A Extended Related Work

In this section, we connect our work with other lines of research, discussing the similarities and differences between our study and previous works in the field. The discussion herein presented corresponds to an extended version of Sec. 5.

A.1 Partial Observability in MARL

Closely related to our work are studies that address the problem of partial observability in MARL. As an example, Omidshafiei et al. (2017) propose a decentralized MARL algorithm that uses RNNs to improve the agents' observability. Mao et al. (2020) use an RNN to first compress the agents' histories into embeddings that are posteriorly fed into deep Q-networks, helping to improve agents' observability. The commonly used paradigm of centralized training with decentralized execution also contributes to alleviating partial observability at train time (Oliehoek et al., 2011; Rashid et al., 2018; Foerster et al., 2016; 2017; Sunehag et al., 2017; Son et al., 2019; Mahajan et al., 2019; Wang et al., 2020b; Yu et al., 2021). Under such paradigm, the calculation of value functions or policy gradients can exploit the centralization of information, thus alleviating partial observability.

Another way to alleviate the problem of partial observability in MARL, especially at test time, is to consider communication between the agents. We review such setting in the following section.

A.1.1 Communication in MARL

Different lines of research focus their attention on the development of communication techniques for MARL (Zhu et al., 2022), focusing on how the sharing of information between the agents can be used to improve the RL agents' learning. Early works addressed communication under partially observable cooperative MARL tasks: Sukhbaatar et al. (2016) share the outputs of the hidden layers of a shared neural network among the agents; Foerster et al. (2016) explicitly learn the content of the messages transmitted between agents by following an end-to-end approach in which gradients are back-propagated through the communication variables. Recent works in the field study how (Niu et al., 2021; Kim et al., 2019a), when (Singh et al., 2018; Hu et al., 2020), and what (Foerster et al., 2016) should be communicated among the agents in order to foster cooperation.

Similarly to our work, previous studies focused on the sharing of (encoded) local observations and actions among agents (Foerster et al., 2016) in a proxy-like manner (Wang et al., 2019). However, as opposed to previous works, we assume that agents have no control over when and with whom to communicate and, instead, should robustly perform under any type of communication policy. We also emphasize that we are not focused on learning the content of the messages being communicated (as in (Foerster et al., 2016)), focusing our attention on a rather "passive" communication setting by considering the sharing of local observations and actions among the agents. Other works focus on learning how to combine received information with local information before feeding it into the RL model (Jiang & Lu, 2018; Wang et al., 2019). In our work, we concatenate received information alongside local information. However, the methods developed by previous studies, which can be seen as orthogonal contributions in comparison to our work, can be readily incorporated into our method.

Finally, some works consider learning robust communication protocols under failing/missing information. Previous studies learn mechanisms that improve communication efficiency by either limiting the variance of exchanged messages (Zhang et al., 2019), or temporally smoothing information shared between agents (Zhang et al., 2020). Due to the decreased variability of the messages exchanged throughout timesteps, such methods achieve improved robustness against transmission loss. However, since in this work we are not focusing our attention on methods that learn the contents of the messages being exchanged, we did not use the aforementioned methods as baselines in our work as the comparison between methods would be deemed inappropriate. Kim et al. (2019b) propose a learning technique for MARL called message-dropout, which aims at: (i) effectively handling the increased input dimension in MARL with communication; and (ii) making learning robust against communication errors in the execution phase. Message-dropout drops the messages received from other agents independently at random during training before inputting them into the RL algorithm. In a similar fashion to message dropout, Wang et al. (2020a) propose a recurrent actor-critic algorithm for handling multi-agent coordination under partial observability with limited communication, showing that recurrency successfully contributes to robust performance when communication fails.

Following both Kim et al. (2019b) and Wang et al. (2020a), we use message-dropout with recurrent learners as a baseline in our work.

We refer to Zhu et al. (2022) for an extensive discussion of the different works that propose communication protocols for MARL.

A.2 Modelling Other Agents

In contexts where a single agent learns in an environment where other agents are also present, some works have explored ways to model information about the other agents, such as their actions and observations, based on local information available to the learning agent. The work of Papoudakis et al. (2021a) uses a recurrent neural network to predict the other agents' actions and observations in order to make better action selections in a centralized training with decentralized execution setting. At execution time, the agent then uses its learned model to make explicit predictions about other agents' observations and actions. Xie et al. (2020) does similarly in a latent space. The work of He et al. (2016) uses, instead, the other agents' observations to predict their actions, which assumes centralization will be available at execution time.

Contrarily to the mentioned settings, in ours, we aim at learning policies for multiple agents. To that end, we use a model of the agents' observations and actions to make predictions about other agents and improve their performance in cooperative tasks.

A.3 Model-based MARL

Recently, different works addressed model-based MARL, being mostly focused on improving the sample efficiency of MARL methods by leveraging the knowledge of the learned environment dynamics in policy optimization (Willemsen et al., 2021; Bargiacchi et al., 2021). Closely related to our work are studies that propose model-based approaches to MARL while considering communication among the agents. As an example, Kim et al. (2021) propose a communication protocol that encodes into the message an agent's imagined trajectory computed by performing rollouts using an opponent model and a dynamics function.

As opposed to previous works, in our work we focus our attention on the study of methods that allow for robust execution under different communication degrees. While our method can be categorized as a model-based MARL method that implicitly models both the dynamics function as well as the other agents' policies, we use it with a rather different objective than the aforementioned articles.

We refer to Wang et al. (2022) for an extensive discussion of model-based approaches to MARL.

A.4 Multi-agent Trajectory Prediction

There exists a number of works that address trajectory prediction under multi-agent settings using sequence models (Alahi et al., 2016; Yeh et al., 2019; Hauri et al., 2020; Omidshafiei et al., 2021). As an example, Alahi et al. (2016) use an RNN to learn and predict the trajectory of pedestrians. Hauri et al. (2020) propose an uncertainty-aware multi-modal deep learning model to predict multiple future trajectories of basketball players. Omidshafiei et al. (2021) propose a method based on graph neural networks and bi-directional RNNs to predict unobserved parts of football players' trajectories.

Our work resembles some similarities with the aforementioned studies since our predictive method MARO solves a similar problem to that of the aforementioned works if we consider that the observations correspond to agents' coordinates. However, in our study, we consider a rather broader setting in which agents' observations can correspond to any type of information collected by the agents. Importantly, we focus our attention on control settings whereas the previous works only deal with predictive settings. Also, we use the predictive model with the objective of being robust to missing information during agency. Nevertheless, we note that our method MARO can possibly benefit from techniques proposed by the aforementioned works.

B EXPERIMENTAL EVALUATION

In this section, we present supplementary materials for Sec. 4. We describe the proposed MARL scenarios in detail in Sec. B.1. In Sec. B.2, we describe our experimental methodology. Finally, we present our complete set of experimental results in Sec. B.3.

B.1 DESCRIPTION OF THE PROPOSED MARL SCENARIOS

SpreadBlindfold (SB) The environment consists of three agents and three designated landmarks in a 2D map. At the start of each episode both the position of the agents and of the landmarks is randomly generated. The goal of the agents is to cover all the landmarks while avoiding collisions: agents are (globally) rewarded considering how far the closest agent is to each landmark (sum of the minimum distances) and are (locally) penalized if they collide with other agents. Differently from the original Simple Spread environment, the agent's observation only includes the position and velocity of the agent itself and the relative position of all landmarks. Through communication with the central proxy, the agents can access the position and velocities of the other agents.

HearSee (HS) The environment consists of two heterogeneous agents and a single landmark in a 2D map. At the start of each episode both the position of the agents and of the landmark is randomly generated. The goal of the agents is to cooperate in order for both of them to cover the landmark: agents are (globally) rewarded considering how far the closest agent is to each landmark (sum of the minimum distances). In this scenario, one of the agents ("Hear" agent) is provided with the absolute position of the landmark in its observation. However, it does not have access to its own position. The other agent ("See" agent) is able to access the position and velocities of both agents in its observation, yet does not have access to the position of the landmark. Only through communicating with the central proxy, can the agents have access to both their positions and the position of the landmark in order to complete the task.

SpreadXY (SXY) The environment consists of two heterogeneous agents and two designated landmarks in a 2D map. At the start of each episode both the position of the agents and of the landmarks is randomly generated. The goal of the agents is to cover all the landmarks while avoiding collisions: agents are (globally) rewarded considering how far the closest agent is to each landmark (sum of the minimum distances) and are (locally) penalized if they collide with other agents. Differently from SSB, one of the agents has access to the X position and velocity of both agents, while the other agent has access to the Y position and velocity of both agents. Both agents observe as well the absolute position of all landmarks. Through communication with the central proxy, the agents can access the complete position and velocities of the other agents and cover the landmarks.

We refer to Lowe et al. (2017) for a visual depiction of the environments.

B.2 EXPERIMENTAL METHODOLOGY, IMPLEMENTATION AND HYPERPARAMETERS

We consider two MARL algorithms: IQL and QMIX. We employ the same LSTM-based controller networks across all evaluations. We follow the hyperparameters suggested by Papoudakis et al. (2021b); we train all models for 4M steps, performing 5 training runs for each experimental setting and 50 evaluation rollouts for each training run. We assume that $p = 1$ at $t = 0$ for the MD and MARO algorithms. The performance of the Obs. and J. Obs. baselines are evaluated by aggregating evaluation rollouts with $p = 0$ and $p = 1$, respectively. The other algorithms are evaluated for p sampled from a discretized uniform distribution. We display our training hyperparameters for the RL controllers and the predictive model in Table 3 and Table 4, respectively.

We developed our code in a Python environment using the EPyMARL framework (Papoudakis et al., 2021b) and PyTorch (Paszke et al., 2019). The computational code is available in Github.

B.3 EXPERIMENTAL RESULTS

In this section, we display the complete experimental results. Our main results are presented in Sec. B.3.1; the results of our ablation study are presented in Sec. B.3.2. In Sec. B.3.4, we display the results for the *Switch* baseline. In Sec. B.3.5 we display a set of figures that illustrates the

Table 3: Hyperparameters for the RL controllers across all environments.

(a) IQL

hidden dimension	128
learning rate	0.0005
reward standardisation	True
network type	GRU
evaluation epsilon	0.0
epsilon anneal	$500,000$
target update	200

(b) QMIX

hidden dimension	128
learning rate	0.0005
reward standardisation	True
network type	GRU
evaluation epsilon	0.0
epsilon anneal	$50,000$
target update	200

Table 4: Hyperparameters for the predictive model across all environments and algorithms.

hidden dimension	128
learning rate	0.001
grad clip	1.0

predictions made by the predictive model. In all plots, alongside scalar mean values, we report the 95% bootstrapped confidence interval.

B.3.1 Main Experimental Results

In this section, we present the complete experimental results of all approaches across all environments and algorithms: in Figs. 5 and 6 we show the performance in all environments of all approaches during training, considering different communication levels p, using the IQL and QMIX algorithms, respectively; in Figs. 7 and 8 we show the performance in all environments of all approaches during training, considering $p \sim \mathcal{U}(0, 1)$, using the IQL and QMIX algorithms, respectively; in Figs. 9 and 10 we show the performance in all environments of all approaches during execution, considering different communication levels p, using the IQL and QMIX algorithms, respectively.

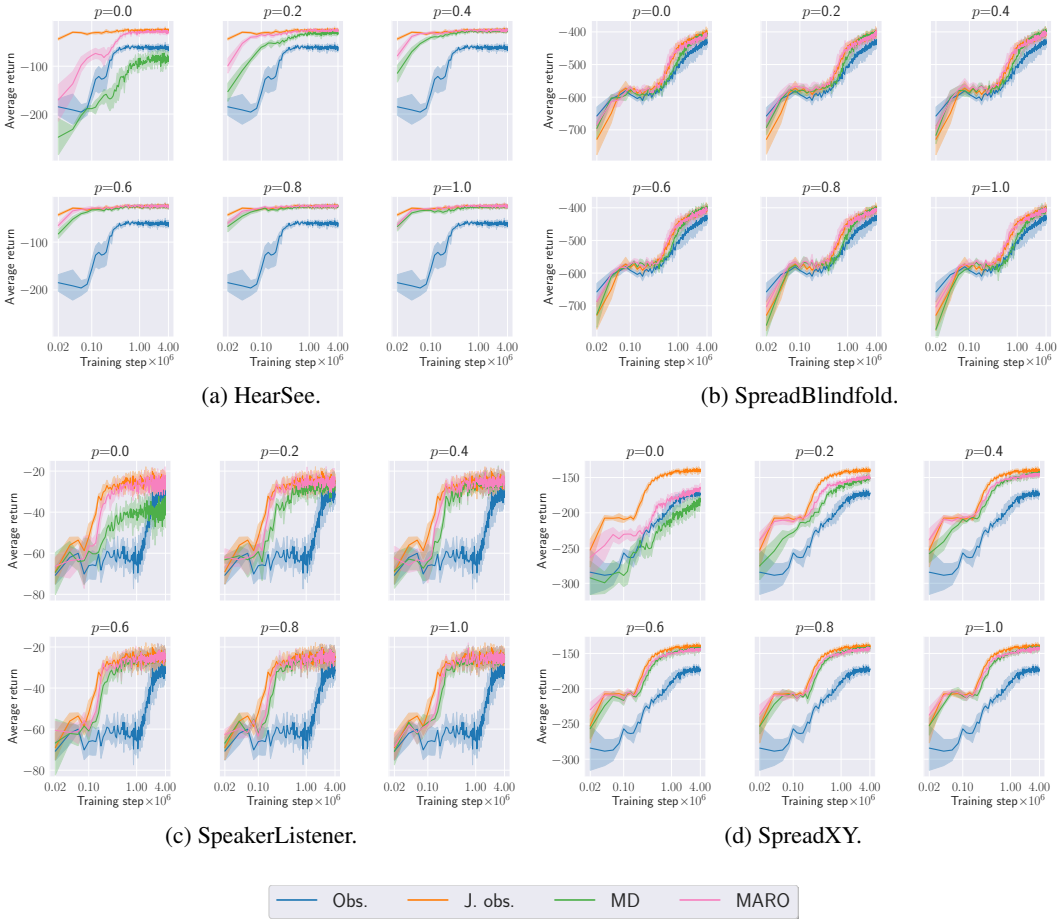
B.3.2 Ablation Study

In this section, we present the complete experimental results of the ablation study of MARO across all environments and algorithms: in Figs. 11 and 12 we show the performance in all environments of MARO and the ablated versions during training, considering different communication levels p, using the IQL and QMIX algorithms, respectively; in Figs. 13 and 14 we show the performance in all environments of MARO and the ablated versions during training, considering $p \sim \mathcal{U}(0, 1)$, using the IQL and QMIX algorithms, respectively; in Figs. 15 and 15 we show the performance in all environments of MARO and the ablated versions during execution, considering different communication levels p, using the IQL and QMIX algorithms, respectively; in Figs. 17 and 18 we show the performance in all environments of different sampling methods for the training scheme of MARO, using the IQL and QMIX algorithms, respectively.

B.3.3 Sampling Schemes of the Communication Matrix

We evaluate three different sampling schemes of the communication matrix: (i) our default setting, p_{default}, under which $p_{i,j} = p_{j,i} = p$ with $p \sim U(0, 1)$, sampled at the beginning of each episode; (ii) a setting $p_{\text{asymmetric}}$ featuring communication matrices C such that $p_{i,j} \neq p_{j,i}$ and all entries $p_{i,j}$ of the matrix are independently sampled from $U(0, 1)$ at the beginning of each episode; and (iii) p_{dynamic}, similar to $p_{\text{asymmetric}}$ but with communication matrices that are sampled every 5 time steps. We present the results in Table 5.

The results highlight the robustness of our approach to different sampling methods of the communication matrices between the agents, with no significant performance difference between the three conditions.



(a) HearSee. (b) SpreadBlindfold.

(c) SpeakerListener. (d) SpreadXY.

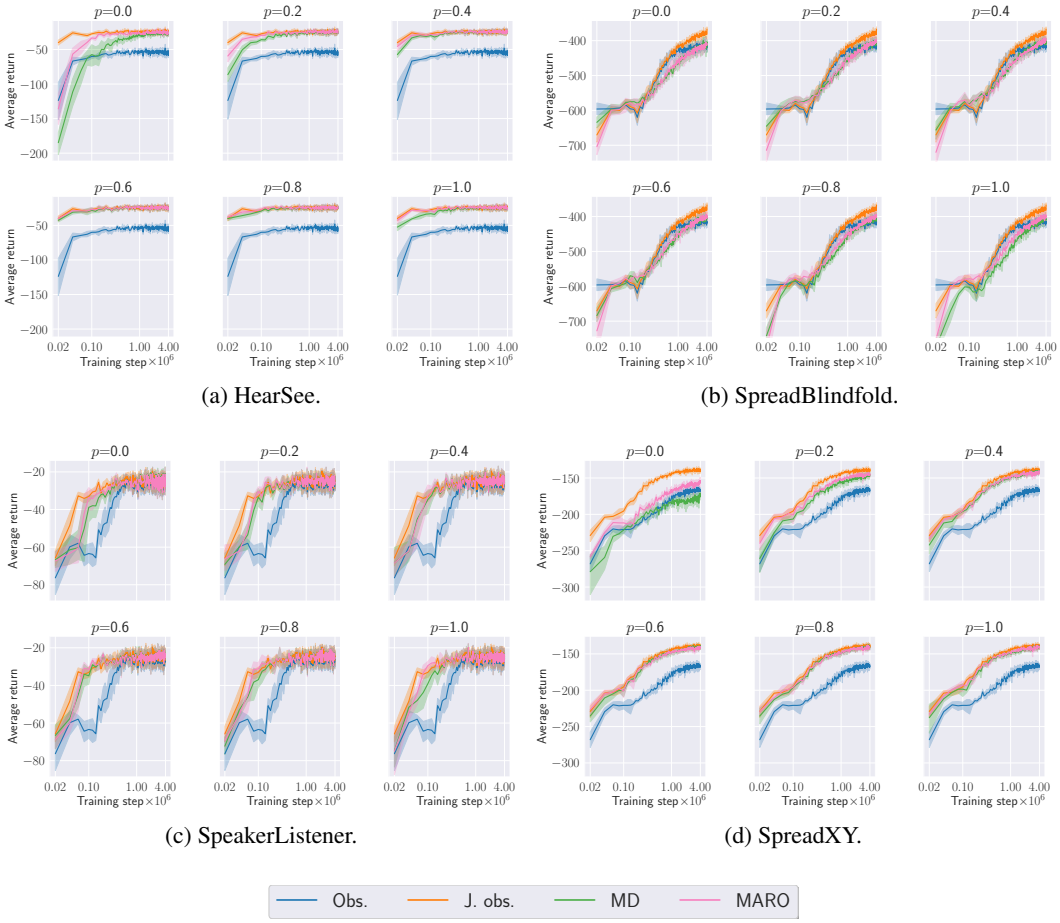
Figure 5: Average episodic returns during training with 95% bootstrapped confidence interval for different communication levels p, for all approaches and environments, using the IQL algorithm.

B.3.4 SWITCH BASELINE

In Fig. 19 we present the experimental results that compare MARO against the *Switch* baseline, introduced in Section 4.4.2, in the HearSee environment. The Switch baseline selects actions using two controllers: one that receives the joint observation, used when communication is allowed, and another that receives only the local observation, used otherwise.

Table 5: Average episodic returns and 95% bootstrapped confidence interval over five seeds for different sampling schemes of the communication matrix in MARO. Higher is better.

Environment	**IQL**			**QMIX**		
	p_{default}	$p_{\text{asymmetric}}$	p_{dynamic}	p_{default}	$p_{\text{asymmetric}}$	p_{dynamic}
SpreadXY	-146.3 (-2.1,+2.3)	-146.8 (-1.5,+1.6)	-146.2 (-1.7,+2.1)	-144.6 (-2.1,+2.1)	-143.0 (-1.9,+1.6)	-142.0 (-4.0,+3.0)
SpreadBlindFold	-405.0 (-2.4,+1.6)	-398.6 (-13.6, +15.2)	-403.6 (-9.6,+8.1)	-400.4 (-5.3,+6.3)	-411.7 (-13.0,+15.3)	-396.3 (-9.5,+9.6)
HearSee	-22.5 (-2.2,+2.5)	-23.8 (-2.4,+2.4)	-23.3 (-2.2,+1.9)	-25.1 (-3.1,+2.9)	-22.7 (-1.0,+1.0)	-23.4 (-5.0,+3.9)
SpeakerListener	-24.1 (-2.1,+2.1)	-28.8 (-5.6,+4.7)	-23.2 (-2.4,+2.8)	-23.9 (-3.4,+3.4)	-27.4 (-5.5,+4.1)	-24.1 (-3.3,+3.8)



(a) HearSee. (b) SpreadBlindfold.

(c) SpeakerListener. (d) SpreadXY.

Figure 6: Average episodic returns during training with 95% bootstrapped confidence interval for different communication levels p, for all approaches and environments, using the QMIX algorithm.

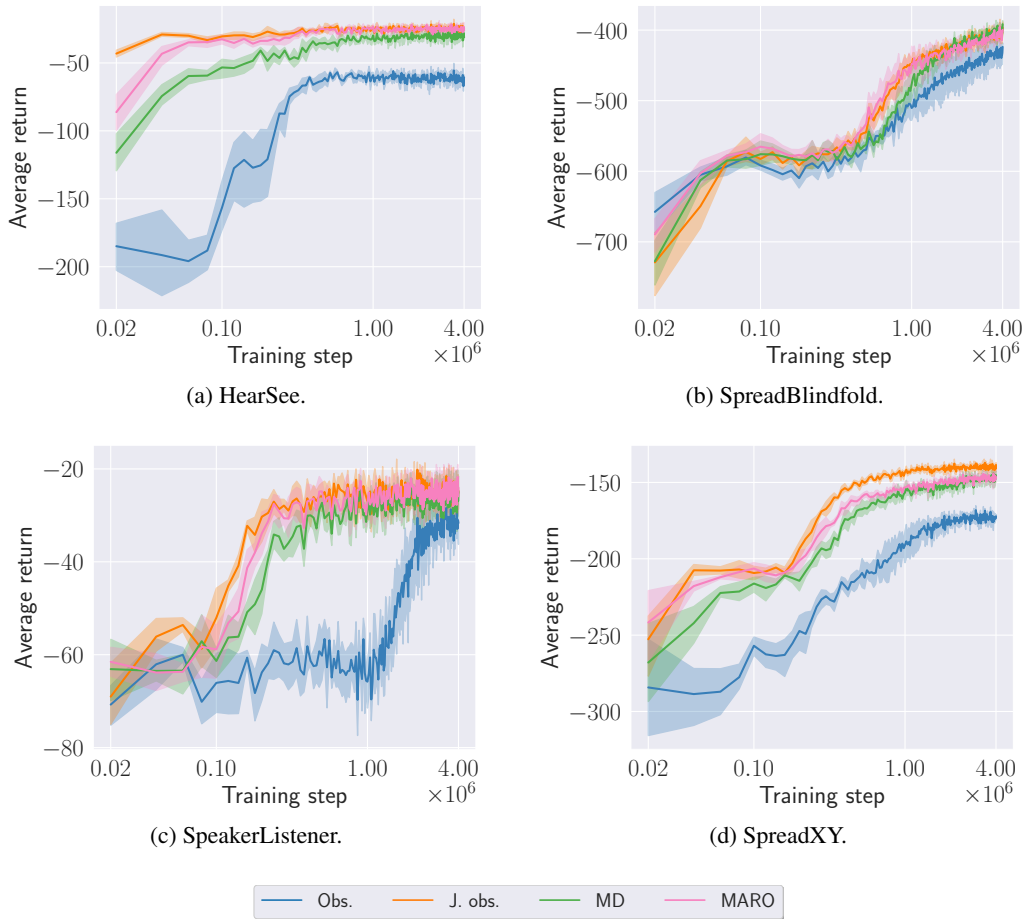
B.3.5 Multi-agent Trajectory Prediction

We display, in Figs. 20, 21 and 22, an illustration of the trajectory predictions made by the predictive model from the perspective of each of the agents. The plots are computed, at each timestep and from the perspective of each agent, by computing the estimated trajectories of all agents for the next 4 timesteps. The 4-step ahead predictions are entirely computed using estimated quantities, i.e., real observations are not incorporated into the predictions and the predictive model works in a fully auto-regressive manner.

B.4 Scaling the number of agents

Here we test the performance of MARO against the MD baseline as well as fully decentralized (Obs.) and fully centralized (J.Obs) agents. To our end, we extend the SpreadXY scenario to 4 agents, two of which observe the X-axis coordinate positions of all agents and two of which observe the Y-axis coordinate positions of all agents. We test with the IQL algorithm.

The results in Table 6 show that MARO can scale to a larger number of agents better than the MD baseline in the SpreadXY environment with the IQL algorithm. We can also see the performance of MARO against the MD baseline for specific communication levels in Figure 23



(a) HearSee.

(b) SpreadBlindfold.

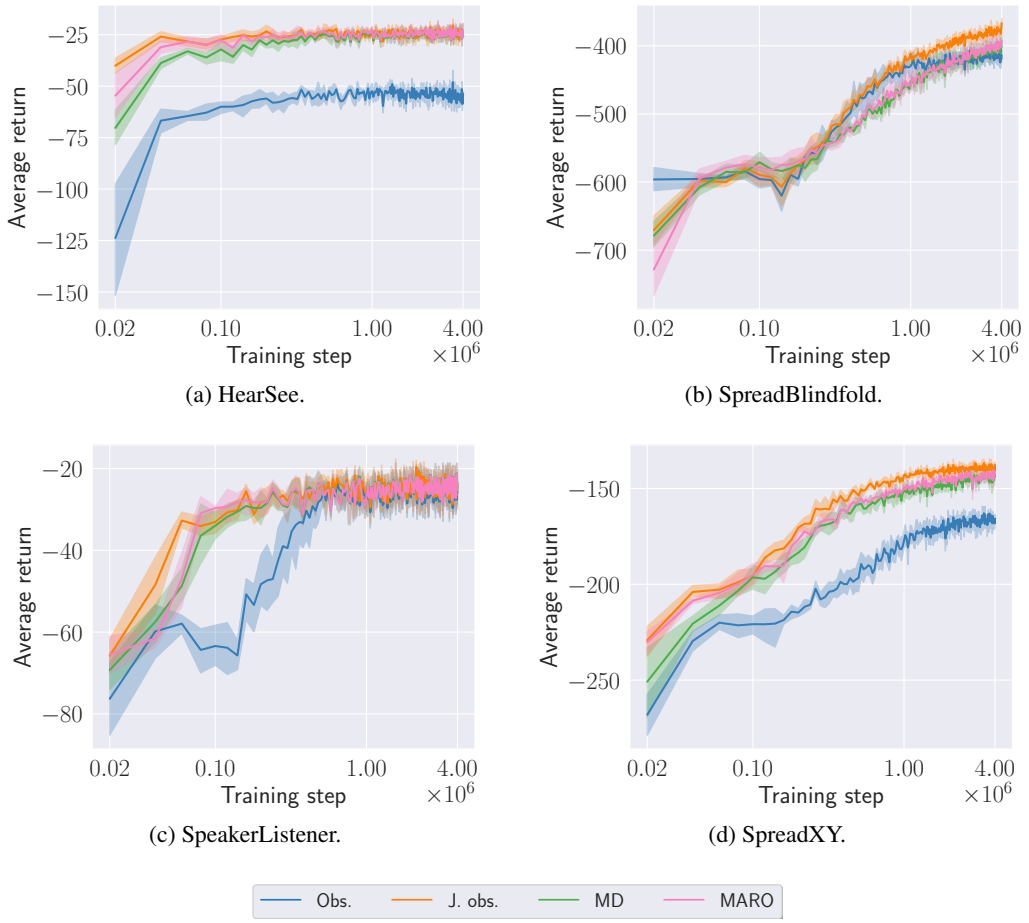
(c) SpeakerListener.

(d) SpreadXY.

Figure 7: Average episodic returns during training with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for all approaches and environments, using the IQL algorithm.

Table 6: Average episodic returns and 95% bootstrapped confidence interval over five seeds for all approaches in the SpreadXY4 environment under the IQL algorithm. Higher is better.

			IQL	
Environment	**Obs**	**J.Obs**	**MD**	**MARO**
SpreadXY4	-1194.5 (-9.0,+12)	-1099.4 (-11.3,+4.1)	-1176.9 (-5.7,+7.1)	**-1152.6** (-4.9,+5.0)



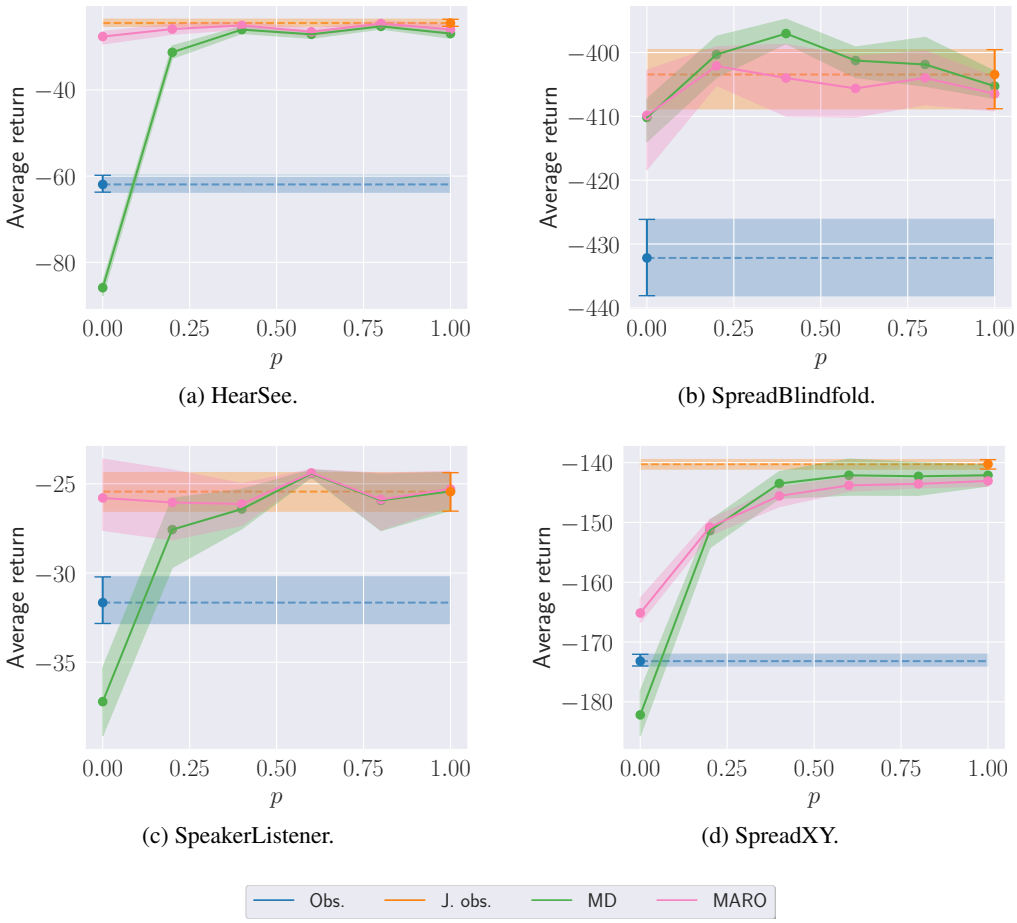
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 8: Average episodic returns during training with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for all approaches and environments, using the QMIX algorithm.



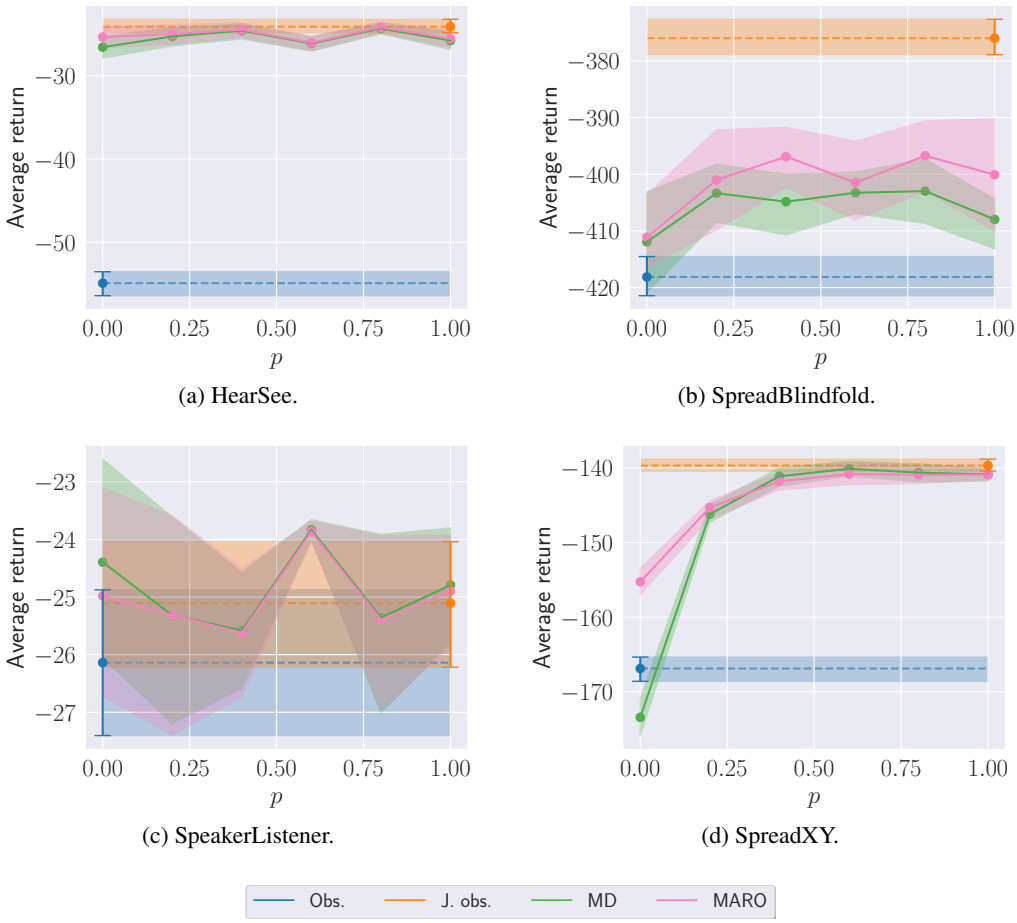
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 9: Average episodic returns with 95% bootstrapped confidence interval for different communication levels p at execution time, for all approaches across different environments, using the IQL algorithm.



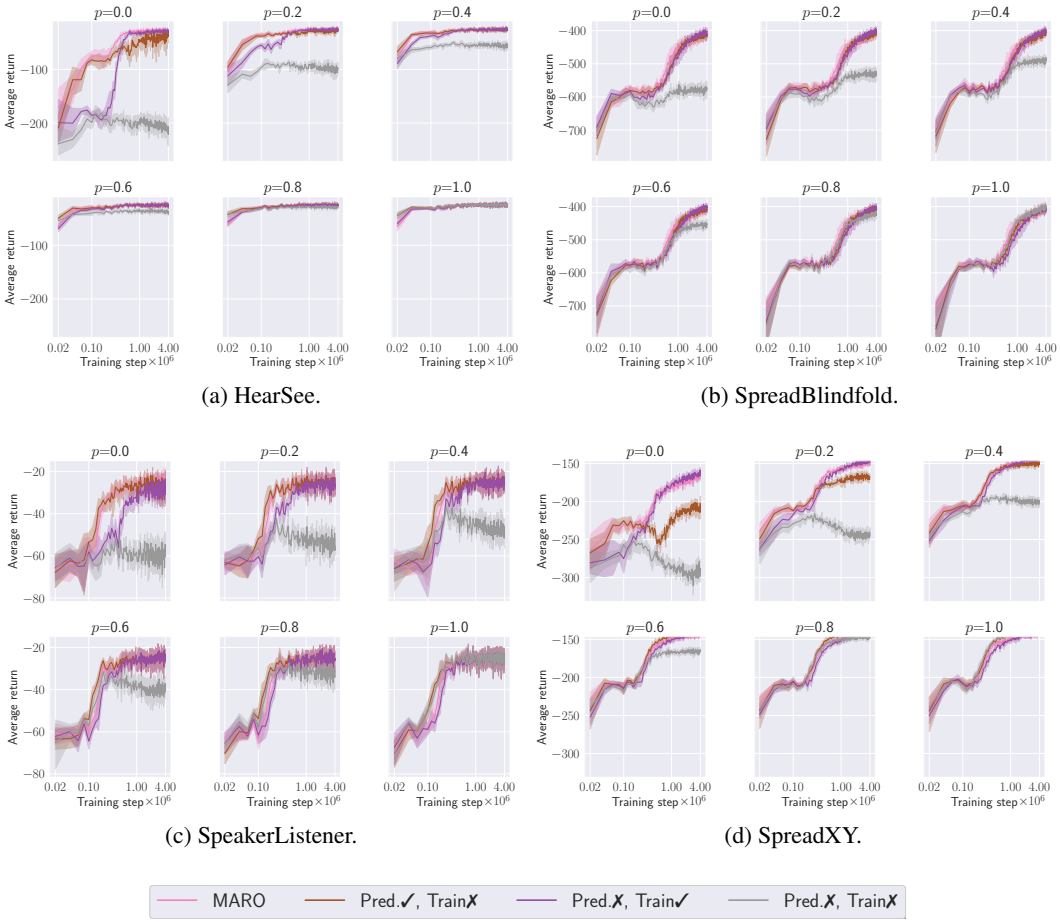
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 10: Average episodic returns with 95% bootstrapped confidence interval for different communication levels p at execution time, for all approaches across different environments, using the QMIX algorithm.



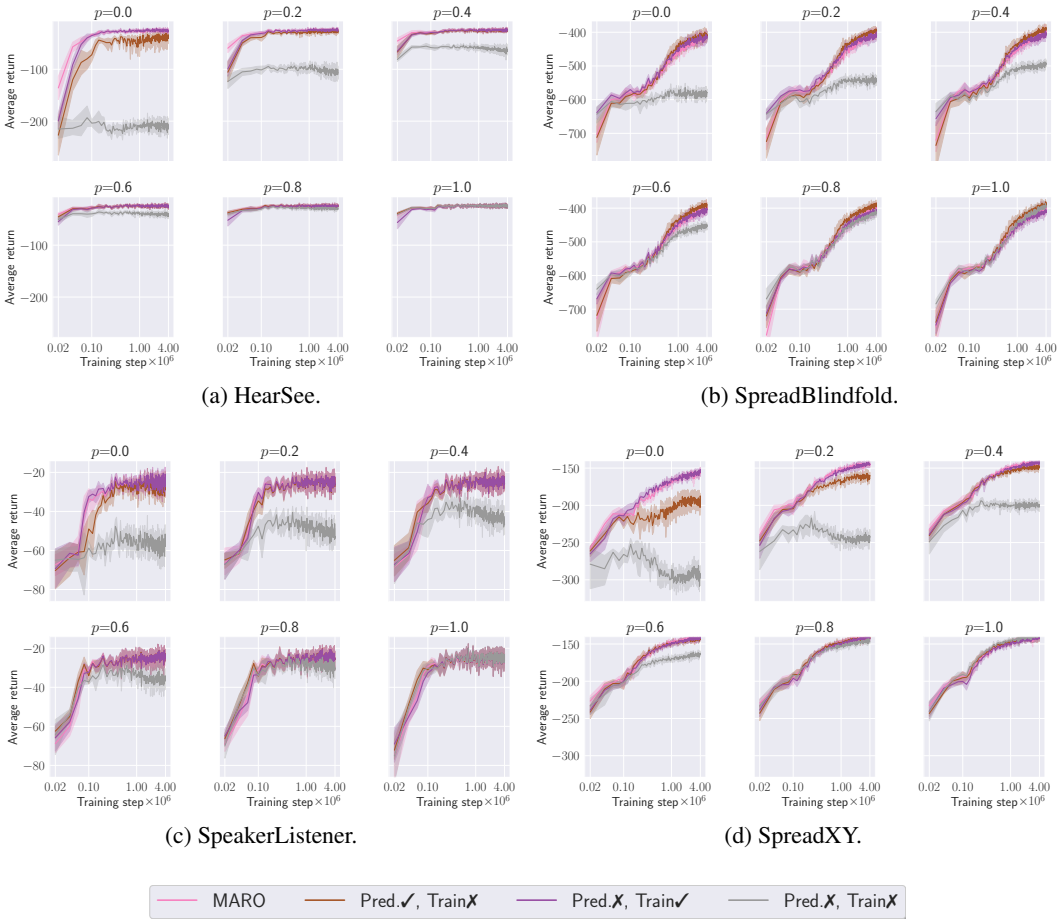
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 11: Average episodic returns during training with 95% bootstrapped confidence interval for different communication levels p, for MARO and the ablated versions across all environments, using the IQL algorithm.



(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 12: Average episodic returns during training with 95% bootstrapped confidence interval for different communication levels p, for MARO and the ablated versions across all environments, using the QMIX algorithm.

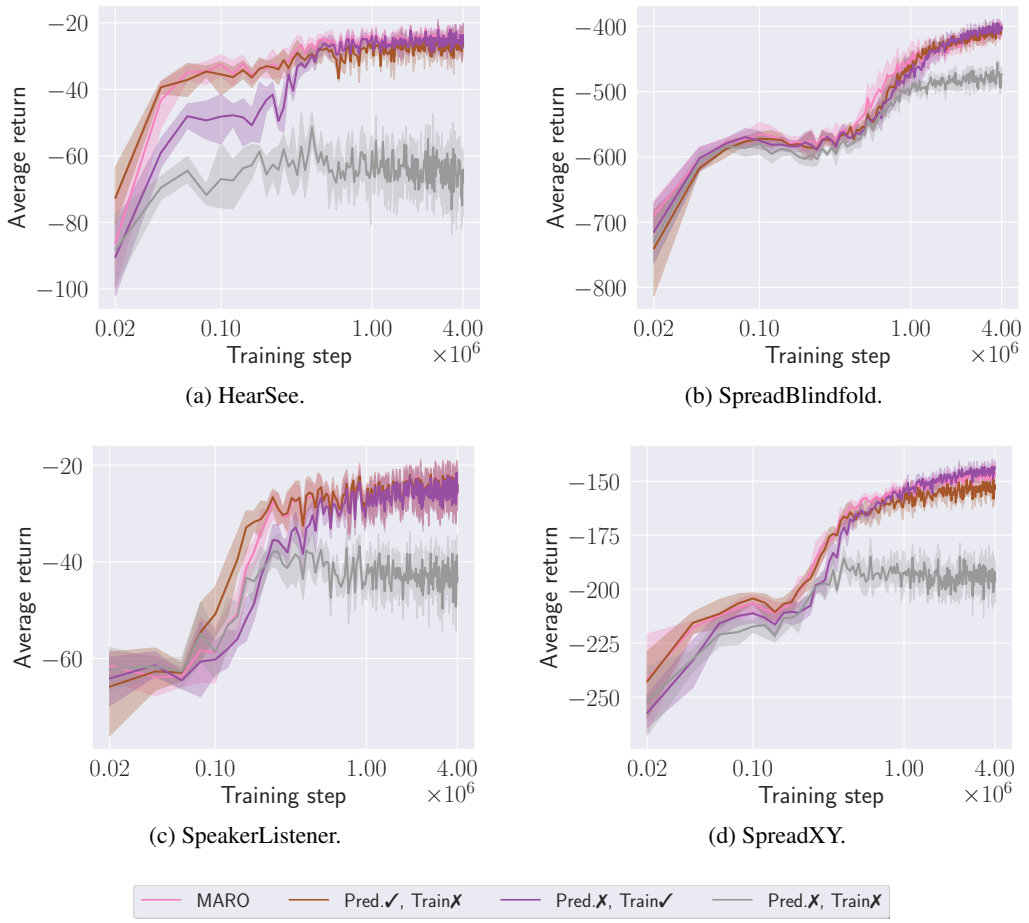
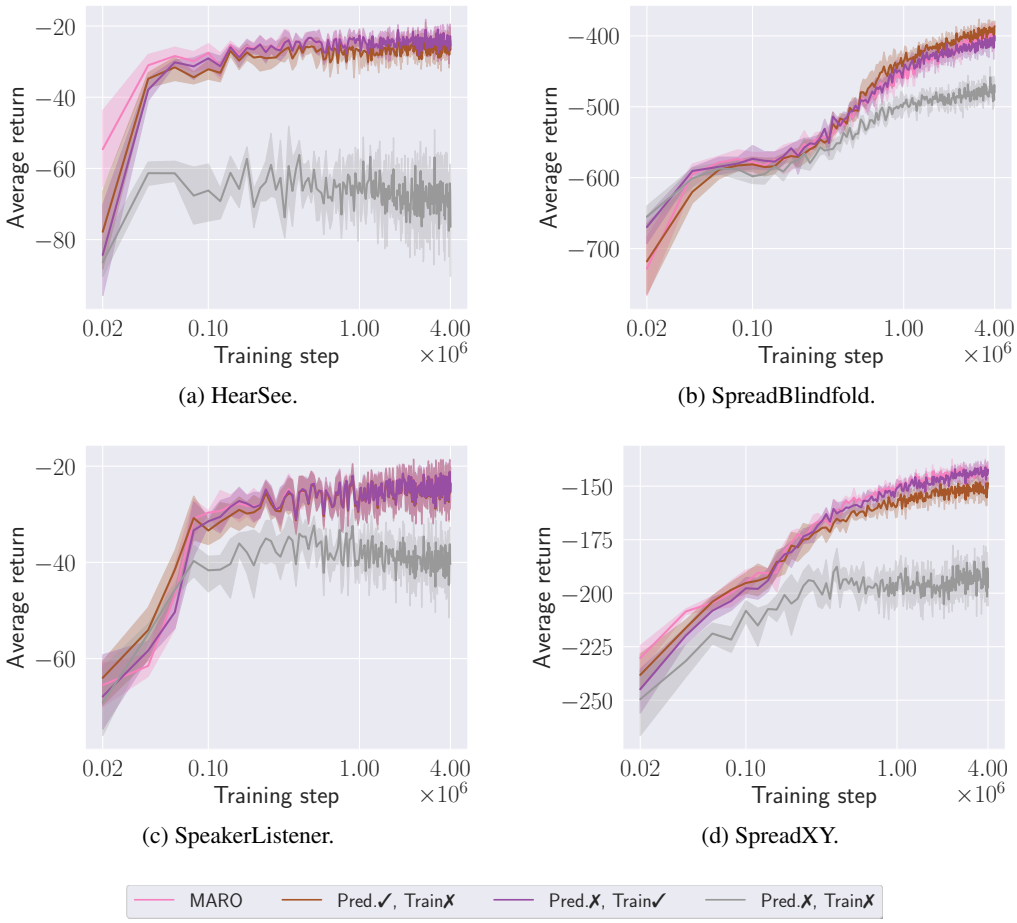


Figure 13: Average episodic returns during training with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for MARO and the ablated versions across all environments, using the IQL algorithm.



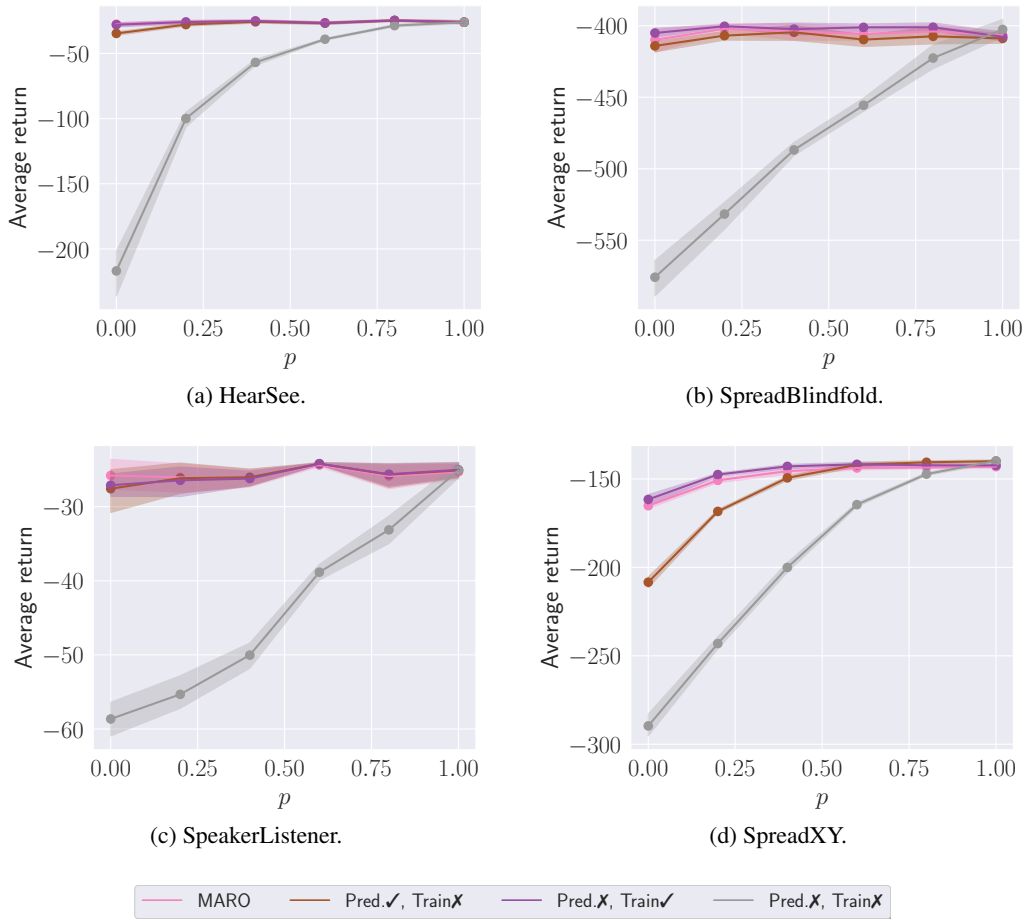
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 14: Average episodic returns during training with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for MARO and the ablated versions across environments, using the QMIX algorithm.



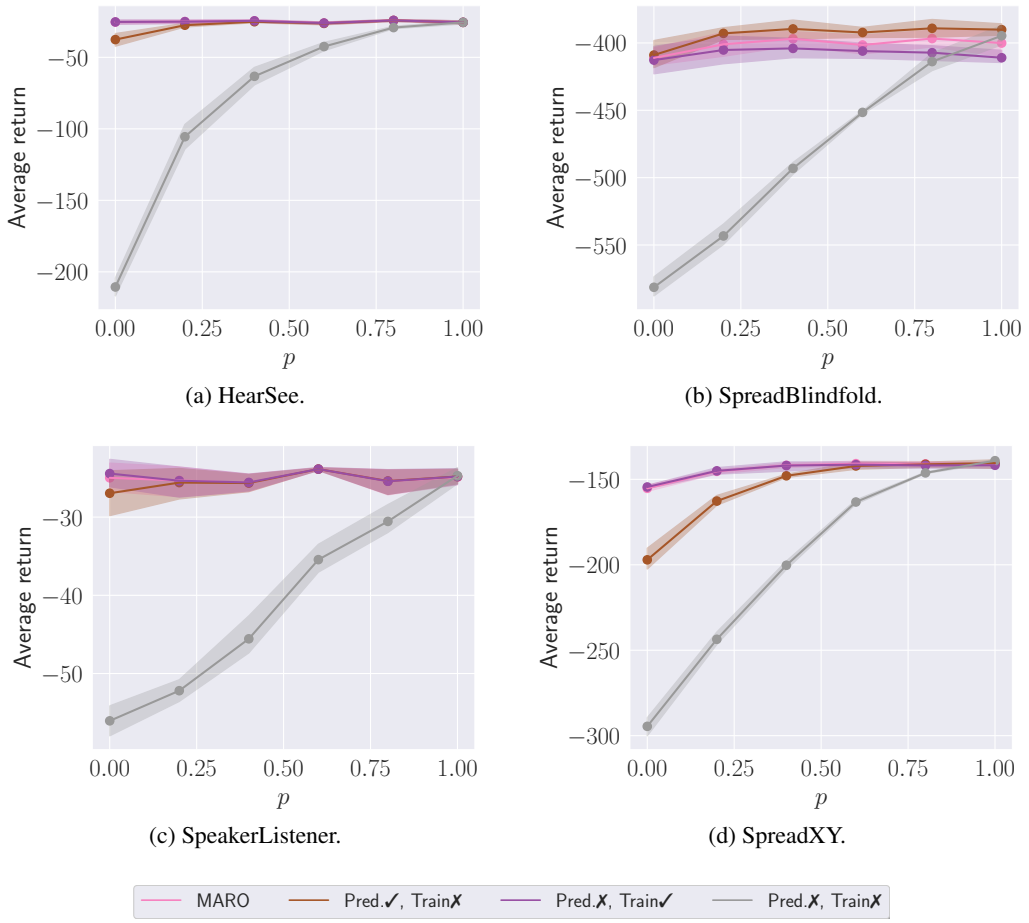
(a) HearSee.

(b) SpreadBlindfold.

(c) SpeakerListener.

(d) SpreadXY.

Figure 15: Average episodic returns with 95% bootstrapped confidence interval for different communication levels p at execution time, for MARO and the ablated versions across different environments, using the IQL algorithm.



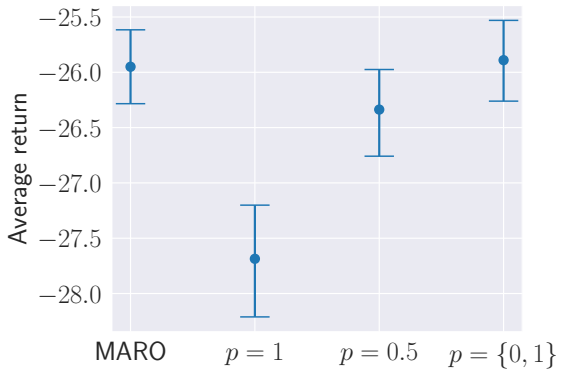
(a) HearSee.

(b) SpreadBlindfold.

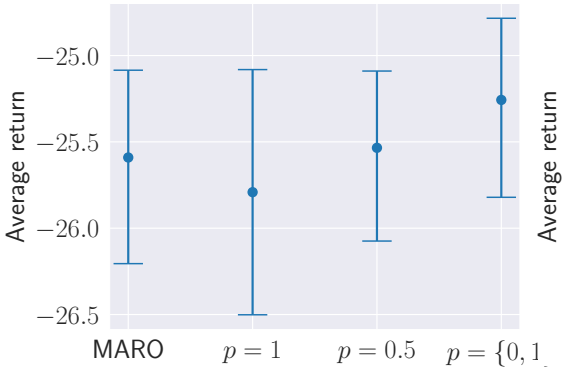
(c) SpeakerListener.

(d) SpreadXY.

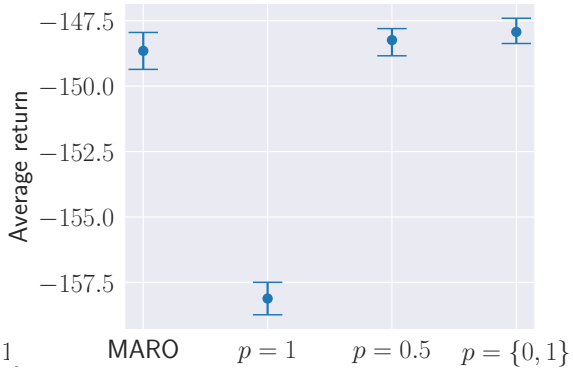
Figure 16: Average episodic returns with 95% bootstrapped confidence interval for different communication levels p at execution time, for MARO and the ablated versions across different environments, using the QMIX algorithm.



(a) HearSee.

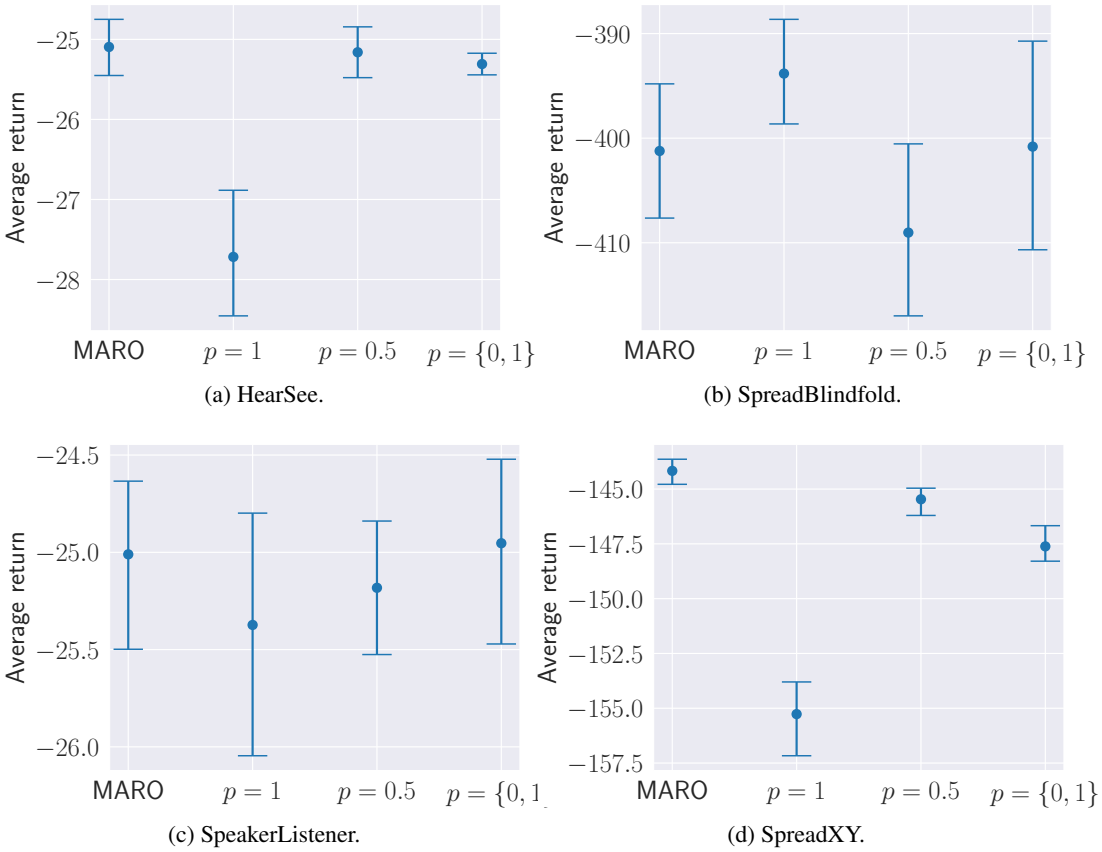


(b) SpeakerListener.



(c) SpreadXY.

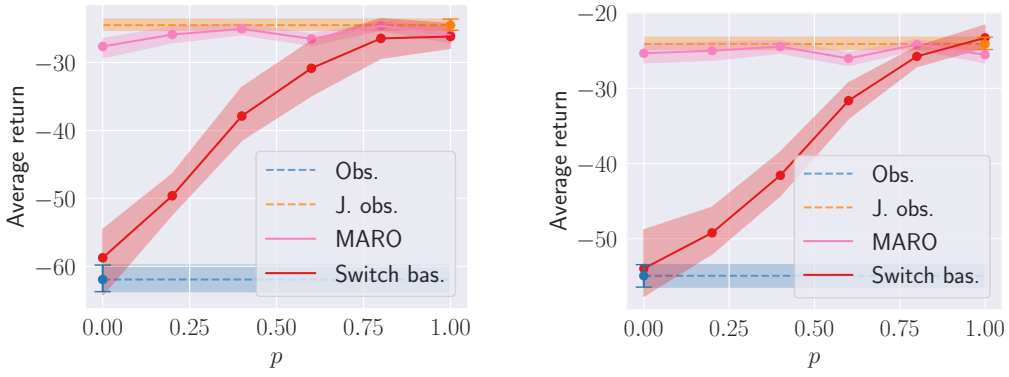
Figure 17: Average episodic returns at execution time with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for different sampling methods of the training scheme of MARO across environments, using the IQL algorithm.



(a) HearSee. (b) SpreadBlindfold.

(c) SpeakerListener. (d) SpreadXY.

Figure 18: Average episodic returns at execution time with 95% bootstrapped confidence interval for $p \sim \mathcal{U}(0, 1)$, for different sampling methods of the training scheme of MARO across environments, using the QMIX algorithm.



(a) IQL. (b) QMIX.

Figure 19: Average episodic returns with 95% bootstrapped confidence interval for different communication levels p at execution time for MARO and the Switch baseline in the HearSee environment.

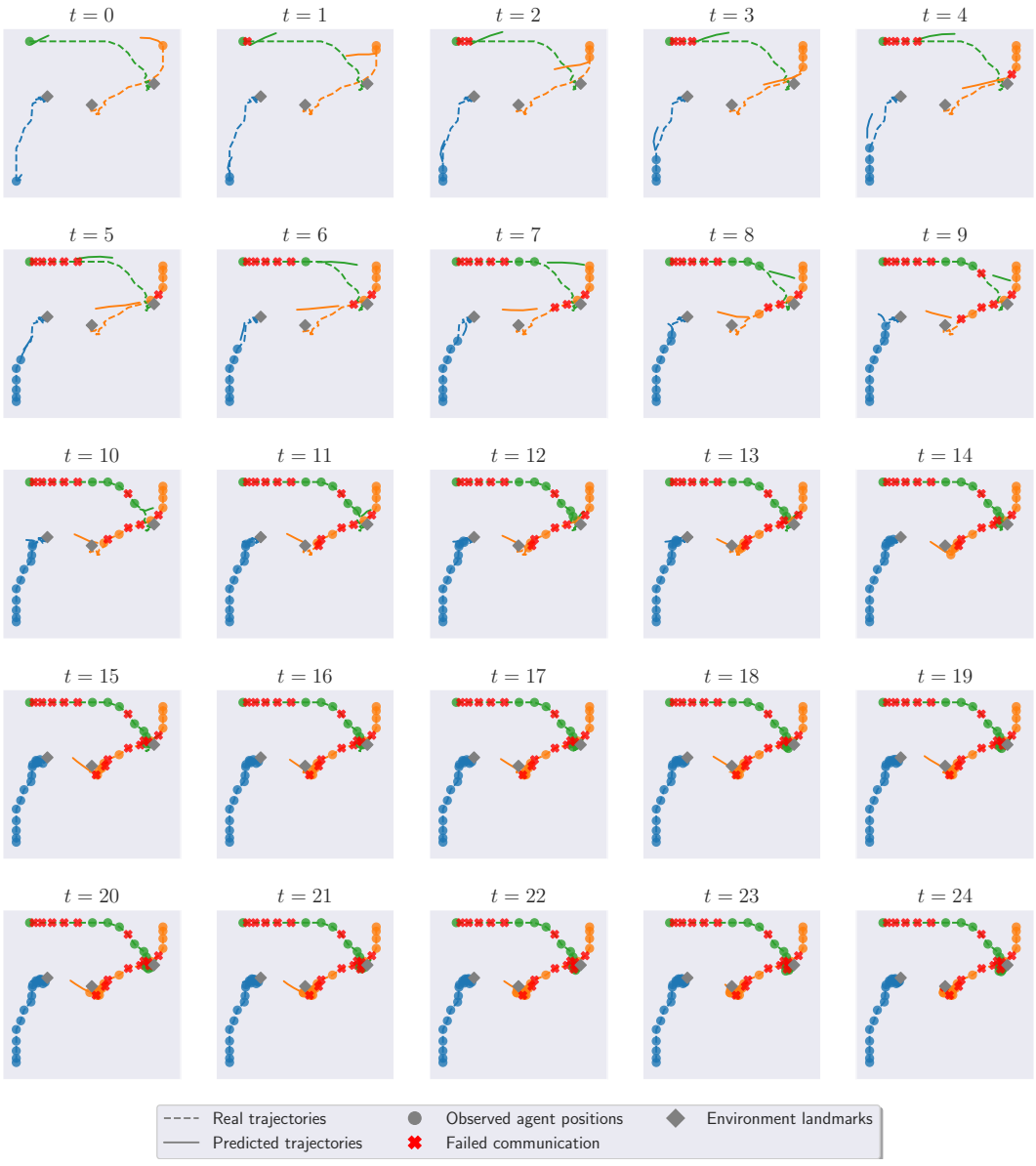


Figure 20: Trajectory prediction plots for the SpreadBlindfold environment under the QMIX algorithm from the perspective of agent 0 (blue).

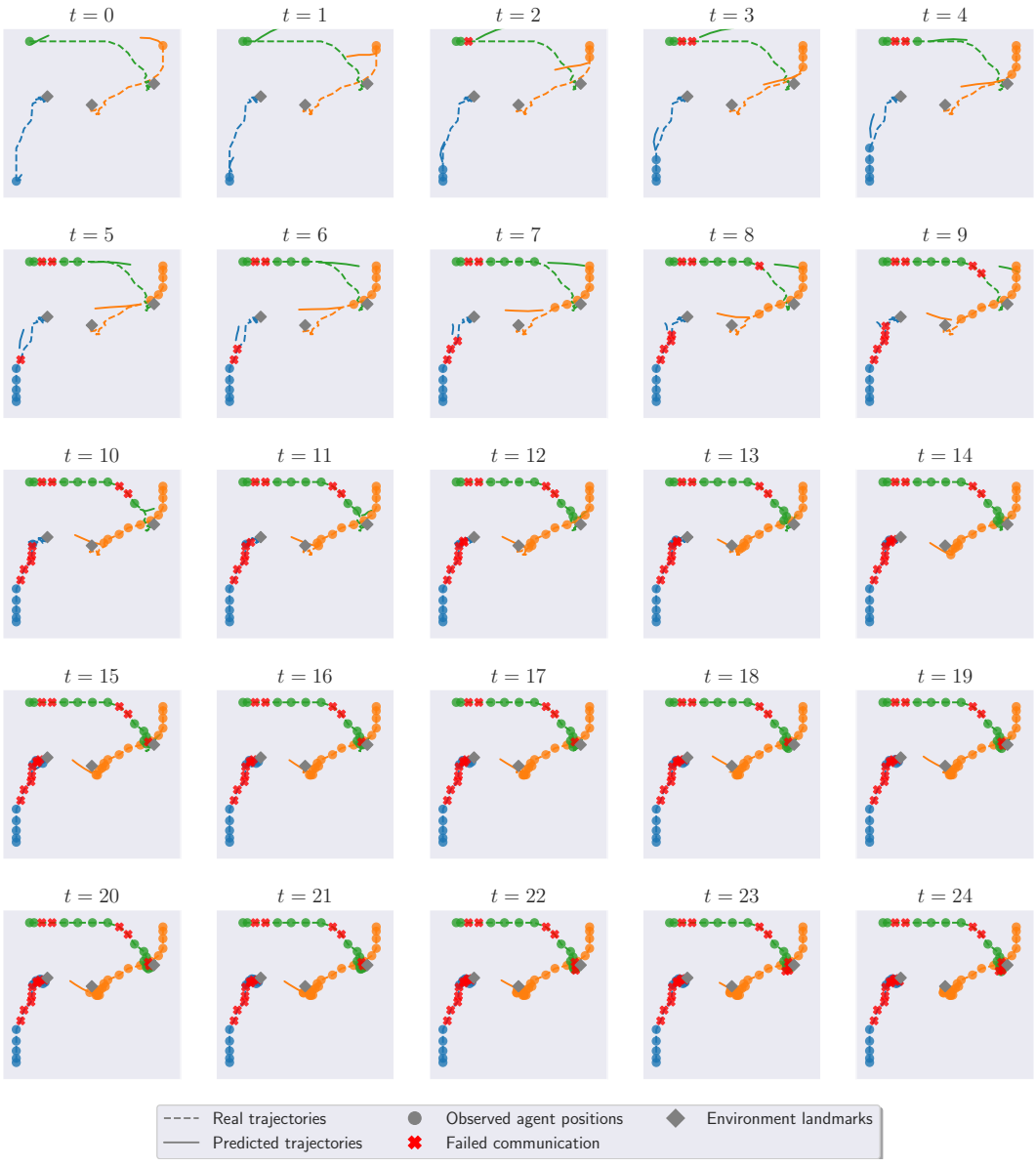


Figure 21: Trajectory prediction plots for the SpreadBlindfold environment under the QMIX algorithm from the perspective of agent 1 (orange).

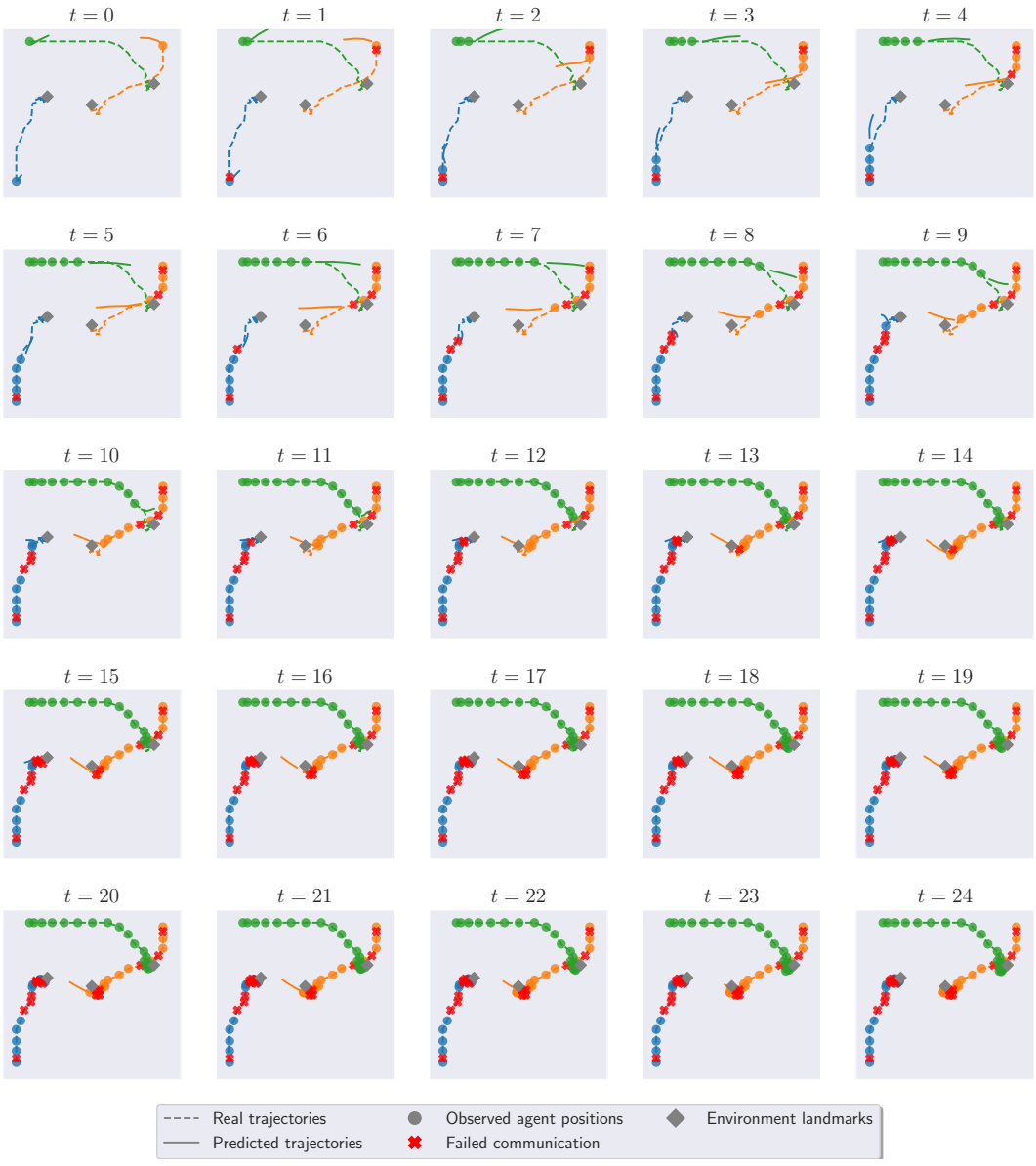


Figure 22: Trajectory prediction plots for the SpreadBlindfold environment under the QMIX algorithm from the perspective of agent 2 (green).

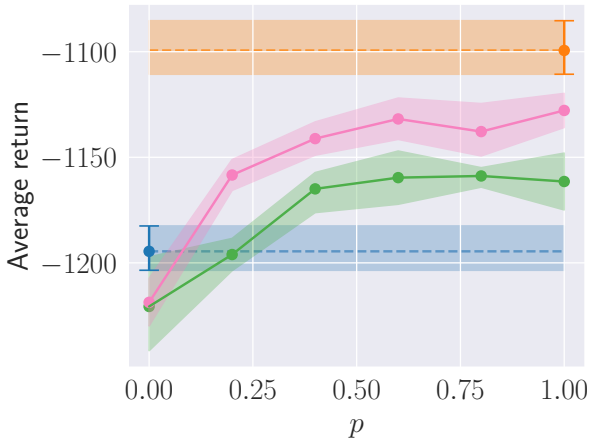


Figure 23: Performance across different communication levels on the SpreadXY environment with 4 agents with the IQL algorithm.