell(\theta_j | o_i^{1:t}, a_i^{1:t-1})} \tag{4}$$

We use this weight to refine the posterior approximated by the recognition model. We show the derivation of Eq. (4) in the supplementary material. The algorithm for approximating the belief $b_{i,\ell}^t$ using importance sampling using N_ℓ samples is

- For sample $n = 1, \ldots, N_\ell$
  - For time $\tau = 1, \ldots, t$
    - Sample $is_{i,\ell}^{\tau,(n)} \sim q_\phi^\ell(\cdot | is_{i,\ell}^{\tau-1,(n)}, o_i^{1:\tau}, a_i^{1:\tau-1})$
  - Evaluate $w_t^{(n)}$ from (4).

The set of weighted samples $\{(is_{i,\ell}^{t,(n)}, w_t^{(n)})\}_{n=1}^{N_\ell}$ is used to approximate the posterior belief $b_{i,\ell}^t$. To approximate a sequence of beliefs from 1 to T, we re-run this importance sampling procedure at every time step. We found that this performs better than sequential Monte Carlo because the quality of the samples from the recognition models drastically increases as the models observe more steps. Re-running importance sampling at every step ensures that our method fully utilizes the higher-quality samples produced by the recognition models at later steps.

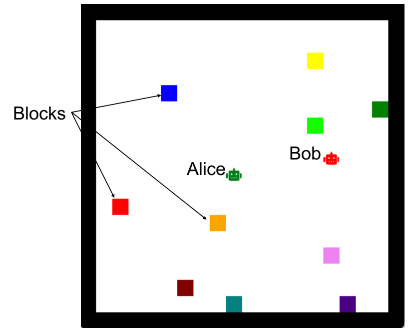

Figure 4: An example environment in Construction.

Experiments

Construction Environment

Setup Inspired by Ullman et al. (2009) and Tejwani et al. (2022), we evaluate our method in a 2D grid-world domain, **Construction**, as illustrated in Figure 4. There are two agents in this domain: *Alice* (a level-0 agent), whose goal is to put two of the blocks next to each other, and *Bob* (a level-1 agent), whose goal is to either help or hinder Alice. Both agents can observe the full state and each other's actions but do not know each other's goals. Each agent can move in 4 directions. Once an agent is on top of a block, it automatically picks it up. When the agent carries a block, it can put it down at any step. At each step, a model is asked to infer Bob's goal (helping or hindering) from a level-2 third-person observer's perspective, based on the observed actions of Alice and Bob up until that step, i.e., online inference of Bob's goal. Since Bob is helping or hindering Alice by also inferring her goal, a successful online goal inference must also infer how Bob reasons about Alice's goal. In each episode, the two agents and ten colored blocks are randomly spawned in 20 × 20 grid. There are 45 possible goals for Alice and two possible goals for Bob. The hypothesis space in this domain (90 hypotheses in total) is much larger than that of prior works. For instance, there are only 2 - 4 hypotheses in Ullman et al. (2009) and Tejwani et al. (2022).

Implementation We implement breath-first-search for both agents' planners and use exact inference for I-POMDP (by enumerating all possible goal hypotheses) for both agents at each level to synthesize the actions of the agents. This allows us to create two training sets – S1 with Alice acting alone and S2 with Bob interacting with Alice. S1 and S2 correspond to the training set for level-1 inference and level-2 inference, respectively. We also synthesized 100 testing episodes of Alice and Bob interacting with each other. More details are included in the supplementary material.

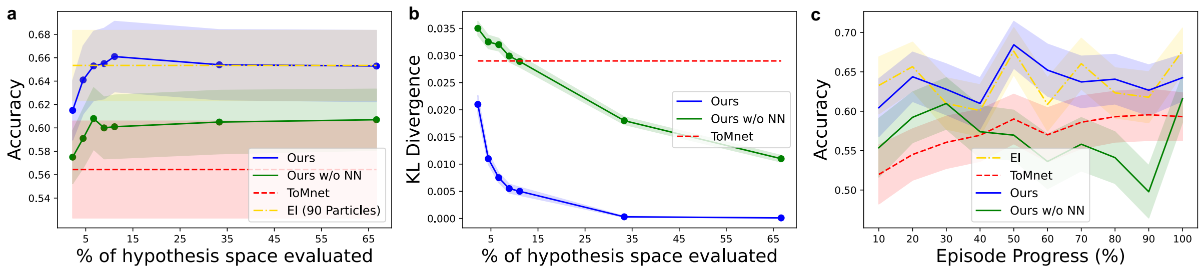

Figure 5: Online goal inference performance in Construction. (a) Accuracy of all steps over the number of particles (in terms of the percentage of all 90 hypotheses). (b) KL-divergence between model inference and exact inference. (c) Averaged goal inference accuracy over the progress of an episode. Ours and Ours w/o NN use 10 particles. The shaded areas show the standard errors of different methods. Note that we omit the standard error of ToMnet in (b) since it is too large for the figure (0.2).

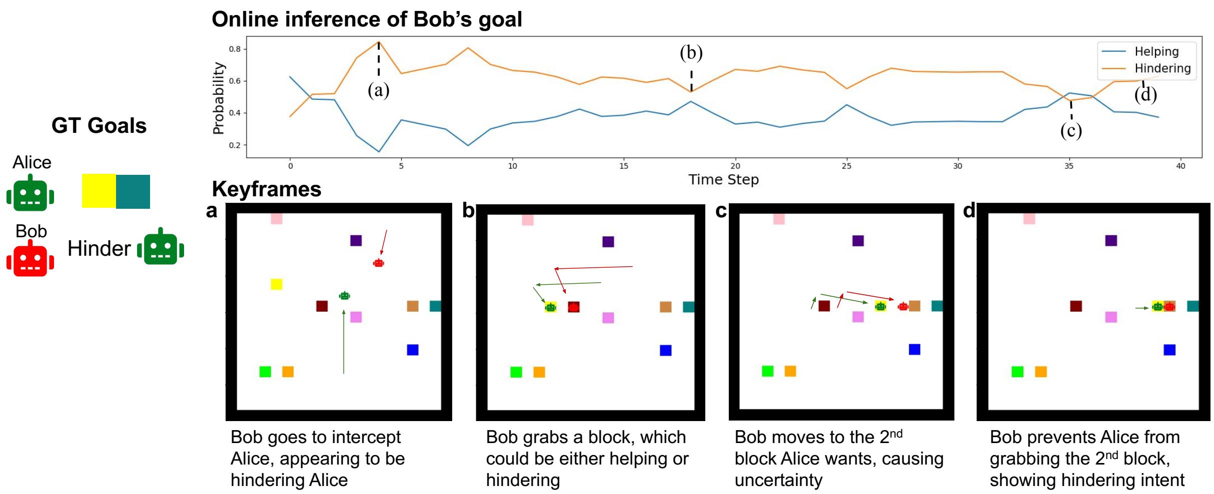

Figure 6: Online goal inference of our method (with 10 particles) in a typical episode in Construction, in which Alice wants to put yellow and teal blocks together and Bob tries to hinder Alice. The plot on the top shows the posterior probabilities of the two hypotheses based on our method's inference at any given time step. The keyframes on the bottom explain why our method adjusts its inference. Note that the arrows in the frames show the trajectories of the agents.

Results We compare our method (**Ours**) against the following baselines.

ToMnet: We adopt ToMnet proposed in Rabinowitz et al. (2018) to infer Bob's goal. The network is trained with cross-entropy loss.

Ours$_{\text{-NN}}$: This baseline uses a uniform distribution as the proposal distribution instead of neural amortized inference.

We also show the exact inference (**EI**) performance, in which we enumerate all possible hypotheses to solve I-POMDP. Note that for Ours and Ours$_{\text{-NN}}$, we amortize the level-1 goal inference by sampling a small number of particles to propose possible goals for Alice and enumerate all 2 possible goals for level-2 goal inference (Bob's goal).

We report the averaged goal inference accuracy over the number of sampled particles in terms of the percentage of the full hypothesis space (90 hypotheses of both agents' goals) in Figure 5**a**. The results show that our method's inference accuracy becomes comparable to the exact inference with only 6 particles (6.7% of the hypothesis space). It is much more efficient compared to the exact inference, which needs 90 particles. With uniform proposals (Ours$_{\text{-NN}}$), on the other hand, the inference accuracy is much lower than the full model that utilizes the data-driven proposals from the recognition network. This demonstrates that neural amortized inference can sample high-quality hypotheses to reduce the computation necessary to produce accurate inference drastically. Interestingly, the inference accuracy of the level-1 goal recognition network is only around 17%. This suggests that we do not need to train a highly accurate network as long as the proposal distribution is better than the uniform distribution. In contrast, ToMnet achieves poor accuracy, as it neither evaluates hypotheses with planning nor explicitly reasons about how the level-1 agent infers the level-0 agent's goals.

The accuracy of our method maintains at a similar level after using more than 6 particles but has a more accurate estimation of the uncertainty of the hypotheses. This is reflected in the lower KL divergence between our method's inference and the exact inference when there are more particles as shown in Figure 5**b**. When considering more hypotheses, the model will likely discover alternative hypotheses that can explain the observed behavior equally well.

Figure 5**c** demonstrates how each method's online goal inference accuracy changes over time. Specifically, we plot the averaged accuracy across all testing episodes as a function of what percentage of an episode a method has seen. The result of Ours is based on 10 particles. The inference of all methods becomes more accurate as they observe more actions. However, the accuracy of ToMnet increases more slowly and reaches a lower plateau compared to EI and Ours.

Figure 6 illustrates a typical example of the online goal inference conducted by our method. It shows that our method not only correctly infers the goal, but can also adjust the certainty in inference dynamically by evaluating alternative hypotheses. For instance, when Bob grabs or moves toward the 2nd block (frames (**b**) and (**c**) in Figure 6), he could try to help Alice by delivering the block to her; he could also try to hinder Alice by moving it away from her. At these moments, our model decreases its certainty. However, with further observation, our model gradually increases its confidence again as the behavior shows a clearer hindering intent. Such uncertainty estimation is key to the robustness of multi-agent nested reasoning, which can be achieved by only sampling a small number of particles with our method. We include additional results in the supplementary material.

Driving Environment

Setup For the second experiment, we simulate the traffic at an intersection as shown in Figure 7 using a commonly used driving environment, CARLO (Cao et al. 2020). In this environment, we randomly assign a goal (forward, left-turn, or right-turn) to a driver, indicating the destination after the intersection. A car can be controlled through 5 actions at each step: accelerating, braking, rotating left, rotating right, and signaling danger to other drivers by honking. Each driver partially observes the world and cannot see cars outside their field of view or obstructed by other cars or buildings. To avoid crashing into other cars, a driver must recursively infer other drivers' mental states (goals and beliefs about the states) and consequently predict other drivers' future actions. When a driver infers that another driver is unaware of a nearby car, it must signal danger by honking to avoid a potential crash. The task for a model is to predict the next action of a car from a third-person observer's perspective, which requires a level-2 inference in this environment.

Implementation For each driver's belief, we represent the world state as the states of the cars in all 8 lanes. For the k-th lane, we model up to 2 cars that are closest to the intersection: $\{\langle e_{k,m}, (x_{k,m}, y_{k,m}), (\sigma_{k,m}, v_{k,m})\rangle\}_{m=1,2}$. The indices of cars, m, are ordered by their distances to the intersection. $e_{k,m} \in \{0, 1\}$ indicates whether a car exists in the k-th lane. If a car exists ($e_{k,m} = 1$), $(x_{k,} \in \mathbb{R}, y_{k,m} \in \mathbb{R})$ indicates its location, we use $\sigma_{k,m} \in [-\pi, \pi]$ to indicate its heading angle and $v_{k,m} \in [0, v_{\text{max}}])$ to indicate its speed (v_{max} is the maximum speed).

All drivers' policies are based on a hierarchical planner.

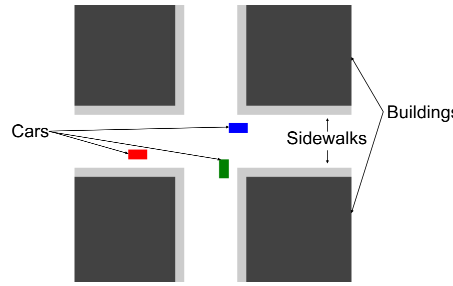

Figure 7: An example environment in Driving. The dark blocks are buildings and the light gray regions are sidewalks.

The planner first decides whether to move, stop, or signal danger at each step; if it decides to move, it then selects the action (accelerate, rotate left, or rotate right) that follows the shortest path; if it decides to stop, then it will brake; otherwise, it will signal danger by honking. We use exact inference for I-POMDP to synthesize driver behavior and create three training sets: S0 with level-0 drivers using exact state inference (for training the state recognition model); S1 with level-0 drivers using amortized state inference (for training the level-1 goal recognition model); and S2 with level-1 drivers using amortized level-1 inference (for training the level-2 goal recognition model). In all training sets, there are 3 cars. For evaluation, we synthesize 100 testing episodes of 3 cars interacting with each other. Additionally, we create two generalization testing sets: i) 4 cars and ii) 3 cars where one of the cars is controlled by an inattentive driver who is not paying attention to other cars. Each set has 100 episodes. To predict a driver's actions, we use level-1 agent policy, i.e., $\pi_{i,1}(a_i^t | b_{i,1}^t, \theta_i)$. Additional implementation details are in the supplementary material.

Results Similar to **Construction**, we compare our method (Ours) against EI, ToMnet, and Ours$_{\text{-NN}}$. Note that ToMnet here is trained to predict the action of each car.

Figure 8 shows that our method significantly outperforms baselines. With only 9 particles (12.5% of the hypothesis space), our method's action prediction accuracy is already comparable to the exact inference (which requires 72 particles[3]). Interestingly, unlike our method and the exact inference, ToMnet's accuracy drops drastically over time in (Figure 8**c**). This is because later in an episode, cars are closer to each other and thus have to carefully coordinate with each other based on nested reasoning. This further demonstrates this domain's difficulty and the necessity of robust nested reasoning in understanding and predicting complex multi-agent interactions.

We further evaluate the generalizability of different methods. In particular, all methods are trained using episodes with 3 cars controlled by normal drivers. We test the methods on episodes with 4 cars (Figure 9**ab**) and on episodes with 3 cars in which one of the drivers is inattentive (Figure 9**cd**). In both cases, our method still performs significantly better than baselines. In Figure 9**c**, our method's ac-

[3]We explain why it needs 72 particles in the supplementary material.

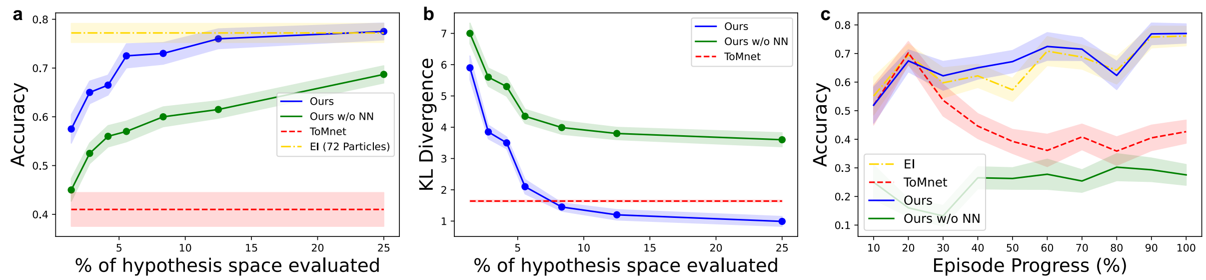

Figure 8: Action prediction performance in Driving. (a) Action prediction accuracy in a 3-car scenario. (b) KL-divergence between model inference and exact inference. (c) Averaged action prediction accuracy over the progress of an episode. Ours and Ours w/o NN all use 9 particles.

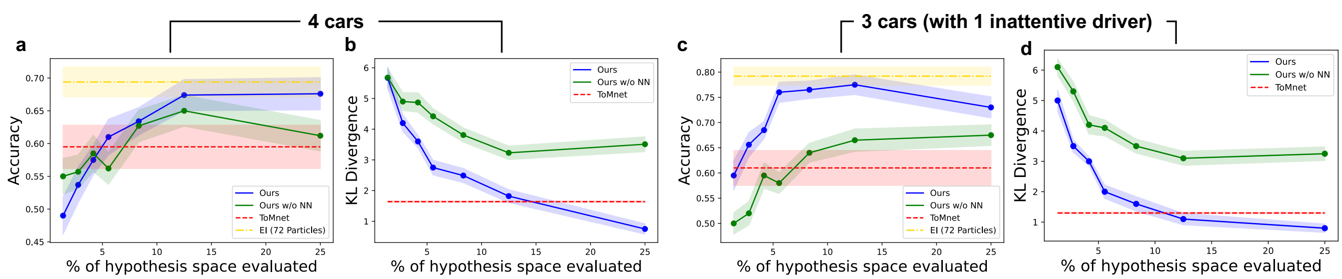

Figure 9: Generalization evaluation results of models trained with 3 cars controlled by normal drivers. (a)(b) Results on episodes with 4 cars. (c)(d) Results on episodes with 3 cars where one of the drivers is inattentive.

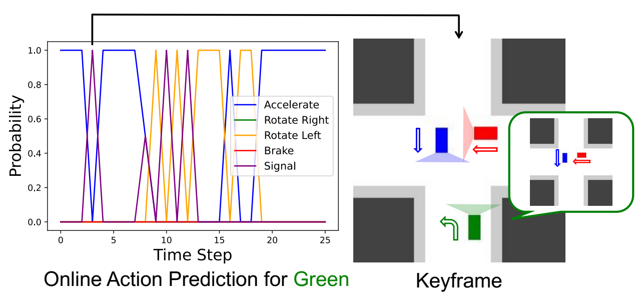

Figure 10: Online action prediction of our method (with 9 particles) in a typical episode in Driving. The plot on the left shows our method's prediction of green's action. The keyframe on the right explains why our method predicts that green will signal danger at the highlighted step. Note that the colored cones show the drivers' fields of view, and the arrows show the predicted goals. The model infers that i) green wants to turn left and that ii) green infers that red wants to go forward and is unaware of green (as shown in the smaller frame). Our method correctly predicts that green will signal danger to alert the red car to green's presence.

curacy drops a bit when using 25% particles. This is because it estimates more hypotheses and consequently decreases its confidence in prediction. However, the KL-divergence becomes lower when the model uses more particles (Figure 9**d**), making the uncertainty estimation more accurate.

Figure 10 depicts a typical example of online action prediction by our method (with 9 particles) in the 3 normal cars condition. By inferring how green infers red's goal and belief, our method correctly anticipates green's action (i.e., signaling danger). Such prediction is made possible by sophisticated nested reasoning. In fact, due to the lack of robust nested reasoning, ToMnet consistently fails to predict any "signal" action correctly. We include additional examples in the supplementary material.

Conclusion

In this work, we propose a neural amortized inference approach to accelerate nested multi-agent reasoning. We evaluate our method in two complex multi-agent domains with large hypothesis spaces. The results demonstrate that our method can significantly improve the efficiency of nested reasoning while maintaining a high level of accuracy. In addition, our method can also estimate the uncertainty in its inference and generalize to unseen scenarios.

In the current experiments, we only amortize up to level-2 reasoning. However, the recursive nature of our method can amortize higher-order reasoning as well, which we intend to study in future work. We only need to train one recognition network for each additional level. We evaluate level-2 reasoning in our experiments because it is sufficient for the majority of the real-world social reasoning that we encounter in our daily lives. Determining the minimum level required to understand multi-agent interaction in a given domain is also an open question. One possibility is to amortize the inference of the necessary level through meta-learning.

Acknowledgements

This work was supported by the DARPA Machine Common Sense program, ONR MURI N00014-13-1-0333, and a grant from Lockheed Martin.

References

Baker, C. L.; Jara-Ettinger, J.; Saxe, R.; and Tenenbaum, J. B. 2017. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1(4): 0064.

Baydin, A. G.; Shao, L.; Bhimji, W.; Heinrich, L.; Meadows, L.; Liu, J.; Munk, A.; Naderiparizi, S.; Gram-Hansen, B.; Louppe, G.; et al. 2019. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 1–24.

Cao, Z.; Biyik, E.; Wang, W. Z.; Raventos, A.; Gaidon, A.; Rosman, G.; and Sadigh, D. 2020. Reinforcement Learning based Control of Imitative Policies for Near-Accident Driving. In *Proceedings of Robotics: Science and Systems (RSS)*.

Chuang, Y.-S.; Hung, H.-Y.; Gamborino, E.; Goh, J. O. S.; Huang, T.-R.; Chang, Y.-L.; Yeh, S.-L.; and Fu, L.-C. 2020. Using machine theory of mind to learn agent social network structures from observed interactive behaviors with targets. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1013–1019. IEEE.

Doshi, P.; and Gmytrasiewicz, P. J. 2009. Monte Carlo sampling methods for approximating interactive POMDPs. *Journal of Artificial Intelligence Research*, 34: 297–337.

Frank, M. C.; and Goodman, N. D. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084): 998–998.

Gmytrasiewicz, P. J.; and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24: 49–79.

Hadfield-Menell, D.; Russell, S. J.; Abbeel, P.; and Dragan, A. 2016. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29.

Han, Y.; and Gmytrasiewicz, P. 2019. Ipomdp-net: A deep neural network for partially observable multi-agent planning using interactive pomdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6062–6069.

Le, T. A.; Baydin, A. G.; and Wood, F. 2017. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, 1338–1348. PMLR.

Netanyahu, A.; Shu, T.; Katz, B.; Barbu, A.; and Tenenbaum, J. B. 2021. Phase: Physically-grounded abstract social events for machine social perception. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, 845–853.

Premack, D.; and Woodruff, G. 1978. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4): 515–526.

Rabinowitz, N.; Perbet, F.; Song, F.; Zhang, C.; Eslami, S. A.; and Botvinick, M. 2018. Machine theory of mind. In *International conference on machine learning*, 4218–4227. PMLR.

Rathnasabapathy, B.; Doshi, P.; and Gmytrasiewicz, P. 2006. Exact solutions of interactive POMDPs using behavioral equivalence. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1025–1032.

Ritchie, D.; Thomas, A.; Hanrahan, P.; and Goodman, N. 2016. Neurally-guided procedural models: Amortized inference for procedural graphics programs using neural networks. *Advances in neural information processing systems*, 29.

Seaman, I. R.; van de Meent, J.-W.; and Wingate, D. 2018. Nested reasoning about autonomous agents using probabilistic programs. *arXiv preprint arXiv:1812.01569*.

Shum, M.; Kleiman-Weiner, M.; Littman, M. L.; and Tenenbaum, J. B. 2019. Theory of minds: Understanding behavior in groups through inverse planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 6163–6170.

Tejwani, R.; Kuo, Y.-L.; Shu, T.; Katz, B.; and Barbu, A. 2022. Social interactions as recursive mdps. In *Conference on Robot Learning*, 949–958. PMLR.

Ullman, T.; Baker, C.; Macindoe, O.; Evans, O.; Goodman, N.; and Tenenbaum, J. 2009. Help or hinder: Bayesian models of social goal inference. *Advances in neural information processing systems*, 22.