Theory of Mind Models for Human Robot Interaction under Partial Observability

Franziska Herbert*[1], Fabian Kalter*[1] and Tobias Niehues*[1]

Abstract— **Humans cooperate with each other on an everyday basis using some form of Theory of Mind (ToM). This term describes a human's ability to infer the mental state of another person, e.g. the person's beliefs or goals, which in turn enables more efficient interaction and cooperation. Robots do not inherently have a similar concept, thus implementing a ToM for robotic agents could potentially enable much more natural Human Robot Interaction. In this paper, we explore existing ToM approaches to infer the mental states of a human interaction partner. In addition, two different approaches are implemented to estimate the goal of another agent given the past environment states. The estimated goal is then used as input for a Reinforcement Learning module that learns our agent's optimal behavior. The proposed methods are evaluated in the Overcooked environment. We are able to show that we can successfully infer the goal of the other agent and use this goal to learn a policy. Yet at the same time, our results show that ToM models are not always as useful or necessary as they seem. The environmental conditions and evaluation metrics can not be chosen as straightforward as one might think and require more thought and research.**

I. Introduction

In recent years we have seen great progress in AI research. Starting from AIs like *Deep Blue* [1], multiple AI systems like *Deepmind's AlphaGo* [2], [3] or *OpenAI Five* [4] were able to beat human champions in competitive games like Chess, Go, or even video games. Yet, we see these remarkable performances only in competitive settings. This leaves us with the question, why we cannot reproduce such outstanding performances in multi-agent settings where the agents must cooperate on collaborative tasks.

On the contrary, humans work together on solving certain tasks on an everyday basis. While doing this, they benefit from remarkable social skills, allowing them to make inferences about the current thoughts, beliefs, and desires of other people around them, represented in the so-called *mental state*. Thus, to achieve high-quality *Human Robot Interaction (HRI)* [5], we need to employ mechanisms in robots that somehow reproduce the social capabilities of humans.

Cognitive Scientists and Psychologists refer to these social capabilities as the so-called *Theory of Mind (ToM)*, which is the conscious knowledge that other people also have their own mental states and consciousness [6]. Humans acquire this ability at the age of two to five years, allowing for more complex cooperation with other children [7].

Since robots do not have a native Theory of Mind, it is clearly beneficial to provide them with appropriate heuristics or models to compensate for these missing social skills, thus enabling proper cooperation with other agents in even complex tasks.

*Equal Contribution [1]Department of Intelligent Autonomous System, Technical University of Darmstadt, Germany, Correspondence to {franziska.herbert, fabian.kalter, tobias.niehues}@stud.tu-darmstadt.de

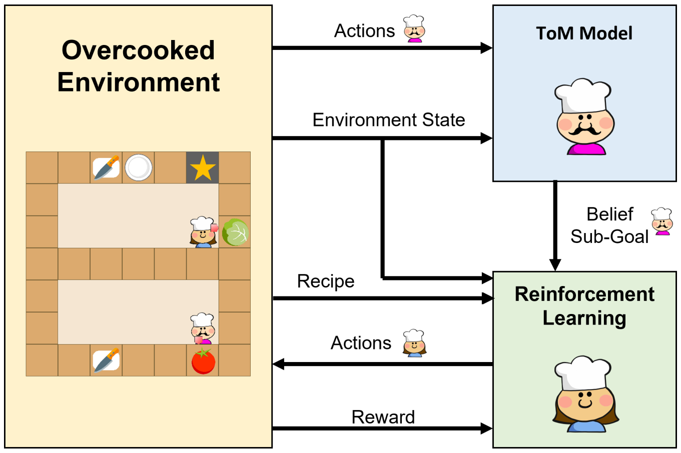


Fig. 1: An overview of our approach. Two agents interact with each other in an Overcooked environment where they must cooperatively prepare a salad. We use a ToM model to estimate one agent's current sub-goal by observing its past actions and past environment states. The resulting belief about the sub-goal, along with the environment state and recipe, is then used as input to a Reinforcement Learning module that learns the other agent's action.

Recent research introduced concepts on how to incorporate a Theory of Mind in agents, e.g. by using general *Artificial Neural Networks*, *Inverse Reinforcement Learning*, or models based on *Bayesian Analysis* (cf. Section II). The output of these ToM models can then be used in Reinforcement Learning, an approach for shaping agents' behaviors by letting them take arbitrary actions and then providing them with a reward, representing whether the taken action helped to reach the current objective or even hindered it [8].

In this work, we implement and compare various approaches to ToM models receiving the recent actions in other agents' trajectories as well as the current state of the world to create an estimate of the other agents' mental states. These mental states provide crucial information on the other agents' current intentions and are therefore essential for good cooperation. By using the predicted mental states as further input for our Reinforcement Learning setting, we can obtain policies that are not only optimal in terms of fulfilling the desired objective but also enable optimal cooperation with the other agents.

The environment used in this study is a multi-agent setting inspired by the video game *Overcooked* [9]. The agent's perception in this environment is constrained, inasmuch as the current environment state is only partially observable. This means that the agents cannot have complete knowledge on all and every entity in the environment, but e.g. only their immediate surroundings, and need to predict the unknown parts of the environment state.

The remaining paper is structured as follows. In Section II, we compare existing ToM models. In Section III, we introduce our approaches to a ToM model and in Section IV we evaluate those approaches in the Overcooked environment. Section V concludes the paper and introduces future work.

II. Related Work

A variety of different methods and approaches exists that implement some kind of Theory of Mind model on an agent for inferring the mental state of another agent. However, not all of these approaches define the mental state in the same way. While some approaches infer the agent's beliefs [10], [11], desires, [12] or goals [13], [14], others simply predict the agent's next action [15], [16]. In this work, we decided to infer the goal of the other agent, as this explains the agent's behavior in a more general and abstract way.
We ordered existing approaches based on their concept of implementing ToM models computationally into the five categories Cognitive Models, Perspective-Taking, Inverse Reinforcement Learning, Bayesian Approaches, and Deep Network approaches. In the following, we give a short overview of these categories.

Cognitive approaches were mainly explored in earlier years and focused on how human ToM models from Cognitive Science can be directly applied to a computational agent. [17] presents first ideas how two cognitive architectures, namely the *Theory of Mind Mechanism* model from [18] and the *Mindreading System* model from [19], could be used on a humanoid robot. However, no concrete implementation of these concepts is given. The approach by [20] uses the ACT-R cognitive architecture to explain human variability in HRI, and is tested both in simulation and on a real humanoid robot. Another widely used cognitive model is the *Simulation Theory* [21] which states that humans perform ToM by simulating another person's mental state and using their own decision-making system to predict the other person's behavior. [13] and [10] use this kind of approach to predict the goals and beliefs of humans in HRI scenarios. Here, the general idea lies in mapping the human's movements to the robot's own movements and thereby simulating the human's mental state. Another approach by [22] proposes to use an agent's own policy to infer other agents' hidden states in multi-agent Reinforcement Learning scenarios. In addition, recent work by [23] introduces a *CogToM* framework that uses a cognitive model that is based on instance-based learning theory to infer the ToM of another agent in a grid world task. Although all the above-mentioned approaches show promising results in their specific application scenario, many are heavily domain-specific and cannot be applied to arbitrary tasks easily. Additionally, while taking inspiration from human ToM seems reasonable, human ToM is also still an open research area and the aforementioned cognitive models do not perfectly describe the ToM of a human.

The next class of approaches uses Perspective-Taking to infer a ToM, which can be seen as a variant of the above-mentioned simulation-theoretic approaches. [24] implement perspective-taking on a robot by simulating the world from the perspective of the human and by using spatial reasoning to assist the human in collaborative tasks. [25] utilize a similar approach to learn from human demonstrations. A robot simulates the environment from the human's perspective and is, therefore, able to infer the human's goal and thus learn the demonstrated task. [26] extend this approach by enabling a robot to predict a goal even if the human fails to achieve it or provides insufficient data. [27] apply perspective-taking in a competitive game scenario by exploiting the fact that the human can only see parts of the environment, and [28] use perspective-taking to model team members' beliefs and achieve collaboration in HRI scenarios. Another approach by [29] introduces a situation assessment reasoner which uses both the human's state in the environment, and the task decomposition to understand the human's current situation. [30] transfer this idea to a shared plan environment by introducing a state-based system. All in all, perspective-taking approaches are able to predict the mental state of another agent in shared environments by taking its situation in the environment into account. However, these approaches rely on a fully observable environment that can be simulated with the necessary precision.

Another class of approaches uses Inverse Reinforcement Learning (IRL) to learn an underlying reward function. The agent is assumed to behave according to this reward function, striving to maximize the cumulated reward. [31] shows that certain relationships exist between ToM and IRL, e.g., by interpreting an agent's reward as its desire and the policy as its intentions. [16] apply this approach by learning the reward function of a human to model the human's next action in an autonomous driving scenario. [32] use IRL to predict the human's plan, also in an autonomous driving task, and [33] learn a cost function to infer the human's goal in a shared autonomy environment. However, many of those approaches assume that the agent is a rational planner and, therefore, acts optimally given its reward function. But it has been known for a long time that humans do not act rationally optimal [34] and might act depending on preferences or other beliefs. Therefore, if those approaches are used to predict the mental state of a human interaction partner, this optimality assumption will most likely not hold.

The class of Bayesian ToM models deals with the problem of inferring the other agent's mental state in a probabilistic way. These approaches maintain a probability distribution over the different mental states by using Bayes' Rule and by conditioning on the agent's history of states and actions. They utilize Inverse Planning to estimate the current value of a state with regard to reaching a specific mental state. For example, [11] introduce a Bayesian model for ToM that infers an agent's beliefs over different states of an environment given the history of states the agent visited in the last time steps. At each time step, the agent updates its beliefs about the environment. Similar approaches infer a human's sub-goal [35], an agent's belief [36], a human's beliefs, desires and precepts [12], relationships between agents [37] or beliefs about other agents' states [38]. One recent representative of the Bayesian approaches is called *Bayesian Delegation* [14]. This approach maintains a probability distribution over sub-goal allocations to the different agents and uses Bayesian Inference to calculate the probability that a specific sub-goal allocation occurs. The approach is tested in an Overcooked environment similar to the environment used in this work. A major advantage of Bayesian ToM models is that they naturally provide a probability distribution over the different mental states, which is required in many cases.

In recent years, several approaches have used deep networks to estimate the mental state of an agent [15], [39], [40]. The approach by [15] introduces the so-called *ToMnet* which can be used to predict the next action but also the goal of an agent in a grid world scenario. The underlying structure of the ToM model is a deep network that is divided into three different parts, the *character net*, the *mental state net*, and the *prediction net*. The character net is used to determine the character of the observed agent by taking its state-action history from past episodes as input, the mental state net infers the current mental state of the agent by looking at the state-action history of the current episode and the prediction net is used to predict the future behavior of the agent. While this approach performs very well on the respective grid world scenario and is able to predict the mental state of many different classes of agents, it also requires a large amount of training data for being able to perform at this level. In this paper, we implement an approach that is inspired by the above introduced *ToMnet*.

III. Our Approach

In the previous section, we compared several classes of how to computationally implement a Theory of Mind model for a robot or an agent. The main idea of this paper is to compare several of those different approaches and see how they perform in comparison with each other. In this section, we present our two approaches to inferring another agent's ToM. Namely, we are going to introduce a simple Bayesian model that forms a belief about an agent's goal and a deep neural network approach. Both approaches return a probability distribution over the agent's goals. Furthermore, we are going to introduce an RL model that learns how our agent should behave according to the inferred mental state of the other agent. An overview of our approach can be found in Figure 1.

A. *Theory of Mind Models*

We define the mental state to be equal to the sub-goal g_{sub} currently pursued by the agent. We are predicting a probability distribution over all n_{sub} possible sub-goals. A higher probability for a sub-goal g_{sub} corresponds to a higher belief in g_{sub} actually being the current sub-goal of the agent. By predicting the current sub-goal probabilistically and not the next action of the agent, we describe the mental state in a more general and abstract way.

The input for our model consists of a history of the state space and a history of actions of the last few time steps.
The state space is represented by a set of feature maps (FM), each consisting of simple bits structured in 2D grids with the same dimensions (H, W) as our environment. Each map represents one certain class of objects in the environment. Every entry in an FM represents one tile of the grid world in our environment and is either equal to 0 or 1. A 1 denotes that the object of the class represented by the FM is present at this tile, while a 0 states the absence of that object.
The complete state space is represented by a tensor of size (N, L, D, H, W), containing the feature maps for the last L time steps. N denotes the batch size, L is the sequence length and D is the number of object classes represented by a single FM. H and W directly correspond to the height H and width W of our environment.
The action history consists of a sequence with a fixed length L. If the sequence already contains L action-observation pairs, the oldest one is discarded in the next time step.
The output of our model is a probability distribution over all n_{sub} possible sub-goals.

1) Belief Tracker: In a first approach to inferring the sub-goal of the other agent, we take advantage of the fact that in most cases the sub-goal is determined by the objects with which the agent interacts. By tracking the agent's movement, we can infer the direction in which the agent is moving, and thus which objects the agent is likely to interact with. Such a goal-directed movement prediction has been successfully applied in other domains, such as in probabilistically predicting the movement goal of a human arm trajectory [41] or in modeling the movement of pedestrians [42]. However, in our case, we deal with a discrete environment, where the agent moves in a grid world.

We maintain a belief $b_{o,t}(\boldsymbol{o})$ to which object the agent is moving, by observing the last position $\boldsymbol{s}_t = (x_t, y_t)$ of the agent, i.e. the direction in which the agent moves in the grid world. At each time step, we update that belief according to Bayes' Rule

$$b_{o,t+1}(o_k) = p(o_k|\boldsymbol{s}_t, \boldsymbol{b}_{o,t}) = \frac{p(\boldsymbol{s}_t|o_k, \boldsymbol{b}_{o,t})b_{o,t}(o_k)}{\sum_j p(\boldsymbol{s}_t|o_j, \boldsymbol{b}_{o,t})b_{o,t}(o_j)},$$

where $b_{o,t}(\boldsymbol{o})$ is the current belief about the agent's object goal in form of a categorical probability distribution and $p(\boldsymbol{s}_t|o_k, \boldsymbol{b}_{o,t})$ is the likelihood that the specific action is performed given the real underlying object goal o and the belief from the last time step. We calculate this likelihood by assuming noisy goal directed movements

$$p(\boldsymbol{s}_t|o_k, \boldsymbol{b}_{o,t}) = \mathcal{N}(\boldsymbol{s}_t|\hat{\boldsymbol{s}}_k, \boldsymbol{I}\sigma_k),$$
$$\hat{\boldsymbol{s}}_k = \boldsymbol{s}_{t-1} + \frac{\boldsymbol{s}_{o,k} - \boldsymbol{s}_{t-1}}{|\boldsymbol{s}_{o,k} - \boldsymbol{s}_{t-1}|},$$

where σ_k is noise on the path towards an object goal, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{s}_{o,k}$ is the position of object o_k and $|.|$ is the Euclidean norm. In the first time step, the belief is initialized uniformly over all objects. When the agent interacts with an object, the belief is reset because we assume that the agent subsequently moves to another object and therefore changes its goal.

Since the execution of one sub-goal of the agent might include the interaction with more than one object, we need to maintain a second belief about the actual sub-goal of the agent which depends on the movement belief. The sub-goal belief $b_{g,t}(\boldsymbol{g})$ is updated in the same way as described above

$$b_{g,t+1}(g_k) = p(g_k|\boldsymbol{b}_{o,t}, \boldsymbol{b}_{g,k}) = \frac{p(\boldsymbol{b}_{o,t}|g_k, \boldsymbol{b}_{g,t})b_{g,t}(g_k)}{\sum_j p(\boldsymbol{b}_{o,t}|g_j, \boldsymbol{b}_{g,t})b_{g,t}(g_j)}.$$

Hereby, $p(\boldsymbol{b}_{o,t}|g_k, \boldsymbol{b}_{g,t})$ is again the likelihood that the motion belief occurs given the current sub-goal and the sub-goal belief from the last time step. It is computed using prior information about the decomposition of the sub-goals, i.e., which object the agent must interact with to satisfy the sub-goal, given the history of objects with which the agent has previously interacted. To compute the likelihood, we again use a Gaussian Distribution

$$p(\boldsymbol{b}_{o,t}|g_k, \boldsymbol{b}_{g,t}) = \mathcal{N}(\boldsymbol{b}_{o,t}|\hat{\boldsymbol{o}}, \boldsymbol{I}\sigma_k), \tag{1}$$

where $\hat{\boldsymbol{o}}$ denotes a one-hot vector over all objects with a 1 at the index of the next expected object. When one sub-goal is completed, the sub-goal belief is reset and in order to get a probability for the sub-goal DO NOTHING, the deviation from the agent's position from its last position is considered. If the agent could interact with multiple objects to complete the sub-goal, both probabilities are calculated according to Equation 1 and the two probabilities are added.

2) Deep Network: The deep neural network architecture used for evaluation in this paper is inspired by the architecture used in [15]. In general, the model can be divided into three sections. The first layers are convolutional layers to handle the state space, which is defined by 2D feature maps. These convolutional layers were chosen to exploit the spatial relations within the feature maps. Afterwards, an LSTM cell with two layers extracts temporal information from a history of observations and actions we also provide as input. Lastly, multiple fully-connected layers are used to infer the actual sub-goal.

As shown in Figure 2, this model uses three distinct inputs. The first consists of most of the environment states (e.g. object positions, agent orientations...), the second contains counter positions as well as cutting board positions and the third input are the actions taken by the other agent. The inputs have the following shapes

$$\begin{aligned}\text{Input}_1 &= (N, L, D_1, H, W)\\ \text{Input}_2 &= (N, L, D_2, H, W)\\ \text{Input}_3 &= (N, L, K),\end{aligned}$$

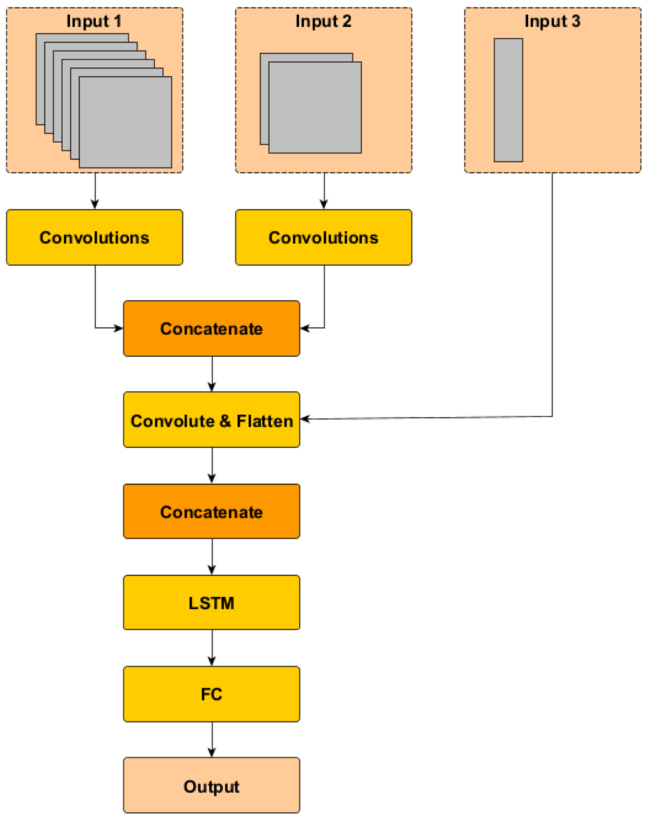


Fig. 2: The general network architecture of our deep ToM model. The network processes the state-action history of another agent and outputs a categorical probability distribution over all possible sub-goals.

where D_1 and D_2 are the number of feature planes in Input_1 and Input_2, respectively, and K is the number of actions an agent can take decoded as a one-hot vector. For an explanation of the other variables refer to Section III-A.

The idea behind this architecture is that the network learns a causal link between certain points of interest (POI) (e.g. a tomato), the neighborhood of said POI's, and the change of the POIs position over time, with the sub-goal an agent has in mind.

B. Belief MDP

We model the agents' interaction with the environment as a multi-agent Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [43]. The Dec-POMDP can be described as a tuple $< \mathcal{D}, S, A, T, R, O, \Omega >$, where $\mathcal{D}$ is the set of n agents, S is the set of states and A is the set of joint actions that consists of the actions a_i of the individual agents. The transition function $T : S \times A \rightarrow S$ describes the transition from one state into the next state when taking a specific joint action and each state transition results in a reward that is described by the reward function $R : S \times A \rightarrow \mathbb{R}$. All agents share the same reward function. Ω describes the set of joined observations and $O : S \times A \rightarrow \Omega$ is the observation function.

We assume that each agent can completely observe the positional state of the environment, i.e. the position of each object and agent in the environment. Therefore, the only part that is not observable is the mental state of the other agents. This makes the complete state of the environment partially observable. To solve the Dec-POMDP, we use the concept of a Belief MDP [44]. We maintain beliefs about the mental states of the other agents as probability distributions and include those into the state space of the POMDP. Thereby, the discrete POMDP becomes a continuous MDP which can be solved with standard Reinforcement Learning algorithms. The observations are the past state-action pairs from the other agents. Those are passed to the ToM model which returns the belief about the agents' mental states.

C. Overcooked Environment

To test the different ToM approaches, we make use of a multi-agent cooperative environment that is inspired by the video game *Overcooked* [9]. In the video game, each player controls a chef and the players need to cooperate with each other to prepare and serve different dishes in a limited time period. The game presents a coordination challenge, as not all ingredients are accessible to all agents and narrow corridors prevent agents from moving freely. In multi-agent RL research, different versions of the Overcooked game have recently been used as multi-agent environments [45], [46], [47], [48], [49], [14].

In this paper, we use a simplified environment that is inspired by and similar to the environment used by [14]. The environment can be seen as a simple grid world where counters are walls that the agent cannot pass and that are used to store and prepare food. An exemplary environment is shown in Figure 3a.
The task of the agents is to prepare salads following predefined recipes. Each recipe requires a certain sequence of high-level actions. For example, to prepare a tomato-lettuce salad, both the tomato and the lettuce must be chopped, be placed on a plate and then be delivered to the delivery square. Figure 3c shows an illustration of this recipe. To complete the recipe, agents can perform six different low-level actions, including moving to each side, interacting with an object, and a no-operation action. To influence the difficulty of the scenario, the map can be designed arbitrarily difficult with counters separating the two agents or different numbers of interaction objects.

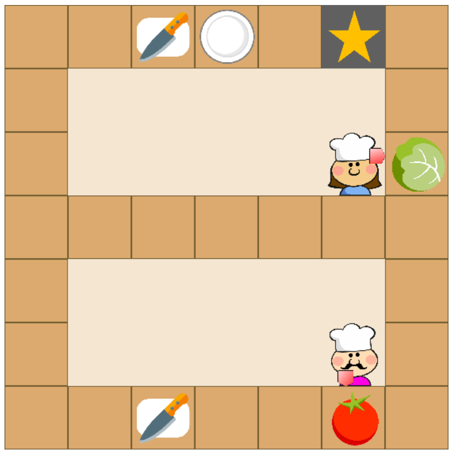
D. Motion Planning

To prepare a dish in the Overcooked environment, a sequence of several high-level actions needs to be executed by the agents. High-level actions could, for example, be CUT_TOMATO or DELIVER_PLATE.
To execute one high-level action, an agent has to perform several low-level actions (RIGHT, LEFT, UP, DOWN, DO NOTHING or INTERACT). For example, the execution of the high-level action CUT_TOMATO requires the agent to move to the tomato, interact with it to pick it up, move to the cutting board and interact with the cutting board twice, once to put the tomato down and the other time to chop the tomato. However, one high-level action always maps to a similar sequence of low-level actions that only varies depending on the location of the corresponding objects. For this reason, we decided to learn actions on a high level and provide the agent with a simple mapping between high-level and low-level actions. This not only simplifies the learning process but also provides us with a general action controller that we can use to control other agents using simple high-level behavior rules.

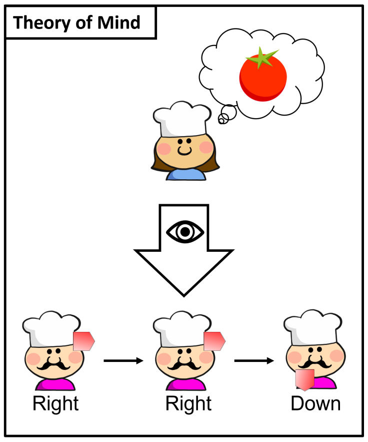
Each high-level action can be broken down into two parts: moving to objects and interacting with objects. The latter just requires executing the low-level action INTERACT. Moving to an object, on the other hand, requires a path to that object. Therefore, we calculate the shortest path using the Floyd-Warshall Algorithm [50], [51] to the nearest free field next to the counter on which the object is placed. The shortest path is recalculated at each time step and the agent always takes the first low-level action on that path.
In case the object is not reachable and the agent holds an object in his hand, the agent places that object on a free counter that is available to the other agent.

IV. Evaluation

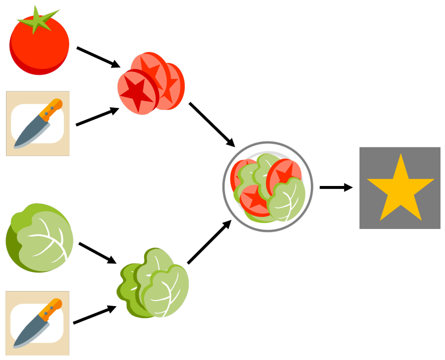
In this section, we evaluate the ToM models introduced in the previous section on two different levels of the Overcooked environment. The room design of the first level, the *reduced level*, can be seen in Figure 3a. Both agents are separated by a wall and can therefore only reach certain objects. This room design is supposed to encourage the agents to cooperate with each other in order to fulfill the task. In this level, the lettuce, the plate, and the delivery square are always only reachable by agent 0 (blue agent), while the tomato is only accessible to agent 1 (pink agent). Both agents have access to a cutting board. Within these requirements, the objects are placed at random positions, and also the starting position of both agents is selected randomly. In total, this level has 6400 different start states.
The second level, the *advanced level*, has the same room design as the first level, but the objects are no longer restricted to be only on one side of the room. The objects are randomly placed in different positions in the whole room. Therefore, both agents can in theory have access to all ingredients depending on their random placement. This level has 14400 start states in total.
In both levels, the agents share a reward function. For each completed sub-goal, i.e. step of the recipe, the agents get a reward of $+10$ and for finishing the complete recipe an additional reward of $+1$. This leads to a total possible reward of $+41$.
In order to train and evaluate our ToM models, we have recorded 1000 games between two agents respectively for each of the two levels. The two agents' behavior was preprogrammed, i.e. we predefined which high-level action the agents should perform in which situation and used our high-to-low-level action mapping to control the agents.



(a) Overcooked environment with wall.



(b) Theory of Mind.



(c) Recipe for a tomato-lettuce salad.

Fig. 3: The Overcooked environment that we use to evaluate the ToM models in this paper. In the environment (a) the two agents are separated by a wall. (b) shows an example of how the blue agent uses Theory of Mind to estimate the sub-goal of the pink agent by observing its actions and (c) shows the different steps to prepare a tomato-lettuce salad.

Figure 3b shows an example of how the estimation of the sub-goal of another agent with a ToM looks like.

A. *Training the ToM Models*

In this section, we describe how the Belief Tracker and the Deep ToM model perform in the Overcooked level designs previously introduced in Section IV. See Table I for the accuracies of the different ToM models on the training data set.

1) Belief Tracker: Since the belief tracker does not have any parameters that can be tuned, it does not require any training. We evaluated the performance of the belief tracker on the recorded game data sets for the two different levels, each time predicting the goal of both agents. Table I shows the achieved accuracies on the different levels for the two agents. As can be seen from the table, the performance of the Belief Tracker is quite constant across the different levels and agents. However, the accuracy does not exceed 80%. This is due to the fact that the two agents often have to wait for another. As soon as the Belief Tracker is very sure about one goal, i.e. the uncertainty of the distribution is very low, it is difficult to react to changes in the agents' behavior. Especially, when the agent waits for the other agent and therefore has the goal DO NOTHING, changes to another goal are almost never recognized.

2) Deep Network: To train the deep network the set of 1000 episodes (i.e. games) is used, each containing multiple deliveries of the final plate and the true sub-goal of each agent at every time step. Since we want this model to be used only by one of the agents, each model gets fed the input data of the other agent. Due to the fact that most of the hyperparameters of the model are determined by the environment that is used, only decisions regarding the sequence length L, the number of episodes and the learning rate need to be made.
Testing has shown that a fixed learning rate is not viable, hence the ADAM optimizer is used to obtain an adaptive learning rate directly via the optimization method. Choosing the right sequence length is not as straightforward, since it is intertwined with the maximal achievable model accuracy, the required number of training episodes, as well as the usability of the model itself. For example, by choosing a sequence length of 15 time steps when, on average, a game only takes 21 time steps to complete, the practical value of the model is greatly diminished. Additionally, the first training results have shown that the initial world size of 5x5 with a wall in the middle separating the agents yields bad results in regard to accuracy. This behavior occurs because the limited possible movement leads to very short and very similar movements for each sub-goal, thus making it hard to distinguish between them.
Because of that, a larger 7x7 environment was chosen. With the new environment, a sequence length of three time steps yields good results in regard to model accuracy (see Figure 4 and Table I), while also being short enough to provide utilizability. With these parameters, the loss function plateaus at around 700 episodes when using the advanced environment, while the loss deviation stops decreasing after approximately 1200 episodes. This training behavior also applies when using the reduced environment. Multiple models were trained on the data set using K-fold cross validation. In Figure 4 the average confusion matrix, for the advanced environment when training with the data of agent 0 and a sequence length of 3, is depicted exemplary. Furthermore, additional experiments have been conducted with a modified network, which does not take the current action of the other agent into account. This yields only slightly worse results in regard to accuracy (0.955) and error margin (0.022) compared to the original network. Such a model is particularly useful when only the environment state is accessible and communication between agents is not possible. In Figure 5 the resulting confusion matrix of the modified network is depicted.

	Reduced		**Advanced**	
	Agent 0	Agent 1	Agent 0	Agent 1
Belief Tracker	0.739	0.788	0.722	0.722
Deep Network	1	1	0.973 ± 0.014	0.985 ± 0.017

TABLE I: Accuracies of the predictions of the two different ToM models on the training data set.

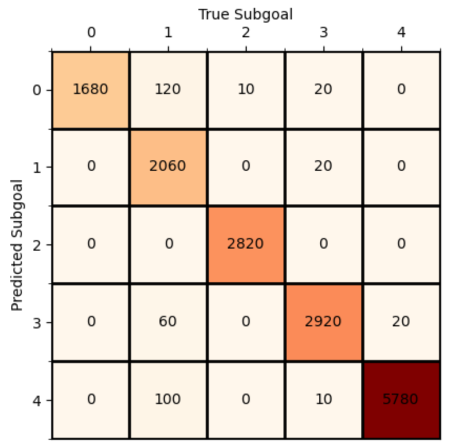


Fig. 4: Confusion matrix of the deep neural network ToM model for the behavior of agent 0, when trained in the advanced environment.

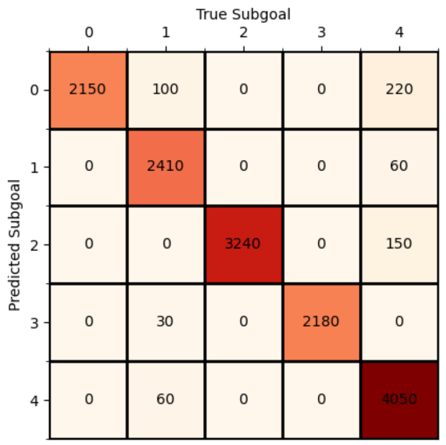


Fig. 5: Confusion matrix of the deep neural network ToM model without action input for the behavior of agent 0, when trained in the advanced environment.

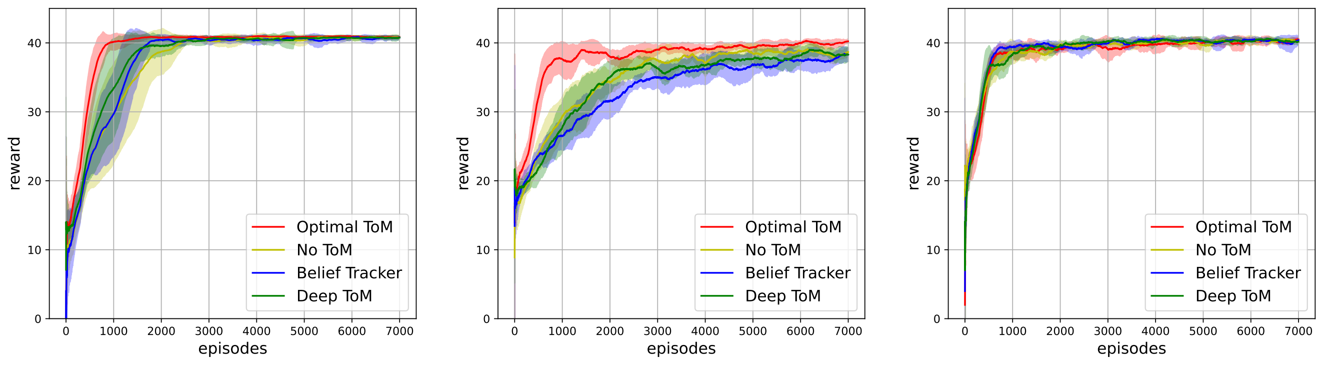
B. *Training a Policy for the Agent*

After training the ToM models, we want to take the estimated belief over the mental state of the other agent into account for finding a policy for our agent. As already introduced in Section III-B, the interaction of an agent with the environment can be modeled as a Belief MDP. To solve this MDP, we use Reinforcement Learning. In more detail, we utilize Double Deep Q-Learning (DQN) [52]. As an input for the deep Q-network we concatenate the several feature maps of the environment state, the belief distribution over the goal of the other agent from the ToM model and a vector which indicates which steps of the recipe have yet to be fulfilled and flatten the resulting vector into a one-dimensional vector. The corresponding DQN consists of two hidden fully-connected layers with 640 and 128 neurons, respectively, and a leaky ReLU activation function each. The output layer is also fully-connected and maps to the twelve different high-level actions. We train the network using an experience replay buffer, where tuples consisting of the past state, action, reward, and following state are stored, for faster training. At each training step, the parameters of the main Q-network are partially copied to a second network with the same architecture, the target network, which helps with the stability of the learning process. We train using a batch size of 64, a discount factor of 0.9 and the ADAM optimizer with a learning rate of 0.001.

In the following, we describe the results of training the DQN on the two different Overcooked levels introduced above. As a baseline, we always trained the DQN using no input from the ToM models, i.e. the corresponding input to the DQN is set to zero. This is supposed to evaluate how the learning algorithm performs without knowing or estimating the goal of the other agent. In addition, we trained using an optimal ToM model, which means that we input the true goal of the other agent to the DQN.

The results of training for agent 0 on the reduced level are shown in Figure 6a. The training was performed with the different versions of the ToM model for 10000 episodes each. The plot shows the mean and the standard deviation of the total achieved reward over 10 runs for each different ToM model plotted over the first 7000 training episodes. As can be seen in the plot, the learning curve from training with the optimal ToM model (red) converges to the optimal reward already after approximately 1000 episodes. None of our two ToM approaches (blue and green) converges significantly faster than the version without any ToM model (yellow). Although we are able to learn cooperative policies using both of our ToM model approaches, neither of our models provides a significant advantage over not using a ToM model in this environment level. However, we can observe that the learning curve originating from the optimal ToM model converges significantly faster than the version without the ToM model. Therefore, in this environment, with this state space, using a ToM seems to be slightly beneficial.

Similar results could be observed when training on the advanced environment as can be seen in the plot in Figure 6b. Here, the learning curve of the Belief Tracker even rises more slowly than the version without a ToM model. In this



(a) Reduced level.

(b) Advanced level.

(c) Advanced level with changed state space.

Fig. 6: Results from training the DQN with the different ToM models on (a) the reduced level, (b) the advanced level and (c) the advanced level with a changed state space. The mean and the standard deviation of the total achieved reward over 10 runs is plotted over the number of episodes.

case, however, all the different trained policies required more than 7000 episodes to fully converge to the optimal reward.

To investigate the comparatively good performance of the version not using a ToM model, we reevaluated the definition of the state space. The state space involves the current position of all objects and all agents in the environment, in addition to the recipe, which is a vector that gives information about the pending sub-goals, and the estimated belief about the other agent's state space. Therefore, the state space seems to be over-defined, and learning an optimal policy is already possible without estimating the other agent's mental state. This seems to be the reason why the version without the ToM model performs at a high level.

We explored if removing information from the state space could potentially highlight the benefit of using a ToM model and therefore estimating the mental state of the other agent. Only training with the belief from the ToM model did not lead to convergence at all. Therefore, we had to include information about the position of the objects in the environment in order to achieve convergence. By neglecting the position of the agents and the current recipe, however, we achieve a performance boost for all four policies. The results are shown in Figure 6c. As can be seen in the plot, all four trained policies converge comparable or even better than the optimal policy when training with the full state space in Figure 6b. There is no longer a significant difference between the convergence behavior of the four policies.

This shows, that the process of learning a policy in our two levels of the Overcooked environment cannot significantly profit from using our ToM models. This is most likely due to the fact that the environment is not suitable for evaluating our ToM approaches since it is too small and provides too little variation in the task execution. The sub-goal of the other agent can already be determined from the presence and position of the objects in the environment and does not require any ToM model. Also, the other agent is preprogrammed to use the shortest path to objects and to always perform the optimal sub-goal in a specific situation. Its behavior is therefore predictable and can be assumed by the other agent.

V. Conclusion

In this paper, we have explained and grouped different Theory of Mind approaches. In addition, we have implemented two different methods to infer the goal of another agent. The first approach is a belief tracker that decides, based on the agent's movements and objects he interacted with, what the most probable goal of the agent is. The second approach is a deep neural network that processes the bygone environment states to infer the most likely goal. The belief over the sub-goals is then fed to a deep Q-network to estimate the best action to take in a situation. Our results show that the simple belief tracker, while fundamentally working, yields significantly worse results in regard to accuracy than the deep ToM network. Moreover, we have shown that supplying the information of another agent's current goals does not significantly influence the resulting performance (in the case of the deep ToM model), or even hinders learning (in the case of the Belief Tracker). With this, it is safe to say, that designing environments that can be used to evaluate the information gain of a ToM model, is not as straightforward as one might think.

This raises the question for future work of how to define a level of cooperation that requires Theory of Mind models and how to evaluate them. Additionally, we plan on evaluating our approaches on an open Overcooked level without the separating wall between the agents and further increasing the capabilities of the motion planer. Another future task is to compare our approaches against other approaches from literature like the Bayesian Delegation [14] or some implementation of Inverse Reinforcement Learning. Also, making the environment less observable by adding for example a field of view for the agents opens up further challenges. Another interesting future line of work would be to not use preprogrammed agents for the counterpart but use agents with variable behavior or to train all agents at the same time.

References

[1] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[5] K. Dautenhahn, "Socially intelligent robots: dimensions of human–robot interaction," *Philosophical transactions of the royal society B: Biological sciences*, vol. 362, no. 1480, pp. 679–704, 2007.

[6] H. Wimmer and J. Perner, "Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception," *Cognition*, vol. 13, no. 1, pp. 103–128, 1983.

[7] H. M. Wellman, *The child's theory of mind.* The MIT Press, 1992.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[9] Ghost Town Games Ltd., "Overcooked," 2016. [Online]. Available: https://store.steampowered.com/app/448510/Overcooked/

[10] C. Breazeal, J. Gray, and M. Berlin, "An embodied cognition approach to mindreading skills for socially intelligent robots," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 656–680, 2009.

[11] C. Baker, R. Saxe, and J. Tenenbaum, "Bayesian theory of mind: Modeling joint belief-desire attribution," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[12] C. L. Baker, J. Jara-Ettinger, R. Saxe, and J. B. Tenenbaum, "Rational quantitative attribution of beliefs, desires and percepts in human mentalizing," *Nature Human Behaviour*, vol. 1, no. 4, pp. 1–10, 2017.

[13] J. Gray, C. Breazeal, M. Berlin, A. Brooks, and J. Lieberman, "Action parsing and goal inference using self as simulator," in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.* IEEE, 2005, pp. 202–209.

[14] R. E. Wang, S. A. Wu, J. A. Evans, J. B. Tenenbaum, D. C. Parkes, and M. Kleiman-Weiner, "Too many cooks: Coordinating multi-agent collaboration through inverse planning," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 2032–2034.

[15] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. A. Eslami, and M. Botvinick, "Machine theory of mind," in *International conference on machine learning*. PMLR, 2018, pp. 4218–4227.

[16] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.

[17] B. Scassellati, "Theory of mind for a humanoid robot," *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.

[18] A. M. Leslie, "Tomm, toby, and agency: Core architecture and domain specificity," *Mapping the mind: Domain specificity in cognition and culture*, vol. 29, pp. 119–48, 1994.

[19] S. Baron-Cohen, *Mindblindness: An essay on autism and theory of mind*. MIT press, 1997.

[20] L. M. Hiatt, A. M. Harrison, and J. G. Trafton, "Accommodating human variability in human-robot teams through theory of mind," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[21] V. Gallese and A. Goldman, "Mirror neurons and the simulation theory of mind-reading," *Trends in cognitive sciences*, vol. 2, no. 12, pp. 493–501, 1998.

[22] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 4257–4266.

[23] T. N. Nguyen and C. Gonzalez, "Theory of mind from observation in cognitive models and humans," *Topics in Cognitive Science*, 2021.

[24] J. G. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz, and A. C. Schultz, "Enabling effective human-robot interaction using perspective-taking in robots," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 35, no. 4, pp. 460–470, 2005.

[25] M. Berlin, J. Gray, A. L. Thomaz, and C. Breazeal, "Perspective taking: An organizing principle for learning in human-robot interaction," in *AAAI*, vol. 2, 2006, pp. 1444–1450.

[26] C. Breazeal, M. Berlin, A. Brooks, J. Gray, and A. L. Thomaz, "Using perspective taking to learn from ambiguous demonstrations," *Robotics and Autonomous Systems*, vol. 54, pp. 385–393, 2006.

[27] J. Gray and C. Breazeal, "Manipulating mental states through physical action," *International Journal of Social Robotics*, vol. 6, no. 3, pp. 315–327, 2014.

[28] K. Talamadupula, G. Briggs, T. Chakraborti, M. Scheutz, and S. Kambhampati, "Coordination in human-robot teams using mental modeling and plan recognition," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2957–2962.

[29] G. Milliez, M. Warnier, A. Clodic, and R. Alami, "A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management," in *The 23rd IEEE international symposium on robot and human interactive communication*. IEEE, 2014, pp. 1103–1109.

[30] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 319–326.

[31] J. Jara-Ettinger, "Theory of mind as inverse reinforcement learning," *Current Opinion in Behavioral Sciences*, vol. 29, pp. 105–110, 2019.

[32] R. Choudhury, G. Swamy, D. Hadfield-Menell, and A. D. Dragan, "On the utility of model learning in hri," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 317–325.

[33] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *Robotics science and systems: online proceedings*, vol. 2015, 2015.

[34] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *science*, vol. 185, no. 4157, pp. 1124–1131, 1974.

[35] R. Nakahashi, C. Baker, and J. Tenenbaum, "Modeling human understanding of complex intentional action with a bayesian nonparametric subgoal model," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[36] M. Ramirez and H. Geffner, "Goal recognition over pomdps: Inferring the intention of a pomdp agent," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[37] M. Shum, M. Kleiman-Weiner, M. L. Littman, and J. B. Tenenbaum, "Theory of minds: Understanding behavior in groups through inverse planning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 6163–6170.

[38] L. Yuan, Z. Fu, L. Zhou, K. Yang, and S.-C. Zhu, "Emergence of theory of mind collaboration in multiagent systems," *arXiv preprint arXiv:2110.00121*, 2021.

[39] I. Oguntola, D. Hughes, and K. Sycara, "Deep interpretable models of theory of mind," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 657–664.

[40] H. Zhu, G. Neubig, and Y. Bisk, "Few-shot language coordination by modeling theory of mind," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 901–12 911.

[41] D. Koert, J. Pajarinen, A. Schotschneider, S. Trick, C. Rothkopf, and J. Peters, "Learning intention aware online adaptation of movement primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3719–3726, 2019.

[42] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*. IEEE, 2015, pp. 454–460.

[43] F. A. Oliehoek, "Decentralized pomdps," in *Reinforcement Learning*. Springer, 2012, pp. 471–503.

[44] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 362–370.

[45] M. Carroll, R. Shah, M. K. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan, "On the utility of learning about humans for human-ai coordination," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5174–5185, 2019.

[46] D. Strouse, K. McKee, M. Botvinick, E. Hughes, and R. Everett, "Collaborating with humans without human data," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[47] P. Knott, M. Carroll, S. Devlin, K. Ciosek, K. Hofmann, A. D. Dragan, and R. Shah, "Evaluating the robustness of collaborative agents," *arXiv preprint arXiv:2101.05507*, 2021.

[48] K. R. McKee, J. Z. Leibo, C. Beattie, and R. Everett, "Quantifying environment and population diversity in multi-agent reinforcement learning," *arXiv preprint arXiv:2102.08370*, 2021.

[49] R. Charakorn, P. Manoonpong, and N. Dilokthanakul, "Investigating partner diversification methods in cooperative multi-agent deep reinforcement learning," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 395–402.

[50] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[51] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 11–12, 1962.

[52] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.