Ensemble Value Functions for Efficient Exploration in Multi-Agent Reinforcement Learning

Lukas Schäfer*
University of Edinburgh
Edinburgh, United Kingdom
l.schaefer@ed.ac.uk

Oliver Slumbers*
University College London
London, United Kingdom
o.slumbers@cs.ucl.ac.uk

Stephen McAleer

Yali Du
King‘s College London
London, United Kingdom
yali.du@kcl.ac.uk

Stefano V. Albrecht
University of Edinburgh
Edinburgh, United Kingdom
s.albrecht@ed.ac.uk

David Mguni
Huawei Technologies
London, United Kingdom
davidmguni@hotmail.com

arXiv:2302.03439v5 [cs.MA] 16 Apr 2023

ABSTRACT

Cooperative multi-agent reinforcement learning (MARL) requires agents to explore to learn to cooperate. Existing value-based MARL algorithms commonly rely on random exploration, such as ϵ-greedy, which is inefficient in discovering multi-agent cooperation. Additionally, the environment in MARL appears non-stationary to any individual agent due to the simultaneous training of other agents, leading to highly variant and thus unstable optimisation signals. In this work, we propose ensemble value functions for multi-agent exploration (EMAX), a general framework to extend any value-based MARL algorithm. EMAX trains ensembles of value functions for each agent to address the key challenges of exploration and non-stationarity: (1) The uncertainty of value estimates across the ensemble is used in a UCB policy to guide the exploration of agents to parts of the environment which require cooperation. (2) Average value estimates across the ensemble serve as target values. These targets exhibit lower variance compared to commonly applied target networks and we show that they lead to more stable gradients during the optimisation. We instantiate three value-based MARL algorithms with EMAX, independent DQN, VDN and QMIX, and evaluate them in 21 tasks across four environments. Using ensembles of five value functions, EMAX improves sample efficiency and final evaluation returns of these algorithms by 53%, 36%, and 498%, respectively, averaged all 21 tasks.

KEYWORDS

Multi-Agent Reinforcement Learning, Ensemble Models, Exploration

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) jointly trains a team of agents to exhibit behaviour which maximises shared cumulative rewards. MARL can tackle problems such as autonomous driving [32, 39] and warehouse logistics [12, 14], but its real-world adaptation is still limited. Two remaining challenges of MARL are the large number of samples required to learn cooperation and the non-stationarity of the optimisation due to agents learning simultaneously [25].

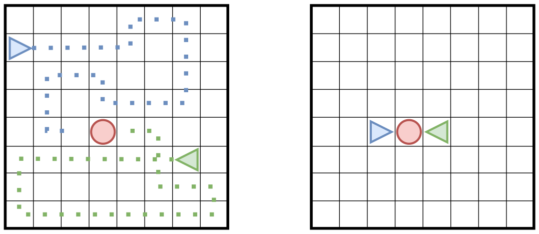


Figure 1: Motivational example: Two agents (triangles) need to cooperate to pick-up a heavy object (red circle). Agents can individually explore their movement (left), but random exploration is inefficient in discovering the cooperation of both agents to pick-up the heavy goal object (right). To overcome this inefficiency, we leverage uncertainty across ensembles of value functions to guide multi-agent exploration towards state-action pairs which require cooperation.

MARL algorithms have shown good performance in various cooperation tasks [26], but value-based MARL algorithms still rely on random exploration processes, such as ϵ-greedy (e.g. Rashid et al. [29], Sunehag et al. [35]). We argue that random exploration is inefficient in exploring the joint action space of all agents to discover cooperation in MARL. To illustrate this inefficiency, consider the following example in which two agents have to navigate an environment to jointly pick-up a heavy object, visualised in Figure 1. Agents can navigate within the shared environment by themselves and thus individually explore their movement. To discover the desired cooperation, both agents have to pick-up the object at the same time. However, this behaviour is highly unlikely following random exploration, leading to poor sample efficiency in tasks which require cooperation.

To address this inefficient exploration for learning coordinated behaviour, it is essential for agents to focus their exploration on parts of the environment which require cooperation. To this end, we propose *ensemble value functions for multi-agent exploration* (EMAX), a general framework to extend any value-based MARL algorithms by training ensembles of value functions for each agent. EMAX guides the exploration of agents towards parts of the environment with significant disagreement of value estimates, given by the deviation of estimates across the ensemble. The key insight into our exploration is that disagreement within the ensemble of value

[*] Work was conducted during an internship at Huawei Technologies.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

functions indicates the possibility of a state-action pair being lucrative and therefore, its importance for exploration. For state-action pairs where little exploration and cooperation is needed, such as the navigation of the agents in our example, disagreement quickly diminishes. However, for state-action pairs where cooperation is needed, such as the picking-up of the heavy object, agents will receive highly variant rewards because they both fail and succeed in cooperating. This variance in received rewards causes a high disagreement of value estimates across the ensemble. Therefore, agents can follow this disagreement to guide their exploration using an upper-confidence bound (UCB) [4] policy. Moreover, EMAX computes average value estimates across the ensemble as target values instead of using target networks. These target values exhibit lower variance [15], eliminate the need for target networks, and stabilise the optimisation of agents.

In a simplified setting of a common-reward normal form game, we demonstrate that EMAX focuses its exploration on parts of the environment which require cooperation, and thereby improves sample efficiency and convergence to high-reward cooperation policies (Section 5). We instantiate three value-based MARL algorithms, independent DQN [21], VDN [35], and QMIX [29], with EMAX and compare them against the corresponding vanilla algorithms in 21 tasks across four diverse multi-agent environments. EMAX improves sample efficiency and final achieved returns across all tasks over all three vanilla algorithms by 53%, 36%, and 498%, respectively, and is shown to reliably reduce variance of gradients throughout optimisation, leading to more stable training (Section 6.2). Lastly, we show that comparably small ensembles with five value functions are sufficient to benefit from the advantages of EMAX and discuss the computational cost of ensemble models.

2 RELATED WORK

In this section, we discuss existing research on ensemble models for single-agent reinforcement learning (RL), and discuss prior MARL research addressing the challenge of exploration as well as how our approach compares to them.

Ensemble models in RL: Several single-agent RL algorithms train ensembles of value functions. Bootstrapped DQN [23] applies ensemble value functions for exploration by randomly sampling a single value function to greedily follow at the beginning of each episode. SUNRISE [13] and MeanQ [15] apply UCB using the average and standard deviation of value estimates across the ensemble to explore. Moreover, SUNRISE uses the ensemble to weight the value loss based on the variance of target values across the ensemble. MeanQ stabilises the optimisation by computing target values with an average value estimate across the ensemble which is shown to reduce the variance of value estimates [3]. Considering the multi-agent problem, EMAX adapts several of these techniques and integrates them into value-based MARL algorithms. Concurrently to our work, Shen and How [33] proposed latent-conditioned policies to approximate ensemble training for robust MARL, but their work focuses on competitive policy-gradient algorithms.

Multi-agent exploration: For the multi-agent setting, Wang et al. [37] incentivise agents to interact with each other by intrinsically rewarding them for mutually influencing their transition dynamics or value estimates. Similar intrinsic rewards can be assigned for reaching goal states to train separate exploration policies [16]. However, intrinsic rewards for exploration have to be carefully balanced for each task due to the modified optimisation objective [31]. To address this challenge, LIGS [20] formulate the assignment of intrinsic rewards as a MARL problem and train an agent to determine when and which intrinsic reward should be given to each agents. Experience and parameter sharing have been leveraged to greatly improve sample efficiency for MARL by synchronising agents' learning and make use of more data [7, 8]. REMAX [30] identifies valuable initial states for episodes to guide exploration based on a latent representation of states learned using the interactions of agents in the environment. However, there is little research using distributional and ensemble-based techniques for MARL exploration. Zhou et al. [40] extend posterior sampling [24] for MARL, but are limited to two-player zero-sum extensive games. We aim to close this gap by proposing EMAX, an ensemble-based technique for efficient exploration in cooperative MARL. We further highlight that EMAX is a plug-and-play algorithm that can enhance any value-based MARL algorithm, including most existing MARL exploration techniques described in this paragraph.

3 BACKGROUND

3.1 Decentralised Partially Observable Markov Decision Process

We formalise cooperative multi-agent environments as decentralised partially observable Markov decision processes (Dec-POMDP) [28] defined by $(\mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, \{O_i\}_{i\in\mathcal{I}}, \mathcal{P}, \mathcal{R}, \Omega)$. Each agent is indexed by $i \in \mathcal{I} = \{1, \dots, N\}$. $\mathcal{S}$ denotes the state space of the environment. Agents receive local observation which are drawn from their observation space O_i and take actions from their action space $\mathcal{A}_i$. We denote the space of joint observations and actions across all agents with $O = O_1 \times \ldots \times O_N$ and $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_N$, respectively. The observation function $\Omega : \mathcal{S} \times \mathcal{A} \times O \mapsto [0, 1]$ determines a distribution over joint observations given the current state and taken joint action. Given the current state and the joint action, the transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ and reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ define a distribution over the successor state of the environment and a scalar reward shared across all agents, respectively. Each agent i only receives its local observation $o^i_t = \Omega(s_t, a_t)_i$ at timestep t and learns a policy $\pi_i : \mathcal{H}_i \times \mathcal{A}_i \mapsto [0, 1]$ defining its action probabilities given the episodic history of actions and observations $h_i = \left(a^i_{t-1}, o^i_t\right)_{t\geq 1} \in \mathcal{H}_i$. Each agent optimises its policy with the objective of learning a joint policy $\pi = (\pi_1, \dots, \pi_N)$ such that $\pi \in \arg\max_{\pi'} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \mathcal{R}(s_t, a_t)\right]$ with discount factor $\gamma \in [0, 1)$.

3.2 Value-Based Multi-Agent Reinforcement Learning

Independent Q-learning: Independent deep Q-network (IDQN) extends DQN [21] for MARL and independently learns a value function Q_i, parameterised by θ_i, for each agent i. Agents store tuples (s, o, a, r, s', o') of experience consisting of state s, joint observation o, applied joint action a, received reward r, next state s', and next joint observation o', respectively, in a replay buffer. The value function of agent i is then optimised by minimising the average loss

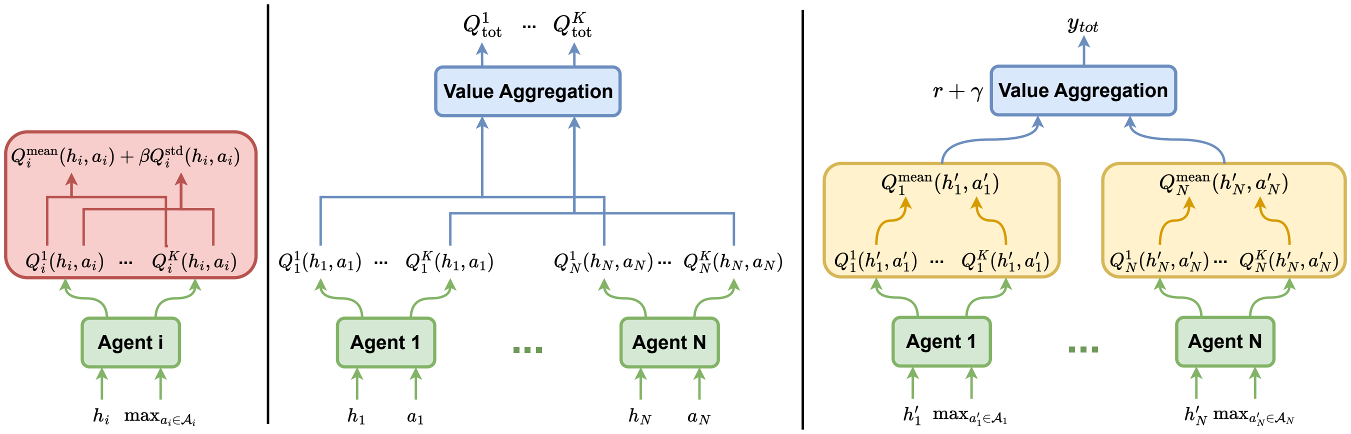

$Q_i^{\text{mean}}(h_i,a_i) + \beta Q_i^{\text{std}}(h_i,a_i)$
$Q_i^1(h_i,a_i)$ ··· $Q_i^K(h_i,a_i)$
Agent i
h_i $\max_{a_i \in \mathcal{A}_i}$
Q_{tot}^1 ··· Q_{tot}^K
Value Aggregation
$Q_1^1(h_1,a_1)$ ··· $Q_1^K(h_1,a_1)$
$Q_N^1(h_N,a_N)$ ··· $Q_N^K(h_N,a_N)$
Agent 1
...
Agent N
h_1 a_1
h_N a_N
y_{tot}
$r+\gamma$
Value Aggregation
$Q_1^{\text{mean}}(h'_1,a'_1)$
$Q_N^{\text{mean}}(h'_N,a'_N)$
$Q_1^1(h'_1,a'_1)$ ··· $Q_1^K(h'_1,a'_1)$
$Q_N^1(h'_N,a'_N)$ ··· $Q_N^K(h'_N,a'_N)$
Agent 1
...
Agent N
h'_1 $\max_{a'_1 \in \mathcal{A}_1}$
h'_N $\max_{a'_N \in \mathcal{A}_N}$

Figure 2: Illustration of EMAX with (left) the UCB exploration strategy for agent i, (middle) the computation of value estimates, and (right) the target computation. Computation of individual agent value functions are highlighted in green, exploration in red, value aggregation for value decomposition algorithms in blue, and target aggregation in orange.

across sampled batches of experience:

$$\mathcal{L}(\theta_i) = \left[Q_i(h_i,a_i) - r - \gamma \max_{a'_i \in \mathcal{A}_i} \bar{Q}_i(h'_i,a'_i) \right]^2 \tag{1}$$

with $\bar{Q}_i$ denoting a target network with parameters $\bar{\theta}_i$ which are periodically copied from θ_i.

Value decomposition: Independent learning serves as an effective baseline in many cooperative MARL tasks [26] but suffers from the non-stationarity arising from the learning of other agents [25] and the multi-agent credit assignment problem, i.e. agents need to identify their individual contribution to received rewards [10, 29]. Value decomposition algorithms address this latter multi-agent challenge using a centralised state-action value function Q_{tot}, conditioned on the state and joint action of all agents. In environments, where the state is not available during training, we approximate the state with the joint observation. Directly learning such a value function is often computationally infeasible due to the exponential growth of the joint action space with the number of agents, so the centralised value function is approximated with an aggregation of individual value functions of each agent conditioned on the local observation-action history. The value functions and aggregation are optimised by minimising the joint value function loss with target values y_{tot}:

$$\mathcal{L}(\theta) = \left[Q_{\text{tot}}(s,a) - y_{\text{tot}} \right]^2 \tag{2}$$

Two common value decomposition algorithms are VDN [35] and QMIX [29]. VDN assumes a linear aggregation of the centralised value function and targets

$$Q_{\text{tot}}(s,a) = \sum_{i \in \mathcal{I}} Q_i(h_i,a_i) \tag{3}$$

$$y_{\text{tot}} = r + \gamma \max_{a' \in \mathcal{A}} \sum_{i \in \mathcal{I}} \bar{Q}_i(h'_i,a'_i) \tag{4}$$

and QMIX assumes a less restrictive monotonic mixing function of individual values

$$\begin{aligned} Q_{\text{tot}}(s,a) &= f_m\left(Q_1(h_1,a_1),\dots,Q_N(h_N,a_N)\right) \\ y_{\text{tot}} &= r + \gamma \max_{a' \in \mathcal{A}} \bar{f}_m\left(\bar{Q}_1(h'_1,a'_1),\dots,Q_N(h'_N,a'_N)\right) \end{aligned} \tag{5}$$

with f_m and $\bar{f}_m$ denoting the deep monotonic mixing function and a delayed target mixing function, respectively.

4 ENSEMBLE VALUE FUNCTIONS FOR MULTI-AGENT REINFORCEMENT LEARNING

In this section, we present ensemble value functions for multi-agent exploration (EMAX), a general framework to leverage ensembles of value functions for improved exploration and stable optimisation in value-based MARL. Following the intuition that the disagreement of value estimates, given by the standard deviation of value estimates across the ensemble, indicates the degree of required cooperation and exploration of states and actions, agents follow a UCB policy to guide their exploration. To stabilise the optimisation, low variance target estimates are computed across the ensemble. In the following, we define the training of ensembles of value functions for IDQN, its integration into value decomposition methods such as VDN and QMIX, and describe the exploration and evaluation policies of EMAX. Figure 2 illustrates the architecture of our approach.

Independent target computation: In the case of independent learning with ensemble value functions, each agent i trains an ensemble of K value functions $\{Q_i^k\}_{k=1}^K$ with Q_i^k being parameterised by θ_i^k. Each value function is conditioned on agent i's local observation-action history. For the following, we define the average and standard deviation of value estimates across the ensemble of agent i:

$$Q_i^{\text{mean}}(h, a) = \frac{1}{K} \sum_{k=1}^{K} Q_i^k(h, a) \tag{6}$$

$$Q_i^{\text{std}}(h, a) = \sqrt{\frac{\sum_{k=1}^{K} \left(Q_i^k(h, a) - Q_i^{\text{mean}}(h, a) \right)^2}{K}} \tag{7}$$

To optimise the kth value function of agent i, we minimise the following loss:

$$\mathcal{L}(\theta_i^k) = \left[Q_i^k(h_i, a_i) - r - \gamma \max_{a_i' \in \mathcal{A}_i} Q_i^{\text{mean}}(h_i', a_i') \right]^2 \tag{8}$$

Computing target values as the average across all value estimates of the ensemble [15] reduces the computational and memory cost of training ensemble networks by alleviating the need for target networks and, as we empirically show later, reduces the variability of gradients. Such reduced variability of gradients improves the stability of training and is particularly valuable in MARL where non-stationarity can make training otherwise unstable.

Value decomposition: Value decomposition techniques such as VDN [35] and QMIX [29] can naturally be extended with EMAX to benefit from its improved training stability. In this case, each agent trains an ensemble of independent value functions as proposed above. The total loss for the kth value functions of all agents with parameters θ^k is given by Equation (2) with centralised value function and targets for VDN

$$\begin{aligned} Q_{\text{tot}}^k(s, a) &= \sum_{i \in \mathcal{I}} Q_i^k(h_i, a_i) \\ y_{\text{tot}} &= r + \gamma \max_{a' \in \mathcal{A}} \sum_{i \in \mathcal{I}} Q_i^{\text{mean}}(h_i', a_i') \end{aligned} \tag{9}$$

and QMIX shown in Equations (9) and (10), respectively.

$$\begin{aligned} Q_{\text{tot}}^k(s, a) &= f_m \left(Q_1^k(h_1, a_1), \dots, Q_N^k(h_N, a_N) \right) \\ y_{\text{tot}} &= r + \gamma \max_{a' \in \mathcal{A}} \bar{f}_m \left(Q_1^{\text{mean}}(h_1', a_1'), \dots, Q_N^{\text{mean}}(h_N', a_N') \right) \end{aligned} \tag{10}$$

The aggregation of QMIX is able to represent a wider set of centralised value functions, but VDN has been shown to be more sample efficient in tasks which do not seem to require a non-linear aggregation for effective cooperation [26]. Therefore, we consider both the extension of VDN and QMIX with EMAX.

Exploration policy: In cooperative MARL, agents should focus their exploration on actions and parts of the state space which require cooperation to achieve high rewards. To incentivise such exploration with ensemble value functions, agent i follows a UCB policy akin to SUNRISE [13] and MeanQ [15]

$$\pi_i^{\text{expl}}(h_i) \in \arg\max_{a \in \mathcal{A}_i} Q_i^{\text{mean}}(h_i, a) + \beta Q_i^{\text{std}}(h_i, a) \tag{11}$$

with uncertainty weighting hyperparameter $\beta > 0$ chosen in consideration of the scale of rewards and the amount of exploration required for a task. This exploration strategy, in contrast to common random exploration for value-based MARL such as ϵ-greedy policies, uses uncertainty over value estimates, given by the disagreement of value estimates across the ensemble, to guide exploration. Value estimates in parts of the environment which require no or limited exploration across agents will quickly converge, leading to low disagreement in ensemble value functions and hence less exploration. In contrast, value estimates in parts of the environment which require cooperation will experience large disagreement due to agents often failing to cooperate and thereby experiencing highly varying rewards. This leads to UCB with ensemble value functions focusing its exploration on parts of the environments which are most interesting for exploration. We empirically demonstrate these benefits in Section 5.

Evaluation policy: When evaluating agents, value-based MARL algorithms typically follow the greedy policy with respect to their value function. With EMAX, agent i selects its action during evaluation using a majority vote across the greedy actions of all models in its ensemble

$$\begin{aligned} \pi_i^{\text{eval}}(h_i) &\in \arg\max_{a \in \mathcal{A}_i} \sum_{k=1}^{K} 1_{\mathcal{A}opt_i^k}(a) \\ \mathcal{A}opt_i^k &= \{ a' \in \mathcal{A}_i \mid a' \in \arg\max_a Q_i^k(h_i, a) \} \end{aligned} \tag{12}$$

with indicator function $1_{\mathcal{A}opt_i^k}$ for the greedy action of Q_i^k. Such a policy decreases the likelihood of taking poor actions because any individual value function preferring a poor action due to errors in value estimates does not impact the action selection as long as the majority of models agree on the optimal action. We empirically study this effect in Appendix D.

Ensemble diversity: All aforementioned ensemble value function techniques rely on value functions within the ensemble staying sufficiently diverse, in particular early in training. Similar to MeanQ [15], we apply three ideas to ensure diversity: (1) Ensemble models are separately and randomly initialised. (2) Each model is trained on bootstrapped samples of the entire experience collected. (3) We sample separate batches of experience from the replay buffer to train each model in the ensemble.

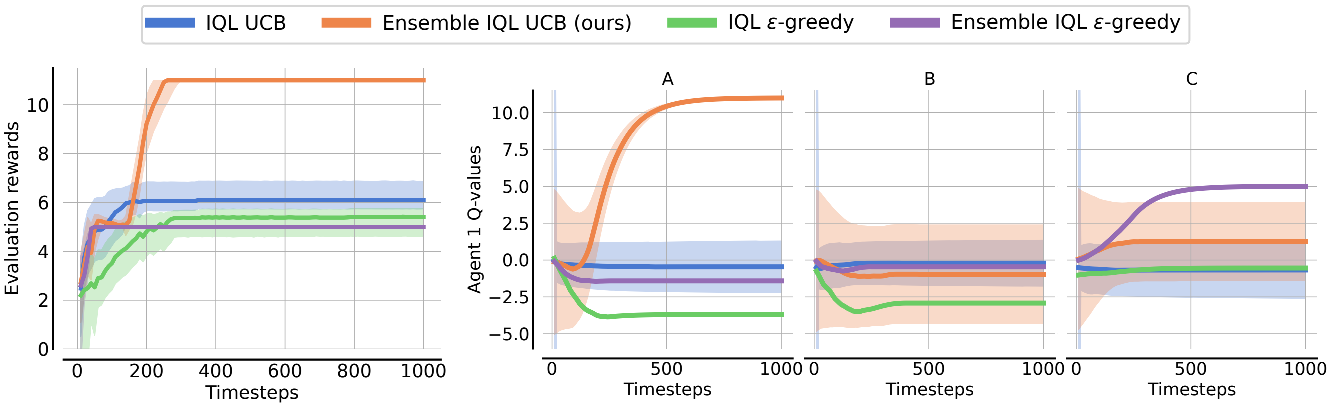
5 DIDACTIC EXAMPLE: EXPLORATION IN TWO-PLAYER COMMON-REWARD MATRIX GAME

In this section, we consider a simplified setting of single-stage (stateless) two-player common-reward matrix games using the example of the climbing game [9]. Table 1 shows the reward table for this game with both agents choosing between actions A, B and C.

		Agent 2		
		A	B	C
Agent 1	A	11	−30	0
	B	−30	7	6
	C	0	0	5

Table 1: Climbing game reward table.

Converging to the optimal cooperation policy of (A, A) in this game is difficult because any agent deviating from this joint policy leads to significant penalties, so agents might converge to suboptimal policies with lower risk. This setting does not require target computation (due to single stage immediate rewards), hence we



(a) Evaluation rewards

(b) Q-values of one agent for actions A, B, C.

Figure 3: Tabular independent Q-learning (IQL) agents trained in the climbing game. We visualise (a) evaluation rewards (interquartile mean with 95% confidence intervals) and (b) Q-values with shading indicating UCB uncertainty throughout training, both given by the interquartile mean of Q-value estimates and uncertainty, respectively. Our approach with ensemble value functions and UCB (orange) consistently converges to the optimal joint policy (A, A), whereas all baselines converge to suboptimal policies in most runs.

can study the impact of our proposed UCB exploration to discover optimal cooperation policies in isolation.

We evaluate *tabular* independent Q-learning (IQL) [36] with and without our ensemble value functions using ϵ-greedy and UCB exploration policies. In order to represent initial uncertainty about value estimates, we initialise value functions of all algorithms with a zero-mean Gaussian distribution. For more details on value initialisation and hyperparameters for all algorithms, see Appendix A.1. Results are reported over 100 runs. Figure 3a shows that only our approach with ensemble value functions and UCB exploration (orange) robustly converges to the optimal solution with a reward of 11, whereas all baselines converge to suboptimal policies in most runs.

Inspecting the Q-values throughout training, visualised in Figure 3b, explains why UCB exploration with ensemble value functions is effective: Early in training, all value estimates are centred at zero with high uncertainty, but uncertainty gradually decreases as the agents explore. In particular the uncertainty of value estimates for actions B and C are quickly reduced due to low variance in received rewards in comparison to A. For action A, exploring agents will sometimes receive very high and sometimes very low rewards, leading to high disagreement of value estimates for A across the ensemble. With high disagreement of value estimates for action A and decreasing disagreement for other actions, both agents will eventually start to continually choose action A which allows them to converge to the optimal policy of (A, A). In contrast, UCB exploration *without* ensemble value functions computes its exploration policy as

$$\pi_i^{\text{expl}} \in \arg\max_{a \in \mathcal{A}_i} Q_i(a) + \beta \frac{t}{N_i(a)} \tag{13}$$

with counts of actions being used to approximate the uncertainty. This measure of uncertainty does not reflect the true variance of received rewards and hence does not benefit from a similar effect of focusing exploration on actions which particularly require cooperation. Likewise, ϵ-greedy explores uniformly at random, so successful cooperation with (A, A) is unlikely. This leads to low value estimates for action A and convergence to suboptimal policies. We provide additional visualisations and analysis in Appendix B.

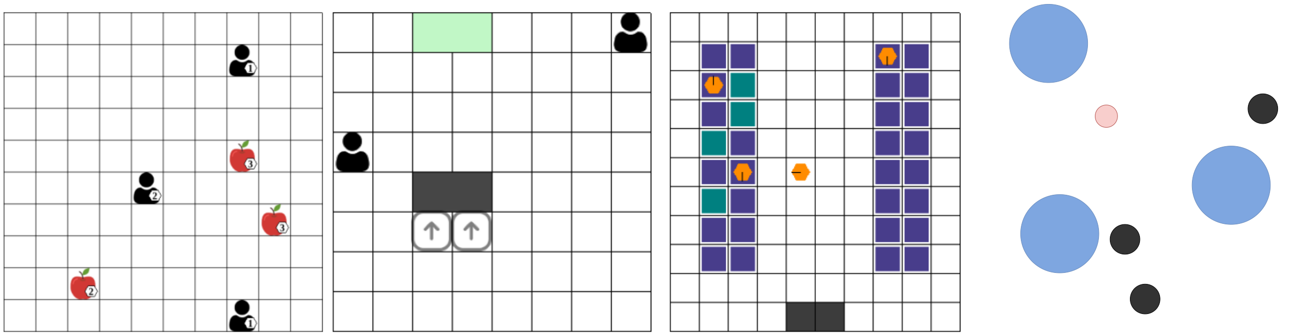
6 EXPERIMENTS

After illustrating the impact of UCB exploration with tabular ensemble value functions in a two-player matrix game, we evaluate EMAX and four deep value-based MARL baselines across 21 diverse multi-agent tasks in four environments.

6.1 Evaluation Details

We evaluate a total of seven deep MARL algorithms: Independent DQN (IDQN), VDN, and QMIX as well as their extensions with EMAX, which we will denote IDQN-EMAX, VDN-EMAX, and QMIX-EMAX, respectively, and MAVEN [18]. MAVEN extends QMIX for cooperative exploration by conditioning the individual value functions of agents and the mixing network on a latent variable sampled from a learned variational distribution. Following suggestions from Agarwal et al. [1], we report performance profiles and use the interquartile mean (IQM) and 95% confidence intervals computed over five runs in all tasks. For every algorithm and task, agents share network parameters and unless stated otherwise EMAX uses ensembles with $K = 5$ value functions. Details on hyperparameters are provided in Appendix A.2. We evaluate in 21 tasks across four multi-agent environments, visualised in Figure 4, focused on cooperation and exploration: eight level-based foraging (LBF) tasks [2, 26], four boulder-push (BPUSH) tasks [6], six multi-robot warehouse (RWARE) tasks [8, 26], and three multi-agent particle environment (MPE) tasks [17, 22].

Level-Based Foraging: The level-based foraging (LBF) environment [2, 26] contains diverse tasks in which agents and food



(a) Level-based foraging

(b) Boulder-push

(c) Multi-robot warehouse

(d) Multi-agent particle environment

Figure 4: Visualisations of four multi-agent environments.

are randomly scattered in a gridworld. Agents observe the location of themselves as well as all other agents and food in the gridworld, and are able to choose between discrete actions $\mathcal{A} = \{\text{do nothing}, \text{move up}, \text{move down}, \text{move left}, \text{move right}, \text{pick-up}\}$. Agents and food are assigned levels and agents can only pick-up food if the level of all agents standing next to the food and choosing the pick-up action together is greater or equal to the level of the food. Agents only receive rewards for successful collection of food. Episodes terminate after all food has been collected or after at most 50 timesteps. Each episode randomises the level and starting locations of agents and food. Tasks vary in the size of the gridworld, the number of agents and food, and the level assignment.

Boulder-Push: In the boulder-push environment (BPUSH) [6], agents need to navigate a gridworld to move a boulder to a target location. Agents observe the location of the boulder, all other agents, and the direction the boulder needs to be pushed in. The action space of all agents consists of the same discrete actions $\mathcal{A} = \{\text{move up}, \text{move down}, \text{move left}, \text{move right}\}$. Agents only receive rewards of 0.1 per agent for successfully pushing the boulder forward in its target direction, which requires cooperation of all agents, and a reward of 1 per agent for the boulder reaching its target location. Unsuccessful pushing of the boulder by some but not all agents leads to a penalty reward of -0.01. Episodes terminate after the boulder reached its target location or after at most 50 timesteps. BPUSH tasks considered in this work vary in the size of the gridworld and the number of agents varying between two and four.

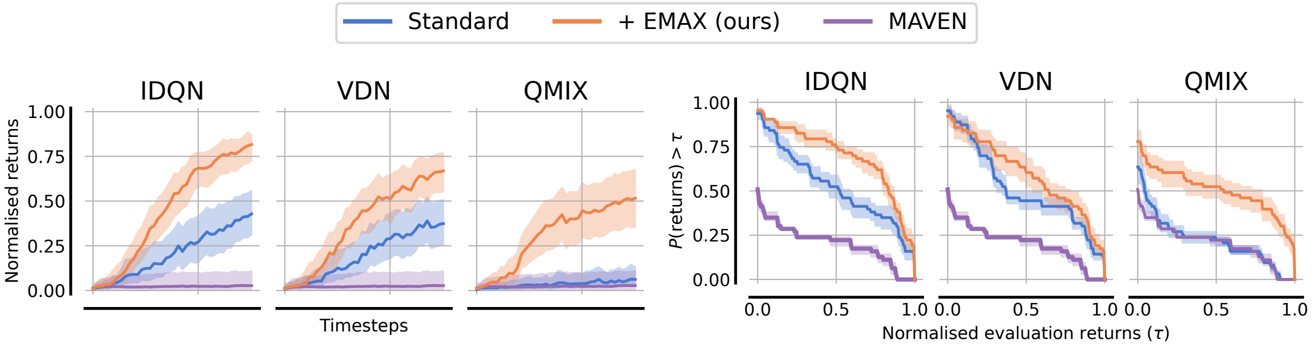
Multi-Robot Warehouse: The multi-robot warehouse environment (RWARE) [8, 26] represents gridworld warehouses with blocks of shelves. Agents need to navigate the warehouse and collect currently requested items. Agents only observe nearby agents and shelves immediately next to their location, and choose discrete actions $\mathcal{A} = \{\text{turn left}, \text{turn right}, \text{move forward}, \text{load/ unload shelf}\}$. Agents are only rewarded for successful deliveries of requested shelves, which require long sequences of actions, with a reward of 1, thus rewards are very sparse making RWARE tasks hard exploration problems. At each timestep, the total number of requested shelves is equal to the number of agents and once requested shelves, a currently unrequested shelf is uniformly at random sampled and added to the list of requested shelves. Episodes terminate after 500 timesteps. It is worth highlighting that no value-based algorithm achieved non-zero rewards in this environment within four million timesteps of training in prior evaluations [26].

Multi-Agent Particle Environment: In the multi-agent particle environment (MPE) [17, 22], agents navigate continuous two-dimensional, fully-observable environments. In all tasks, agents observe the relative position and velocity of all agents, as well as the relative positions of landmarks in the environment. Agents choose between five discrete actions consisting of doing nothing and movement in all four cardinal directions. We evaluate agents in three diverse tasks within MPE which all require cooperation between all agents with densely rewarded objectives. (1) Predator-prey in which three agents control predators in an environment with three landmarks, representing obstacles, and a faster, pre-trained[1] prey. The agents are rewarded with +10 for touching the prey agent. (2) Spread in which three agents need to cover three landmarks while avoiding collisions with each other. At each timestep, agents receive a negative reward corresponding to the minimum distance from each landmark to its closest agent as well as a small negative reward of -1 for agent collisions. (3) Adversary in which two agents are in an environment with an pre-trained adversary and two landmarks. At the beginning of each episode, one of the two landmarks is randomly determined as the goal landmark for the agents (agents observe this goal landmark but the adversary has no information about it). The agents receive rewards corresponding to the negative distance from the goal landmark to the closest agent and a reward corresponding to the distance of the adversary agent to the goal landmark.

6.2 Evaluation Results

Figure 5 visualises the learning curve and performance profile of evaluation returns of all algorithms across all 21 tasks. Across all tasks, EMAX improves final evaluation returns of IDQN, VDN, and QMIX, shown in Figure 5a, by 53%, 36%, and 498%, leading to higher final returns compared to their vanilla baselines in 19, 16, and 20 out of 21 tasks, respectively. These results mostly arise from improved sample efficiency for IDQN and VDN, and QMIX-EMAX

[1] Pre-trained agents are obtained from the EPyMARL codebase [26]. They were obtained by training all agents (including adversaries) with the MADDPG algorithm for 25,000 episodes.



(a) Learning curve of normalised evaluation returns

(b) Performance profile of final evaluation returns

Figure 5: (a) Evaluation returns throughout training and (b) performance profile [1] visualising the distribution of evaluation returns at the end of training of all algorithms, both aggregated across all 21 tasks. EMAX (orange) significantly improves the sample efficiency and final achieved returns of all algorithms. Lines and shading represent the interquartile mean and 95% confidence intervals of evaluation returns, respectively, aggregated over five runs for every task, for a total of 105 runs per algorithm. For each task, evaluation returns are normalised between the minimum (0) and maximum (1) achieved returns.

learning in several hard exploration tasks where QMIX fails to achieve any reward. The performance profile in Figure 5b visualises the the distribution of evaluation returns at the end of training across all algorithms and tasks. These profiles indicate that EMAX significantly improves the robustness of all algorithms, consistently achieving higher returns. We provide learning curves in all individual tasks, normalised evaluation returns for each environment, as well as a table with evaluation returns of final returns in any task in Appendix C.

In LBF, EMAX significantly improves the performance of QMIX whereas minor improvements can be seen for IDQN and VDN. Inspecting learning curves in individual tasks (see Appendix C) shows that QMIX fails to achieve any rewards in several LBF tasks with particularly sparse rewards. We hypothesise that QMIX, similarly MAVEN, suffer from the large dimensionality of the joint observation as input to the mixing network which is inefficient to train with the sparse learning signal of these tasks. The uncertainty-guided exploration of EMAX seems to alleviate these inefficiencies.

In BPUSH, a similar trend can be observed with, most notably, VDN-EMAX and QMIX-EMAX learn to solve a BPUSH task with four agents in which no baseline demonstrates any positive rewards (see Figure 12d). This task requires complex cooperation because four agents need to move in unison to successfully complete this task and any miscoordination leads to negative rewards.

In RWARE, consistent with prior work [26], independent learning value-based algorithms outperform centralised value decomposition methods due to highly sparse rewards. IDQN-EMAX outperforms all baselines across all six RWARE tasks, and IDQN-EMAX and VDN-EMAX both significantly improve upon their vanilla baselines in all RWARE tasks, achieving 330% and 252% higher final evaluation returns, respectively, whereas QMIX with and without EMAX as well as MAVEN fail to learn.

In contrast to other environments, MPE has continuous observations and dense rewards. In all three MPE tasks, we see improvements in sample efficiency and final performance for algorithms with EMAX compared to all the baselines.

Training stability: In MARL, the environment becomes non-stationary from the perspective of each agent as its perceived transitions and rewards are impacted by the constantly changing policies of other agents. EMAX computes target values as average value estimates across an ensemble which have been shown to reduce variance of target values [15]. To demonstrate the stabilising effect of these target values on the optimisation, we visualise the stability of gradients measured by the conditional value at risk (CVaR) of gradient norms, detrended over consecutive values, during the optimisation of IDQN, VDN, QMIX with and without EMAX

$$\mathrm{CVaR}(g') = \mathbb{E}\left[g' \mid g' \geq \mathrm{VaR}_{95\%}(g')\right] \tag{14}$$

$$g'_t = |\nabla_{t+1}| - |\nabla_t| \tag{15}$$

where the value at risk (VaR) corresponds to the value at the 95% quantile of all detrended gradient norm values. Figure 6 shows the average and standard error of these CVaR values across all 21 tasks. We observe that the target computation of EMAX indeed significantly reduces the CVaR of gradient norms for IDQN, VDN, and QMIX indicating more stable optimisation. We hypothesise that the difference for QMIX is less significant because it fails to learn in several tasks, leading to little training signal with low gradient variability independent of the target values.

Ensemble size: The computational cost of training an ensemble of models scales with the ensemble size K. Hence, we investigate the cost of training these ensemble models for varying K and pose the question of how many models are needed in the ensemble for EMAX to benefit from the improved exploration and stability. Table 2 shows the average time to train IDQN, VDN, QMIX, and their corresponding EMAX extensions with $K \in \{2, 5, 8\}$ for 10,000 timesteps in the LBF 10x10-3p-3f task. These times were averaged across ten runs. We can see that training an ensemble of $K = 5$

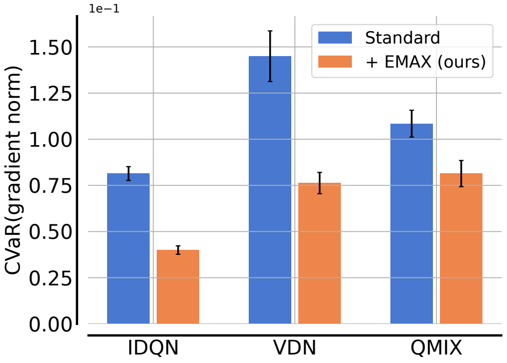


Figure 6: Average and standard error of the conditional value at risk (CVaR) of detrended consecutive gradient norms of all algorithms across all tasks. We detrend consecutively logged gradient norms for each task by computing the difference between them and compute the CVaR as the expected value of these gradient norm differences in the top 5% percentile before computing the average and standard error across all tasks. This metric corresponds to the short-term risk across time suggested by Chan et al. [5].

value functions, as applied in our evaluation, increases the training time by less than 100%. While this cost is significant, we believe that it is justified in cases where sample efficiency and stability are of importance as EMAX offers significant improvements in both of these. To investigate the question of how many models are needed in the ensemble, we evaluate all algorithms with varying K in the RWARE 11x10 task with four agents (Figure 7), in which EMAX led to substantial improvements for IDQN and VDN. It appears that the benefits of larger ensemble models saturate at $K = 5$. EMAX with $K = 8$ performs identical or worse for all algorithms, and the smaller ensemble $K = 2$ reaches lower returns for IDQN and VDN. These results suggest that a comparably small ensemble with $K = 5$, which approximately doubles wall clock training time, can significantly improve sample efficiency with EMAX. Additionally, we hypothesise that larger ensemble value functions may require more data to train, thus leading to diminishing benefits for ensembles of many value functions.

All gridsearches and evaluations for deep experiments were conducted on (1) desktop computers with two Nvidia RTX 2080 Ti GPUs, Intel i9-9900X @ 3.50GHz CPU, 62GB RAM, running Ubuntu 20.04, and (2) two server machines with four Nvidia V100 GPUs, Intel Xeon Platinum 8160 @ 2.10GHz CPU, 503GB RAM, running CentOS Linux 7 OS. The speedtest for varying ensemble sizes has been conducted on the desktop computer.

7 CONCLUSION

In this paper, we propose EMAX, a general framework to extend any value-based MARL algorithms using ensembles of value functions. EMAX leverages the disagreement of value estimates across the ensemble with a UCB policy to guide exploration towards parts of the environment which require coordination. Additionally, gradients during training are stabilised by computing target values as

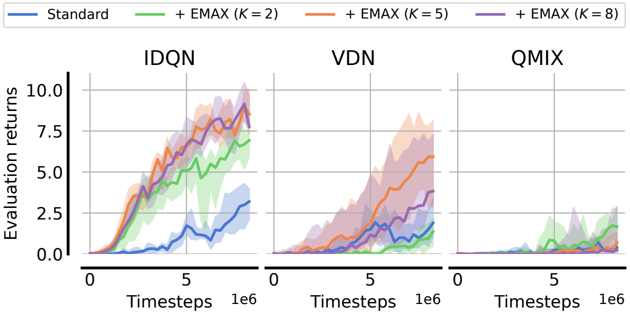


Figure 7: Evaluation returns for all vanilla and EMAX algorithms with varying ensemble sizes $K \in \{2, 5, 8\}$ in RWARE 11x10 4ag.

Table 2: Average time (in seconds) for vanilla and EMAX algorithms with varying ensemble sizes K to complete 10,000 timesteps of training in the LBF 10x10-3p-3f task. Relative increase to the training time of the baseline algorithm ($K = 1$) is given in parenthesis. Times are averaged across ten runs.

Algorithm	Baseline	$K = 2$	$K = 5$	$K = 8$
IDQN	16.80	21.29 (+27%)	33.04 (+97%)	48.06 (+186%)
VDN	16.92	21.56 (+27%)	33.25 (+97%)	48.16 (+185%)
QMIX	17.70	22.53 (+27%)	33.71 (+90%)	48.66 (+175%)

the average value estimate across the ensemble. Empirical results in 21 tasks across four environments demonstrate that EMAX significantly improves sample efficiency, final performance, and training stability for all three extended algorithms. Lastly, we discuss the computational cost introduced by EMAX and show that comparably small ensemble models are sufficient to achieve the demonstrated improvements.

EMAX is currently limited to value-based cooperative MARL algorithms. Firstly, future work should consider the extension of EMAX to multi-agent actor-critic algorithms such as MAPPO and IPPO, which have shown to be effective in cooperative MARL [38]. Ensembles of critics and policies could be trained for each agent, with similar target computation and UCB policies across actors being used to leverage the techniques proposed in this work. Secondly, future work could aim to reduce the computational cost of training ensembles of value functions. Prior work has explored the application of hypernetworks [11] and latent-conditioned models [33] to approximate ensembles using a single network. Similar techniques could help to significantly reduce the computational cost of EMAX, thereby making it more widely accessible. Lastly, ensembles of value functions can be used to efficiently explore in two-player zero-sum games [19, 27, 34].

REFERENCES

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. 2021. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems.*

[2] Stefano V. Albrecht and Subramanian Ramamoorthy. 2013. A Game-Theoretic Model and Best-Response Learning Method for Ad Hoc Coordination in Multiagent Systems. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

[3] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*.

[4] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3 (2002).

[5] Stephanie C.Y. Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. 2020. Measuring the reliability of reinforcement learning algorithms. In *International Conference on Learning Representations*.

[6] Filippos Christianos, Georgios Papoudakis, and Stefano V. Albrecht. 2022. Pareto Actor-Critic for Equilibrium Selection in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2209.14344* (2022).

[7] Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V. Albrecht. 2021. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*.

[8] Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*.

[9] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* (1998).

[10] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*.

[11] Vikranth Dwaracherla, Xiuyuan Lu, Morteza Ibrahimi, Ian Osband, Zheng Wen, and Benjamin Van Roy. 2020. Hypermodels for exploration. In *International Conference on Learning Representations*.

[12] Aleksandar Krnjaic, Jonathan D Thomas, Georgios Papoudakis, Lukas Schäfer, Peter Börsting, and Stefano V. Albrecht. 2022. Scalable Multi-Agent Reinforcement Learning for Warehouse Logistics with Robotic and Human Co-Workers. *arXiv preprint arXiv:2212.11498* (2022).

[13] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2021. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*.

[14] Xihan Li, Jia Zhang, Jiang Bian, Yunhai Tong, and Tie-Yan Liu. 2019. A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. In *International Conference on Autonomous Agents and Multi-Agent Systems*.

[15] Litian Liang, Yaosheng Xu, Stephen McAleer, Dailin Hu, Alexander Ihler, Pieter Abbeel, and Roy Fox. 2022. Reducing Variance in Temporal-Difference Value Estimation via Ensemble of Deep Networks. In *International Conference on Machine Learning*.

[16] I.-J. Liu, U. Jain, R. A. Yeh, and A. G. Schwing. 2021. Cooperative Exploration for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*.

[17] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems* (2017).

[18] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems* (2019).

[19] Stephen McAleer, Gabriele Farina, Marc Lanctot, and Tuomas Sandholm. 2023. ESCHER: Eschewing Importance Sampling in Games by Computing a History Value Function to Estimate Regret. *International Conference on Learning Representations* (2023).

[20] David Henry Mguni, Taher Jafferjee, Jianhong Wang, Nicolas Perez-Nieves, Oliver Slumbers, Feifei Tong, Yang Li, Jiangcheng Zhu, Yaodong Yang, and Jun Wang. 2022. LIGS: Learnable Intrinsic-Reward Generation Selection for Multi-Agent Learning. In *International Conference on Learning Representations*.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015).

[22] Igor Mordatch and Pieter Abbeel. 2018. Emergence of grounded compositional language in multi-agent populations. In *AAAI Conference on Artificial Intelligence*.

[23] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*.

[24] Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*.

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

[26] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*.

[27] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. 2022. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science* 378, 6623 (2022).

[28] David V. Pynadath and Milind Tambe. 2002. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research* 16 (2002).

[29] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* 21, 1 (2020).

[30] Heechang Ryu, Hayong Shin, and Jinkyoo Park. 2022. REMAX: Relational Representation for Multi-Agent Exploration. In *International Conference on Autonomous Agents and Multiagent Systems*.

[31] Lukas Schäfer, Filippos Christianos, Josiah P Hanna, and Stefano V. Albrecht. 2022. Decoupled Reinforcement Learning to Stabilise Intrinsically-Motivated Exploration.. In *International Conference on Autonomous Agents and Multiagent Systems*.

[32] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[33] Macheng Shen and Jonathan P. How. 2023. Implicit Ensemble Training for Efficient and Robust Multiagent Reinforcement Learning. *Transactions on Machine Learning Research* (2023).

[34] Samuel Sokota, Ryan D'Orazio, J Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer. 2022. A unified approach to reinforcement learning, quantal response equilibria, and two-player zero-sum games. *arXiv preprint arXiv:2206.05825* (2022).

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

[36] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning*.

[37] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. 2020. Influence-based multi-agent exploration. In *International Conference on Learning Representations*.

[38] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative, multi-agent games. In *Advances in Neural Information Processing Systems, Track on Datasets and Benchmarks*.

[39] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, IMAN FADAKAR, Zheng Chen, et al. 2021. Smarts: An open-source scalable multi-agent rl training school for autonomous driving. In *Conference on Robot Learning*.

[40] Yichi Zhou, Jialian Li, and Jun Zhu. 2020. Posterior sampling for multi-agent reinforcement learning: solving extensive games with imperfect information. In *International Conference on Learning Representations*.

A HYPERPARAMETER SETTINGS

A.1 Tabular Algorithms

Initialisation of tabular value functions. Tabular value functions are often initialised with zero values. In order to represent initial uncertainty about value estimates and remain close to the deep setting from Section 4, we randomly initialise value functions with a zero-mean Gaussian distribution. We observe that such initialised values also improve the performance of the baselines and hence initialise the tabular value functions of all algorithms in this way.

Table 3: Hyperparameters for tabular IQL with and without ensemble value functions in the climbing matrix game. We conducted a gridsearch over all listed hyperparameter values using ten random seeds with the bold entries corresponding to the best identified configuration.

Algorithm	Hyperparameter	Values
General	γ	0.99
	Greedy evaluation	True
IQL ϵ-greedy	Learning rate	**0.01**, 0.03, 0.1
	Value initialisation	0, $\mathcal{N}(0, 1)$, $\mathcal{N}(0, 5)$, $\mathcal{N}(\mathbf{0}, \mathbf{10})$
	Decay over steps	**250**, 500, 1000
	Final ϵ	0.05, **0.0**
IQL UCB	Learning rate	**0.01**, 0.03, 0.1
	Value initialisation	0, $\mathcal{N}(0, 1)$, $\mathcal{N}(\mathbf{0}, \mathbf{5})$, $\mathcal{N}(0, 10)$
	UCB uncertainty coefficient β	0.1, **0.3**, 1, 3, 10
Ensemble IQL ϵ-greedy	Ensemble size K	**10**, 50
	Learning rate	**0.01**, 0.03, 0.1
	Value initialisation	0, $\mathcal{N}(\mathbf{0}, \mathbf{1})$, $\mathcal{N}(0, 5)$, $\mathcal{N}(0, 10)$
	Bernoulli p	0.9
	Decay over steps	**250**, 500, 1000
	Final ϵ	0.05, **0.0**
Ensemble IQL UCB	Ensemble size K	10, **50**
	Learning rate	**0.01**, 0.03, 0.1
	Value initialisation	0, $\mathcal{N}(0, 1)$, $\mathcal{N}(\mathbf{0}, \mathbf{5})$, $\mathcal{N}(0, 10)$
	Bernoulli p	0.9
	UCB uncertainty coefficient β	0.1, 0.3, 1, **3**, 10

A.2 Deep Algorithms

For IDQN, VDN, QMIX and extensions with EMAX, we conduct a gridsearch to identify best hyperparameters in one selected task within each environment by evaluating each algorithm configuration for three runs and selecting the hyperparameter configuration which led to highest average evaluation returns throughout training. Our implementation of IDQN, VDN, QMIX, and EMAX are based on the EPyMARL codebase[2]. For the baseline of MAVEN, we migrated the provided codebase from the authors[3] into EPyMARL to support all environments. For MAVEN, we use the hyperparameters identified for QMIX for each environment with the MAVEN-specific hyperparameters provided by the authors.

Table 4: Hyperparameters for IDQN, VDN, QMIX and extensions with EMAX in LBF. The gridsearch was conducted in Foraging-10x10-4p-3f-coop for 4M time steps, and the bold entries corresponding to the best identified configuration.

Algorithm	Hyperparameter	Value
Shared	γ	0.99
	Activation function	ReLU
	Parameter sharing	True
	Optimiser	Adam
	Maximum gradient norm	5
	Minimum ϵ	0.05
	Evaluation ϵ	0.05
	Learning rate	e^{-4}
	Target update frequency	200
	Replay buffer capacity (episodes)	5,000
	Batch size (episodes)	32
QMIX	Mixing embedding size	32
	Hypernetwork embedding size	64
IDQN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
VDN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	50,000, **200,000**
QMIX	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	50,000, **200,000**
IDQN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, 0.3, **1**
VDN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	**0.1**, 0.3, 1
QMIX-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1

[2] Available at https://github.com/uoe-agents/epymarl.

[3] Available at https://github.com/AnujMahajanOxf/MAVEN.

Table 5: Hyperparameters for IDQN, VDN, QMIX and extensions with EMAX in BPUSH. The gridsearch was conducted in BPUSH 12×12 2ag for 7.5M time steps, and the bold entries corresponding to the best identified configuration.

Algorithm	Hyperparameter	Value
Shared	γ	0.99
	Activation function	ReLU
	Parameter sharing	True
	Optimiser	Adam
	Maximum gradient norm	5
	Minimum ϵ	0.05
	Evaluation ϵ	0.05
	Learning rate	e^{-4}
	Target update frequency	200
	Replay buffer capacity (episodes)	5,000
	Batch size (episodes)	32
QMIX	Mixing embedding size	32
	Hypernetwork embedding size	64
IDQN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
VDN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	50,000, **200,000**
QMIX	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	50,000, **200,000**
IDQN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, 0.3, **1**
VDN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	**0.1**, 0.3, 1
QMIX-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1

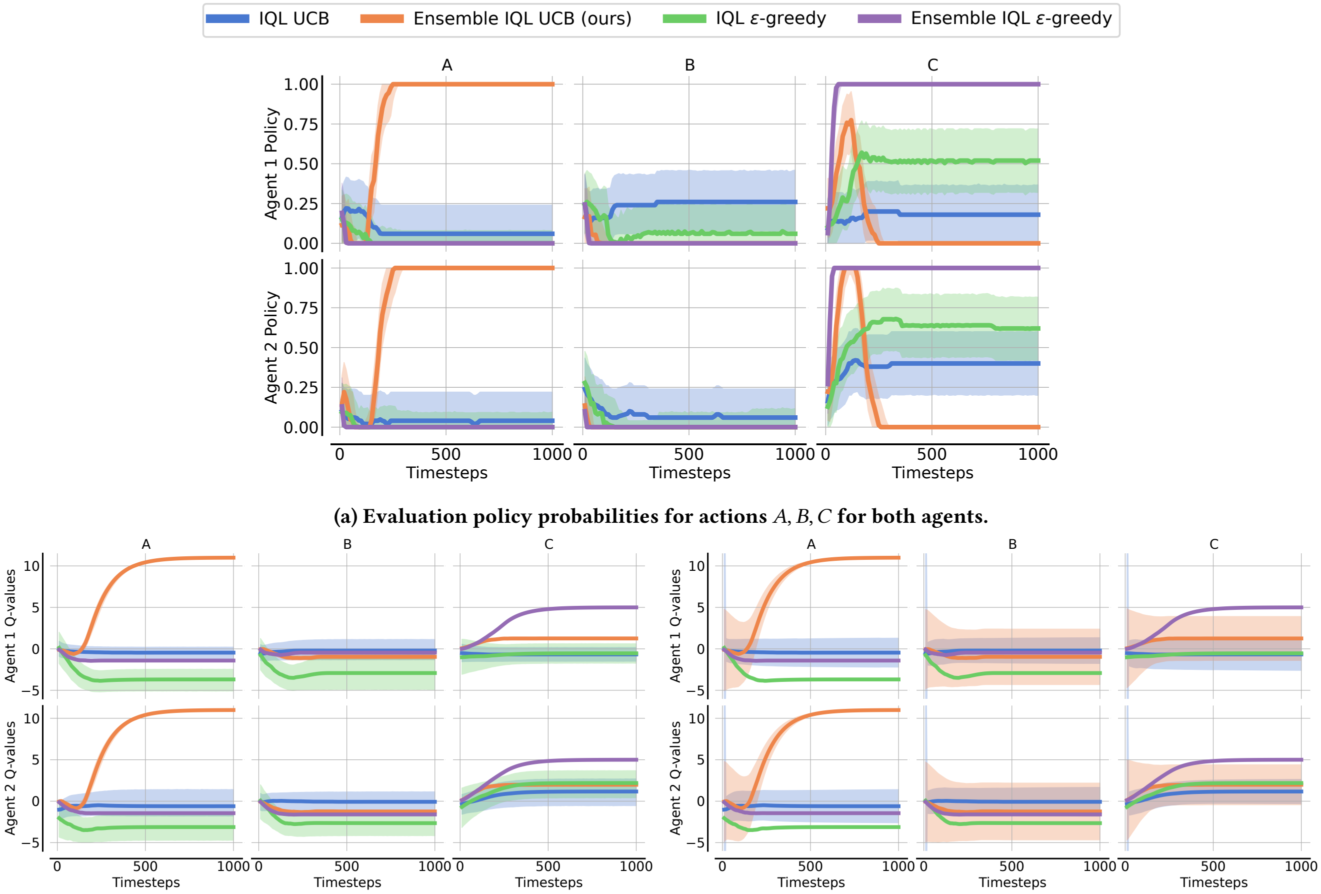
Table 6: Hyperparameters for IDQN, VDN, QMIX and extensions with EMAX in RWARE. The gridsearch was conducted in RWARE 11×10 4ag for 5M time steps, and the bold entries corresponding to the best identified configuration.

Algorithm	Hyperparameter	Value
Shared	γ	0.99
	Activation function	ReLU
	Parameter sharing	True
	Optimiser	Adam
	Maximum gradient norm	5
	Minimum ϵ	0.05
	Evaluation ϵ	0.05
	Learning rate	e^{-4}
	Target update frequency	200
	Replay buffer capacity (episodes)	5,000
	Batch size (episodes)	32
QMIX	Mixing embedding size	32
	Hypernetwork embedding size	64
IDQN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
VDN	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
QMIX	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
IDQN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1
VDN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1
QMIX-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1

Table 7: Hyperparameters for IDQN, VDN, QMIX and extensions with EMAX in MPE. The gridsearch was conducted in Spread for 1M time steps, and the bold entries corresponding to the best identified configuration.

Algorithm	Hyperparameter	Value
Shared	γ	0.99
	Activation function	ReLU
	Parameter sharing	True
	Optimiser	Adam
	Maximum gradient norm	5
	Minimum ϵ	0.05
	Evaluation ϵ	0.05
	Learning rate	e^{-4}
	Target update frequency	200
	Replay buffer capacity (episodes)	5,000
	Batch size (episodes)	32
QMIX	Mixing embedding size	32
	Hypernetwork embedding size	64
IDQN	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
VDN	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
QMIX	Network architecture	**FC**, FC + GRU
	Network size	64, **128**
	Reward standardisation	False, **True**
	ϵ decay steps	**50,000**, 200,000
IDQN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, 0.3, **1**
VDN-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	**0.1**, 0.3, 1
QMIX-EMAX	Network architecture	FC, **FC + GRU**
	Network size	64, **128**
	Reward standardisation	False, **True**
	UCB uncertainty coefficient β	0.1, **0.3**, 1

B MORE EVALUATION ANALYSIS ON TABULAR EXPERIMENTS



(a) Evaluation policy probabilities for actions A, B, C for both agents.

(b) Q-values with CIs

(c) Q-values with UCB uncertainty

Figure 8: Visualisation of the convergence of evaluation policies and Q-values for both agents trained with IQL with or without ensemble value functions and UCB or ϵ-greedy exploration. Figure (a) visualises the convergence of agents to cooperation policies during evaluation given by the IQM and 95% confidence intervals over their evaluation policy. Our approach stably converges to the optimal policy (A, A) whereas both non-ensemble baselines, and ensemble IQL with ϵ-greedy exploration converge to suboptimal policies. Plots (b) and (c) show the IQM across Q-value estimates as lines, with ensemble value functions computing Q-value estimates as the average value across the ensemble. The shading for (b) shows the 95% confidence intervals across the Q-values across all 100 runs. The shading for (c) shows the IQM of the uncertainty computed for both algorithms with UCB exploration indicating the decay of uncertainty for actions which are continuously explored.

C INDIVIDUAL TASK RESULTS FOR DEEP EVALUATION

Table 8: Average returns and standard deviation over five seeds at the end of training for all algorithms in all tasks. Highest average returns and standard deviation are highlighted in bold and entries within a single standard deviation are marked with an asterisk.

	Tasks \Algs.	IDQN	IDQN-EMAX	VDN	VDN-EMAX	QMIX	QMIX-EMAX	MAVEN
LBF	10x10-4p-1f-coop	**1.00 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.00**	0.98 ± 0.04	0.17 ± 0.23	0.80 ± 0.40	0.00 ± 0.00
	10x10-4p-2f-coop	0.87 ± 0.10	**0.98 ± 0.02**	0.95 ± 0.07	0.71 ± 0.19	0.02 ± 0.02	0.78 ± 0.39	0.00 ± 0.00
	10x10-4p-3f-coop	0.59 ± 0.30	**0.89 ± 0.13**	0.46 ± 0.25	0.71 ± 0.30	0.01 ± 0.01	0.79 ± 0.39 *	0.00 ± 0.00
	10x10-4p-4f-coop	0.30 ± 0.20	0.42 ± 0.21	0.25 ± 0.03	**0.76 ± 0.21**	0.02 ± 0.02	0.57 ± 0.47 *	0.00 ± 0.00
	10x10-3p-5f	0.68 ± 0.23	0.96 ± 0.03	0.68 ± 0.15	0.92 ± 0.07	0.12 ± 0.02	**1.00 ± 0.00**	0.02 ± 0.01
	15x15-8p-1f-coop	**1.00 ± 0.00**	0.24 ± 0.38	**1.00 ± 0.00**	0.64 ± 0.36	0.00 ± 0.00	0.80 ± 0.40	0.00 ± 0.00
	5x5-2p-1f-coop-pen	0.62 ± 0.44	**1.00 ± 0.00**	0.32 ± 0.45	**1.00 ± 0.00**	−0.03 ± 0.00	0.38 ± 0.48	0.00 ± 0.00
	5x5-2p-2f-coop-pen	0.45 ± 0.42 *	**0.71 ± 0.36**	−0.02 ± 0.00	0.14 ± 0.18	−0.04 ± 0.02	−0.02 ± 0.00	0.24 ± 0.15
BPUSH	8x8 2ag	2.30 ± 0.50	**2.73 ± 0.10**	2.69 ± 0.07 *	2.70 ± 0.05 *	1.91 ± 0.70	2.66 ± 0.13 *	1.84 ± 1.06
	12x12 2ag	0.87 ± 0.64 *	**1.59 ± 0.89**	1.32 ± 0.75 *	0.83 ± 0.57 *	0.35 ± 0.25	**1.59 ± 0.82**	0.42 ± 0.34
	20x20 2ag	0.00 ± 0.00	0.33 ± 0.67 *	**0.36 ± 0.30**	0.11 ± 0.13 *	0.07 ± 0.13 *	0.33 ± 0.28 *	0.09 ± 0.10 *
	5x5 4ag	0.07 ± 0.17	0.85 ± 1.28	0.09 ± 0.18	1.79 ± 1.64 *	−0.01 ± 0.00	**2.26 ± 1.31**	−0.01 ± 0.00
RWARE	11x10 2ag	0.70 ± 0.48	**3.04 ± 0.67**	0.30 ± 0.25	2.28 ± 1.23	0.04 ± 0.08	0.02 ± 0.04	0.00 ± 0.00
	11x10 4ag	4.22 ± 1.01	**8.78 ± 1.11**	2.44 ± 0.75	7.14 ± 2.01	0.36 ± 0.30	1.64 ± 1.12	0.00 ± 0.00
	20x10 2ag	0.26 ± 0.24	**1.64 ± 0.82**	0.34 ± 0.38	0.78 ± 0.91	0.04 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
	20x10 4ag	2.60 ± 1.01	**4.36 ± 0.34**	1.24 ± 0.67	2.66 ± 1.08	0.08 ± 0.11	0.52 ± 0.26	0.00 ± 0.00
	20x16 4ag	1.16 ± 0.67	**3.78 ± 0.64**	0.90 ± 0.54	0.58 ± 0.69	0.06 ± 0.08	0.06 ± 0.04	0.00 ± 0.00
	29x16 4ag	0.54 ± 0.58	**2.37 ± 1.67**	0.08 ± 0.07	0.02 ± 0.04	0.02 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
MPE	Spread	−144.95 ± 3.88	−135.21 ± 8.99	−144.69 ± 2.49	**−118.28 ± 1.47**	−154.08 ± 3.75	−122.41 ± 4.67	−208.92 ± 7.96
	Predator-prey	10.67 ± 3.39	20.00 ± 10.19*	13.67 ± 2.05	17.33 ± 7.13 *	9.67 ± 5.79	**25.33 ± 7.93**	1.02 ± 0.56
	Adversary	9.37 ± 1.17	10.35 ± 0.74	9.76 ± 1.07	10.48 ± 0.20	7.61 ± 0.91	**12.18 ± 0.86**	7.17 ± 0.76

C.1 Level-Based Foraging

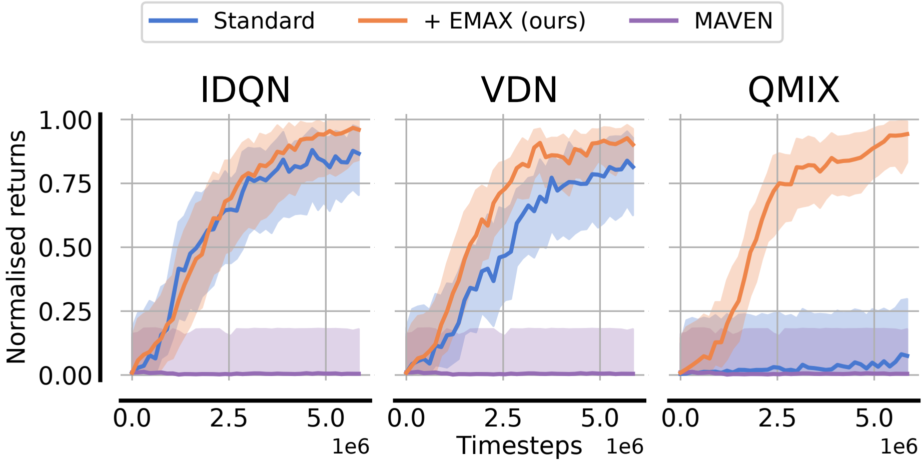
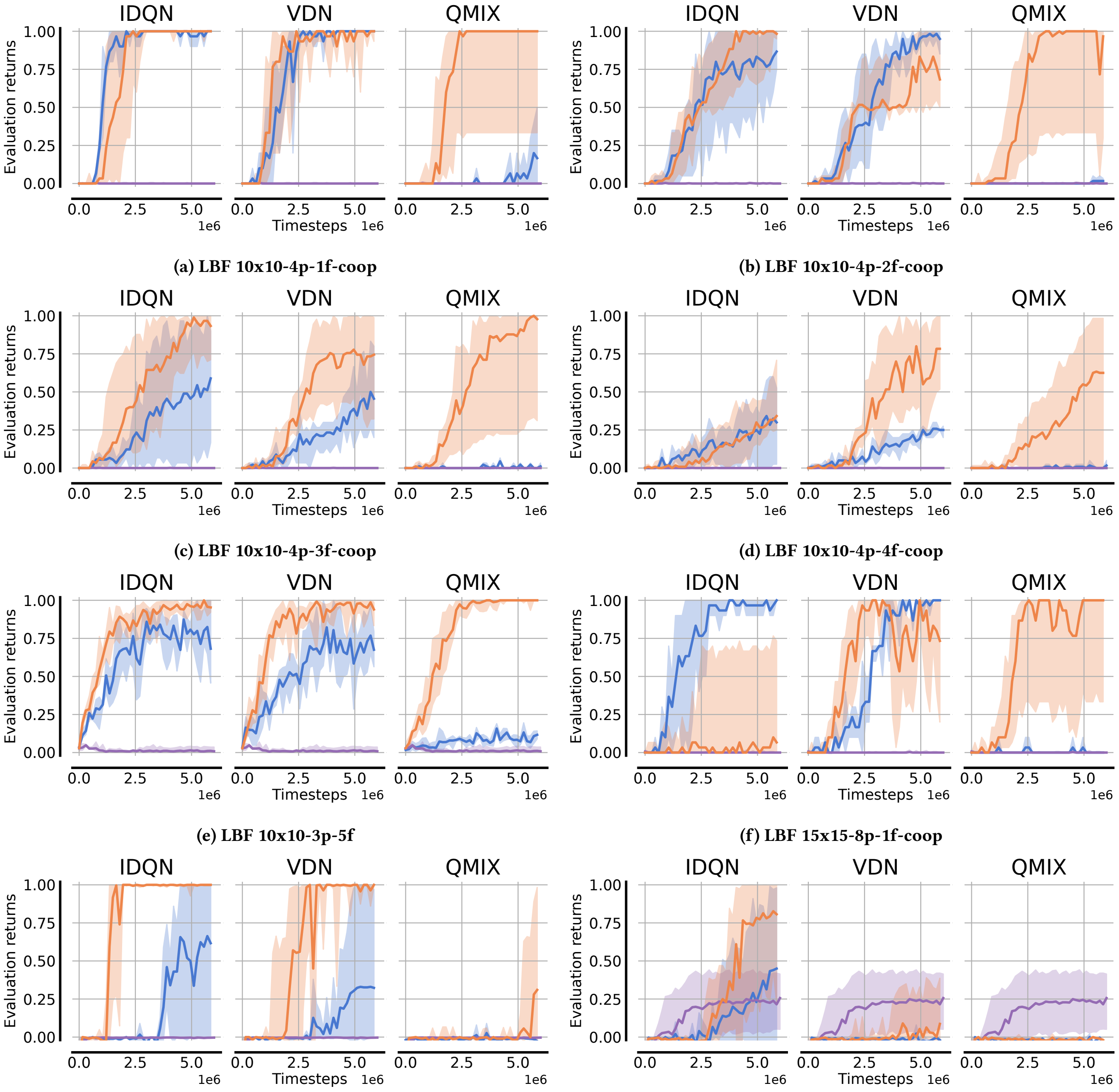


Figure 9: Interquartile mean and 95% confidence intervals of normalised evaluation returns for all algorithms in LBF. Evaluation returns are normalised between 0 and 1 for each task with the minimum and maximum achieved evaluation return of any algorithm before computing the interquartile mean and confidence intervals over all tasks and runs.



(a) LBF 10x10-4p-1f-coop

(b) LBF 10x10-4p-2f-coop

(c) LBF 10x10-4p-3f-coop

(d) LBF 10x10-4p-4f-coop

(e) LBF 10x10-3p-5f

(f) LBF 15x15-8p-1f-coop

(g) LBF 5x5-2p-1f-coop-pen

(h) LBF 5x5-2p-2f-coop-pen

Figure 10: Interquartile mean and 95% confidence intervals of evaluation returns for all algorithms in LBF tasks.

C.2 Boulder-Push

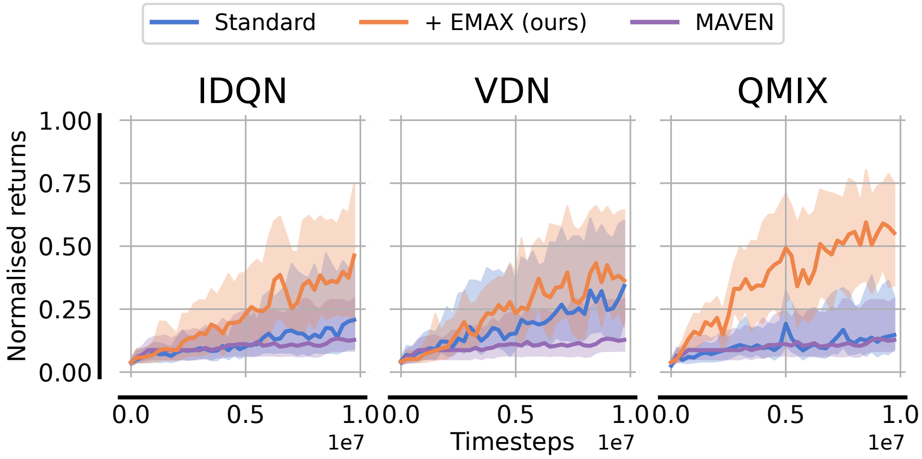
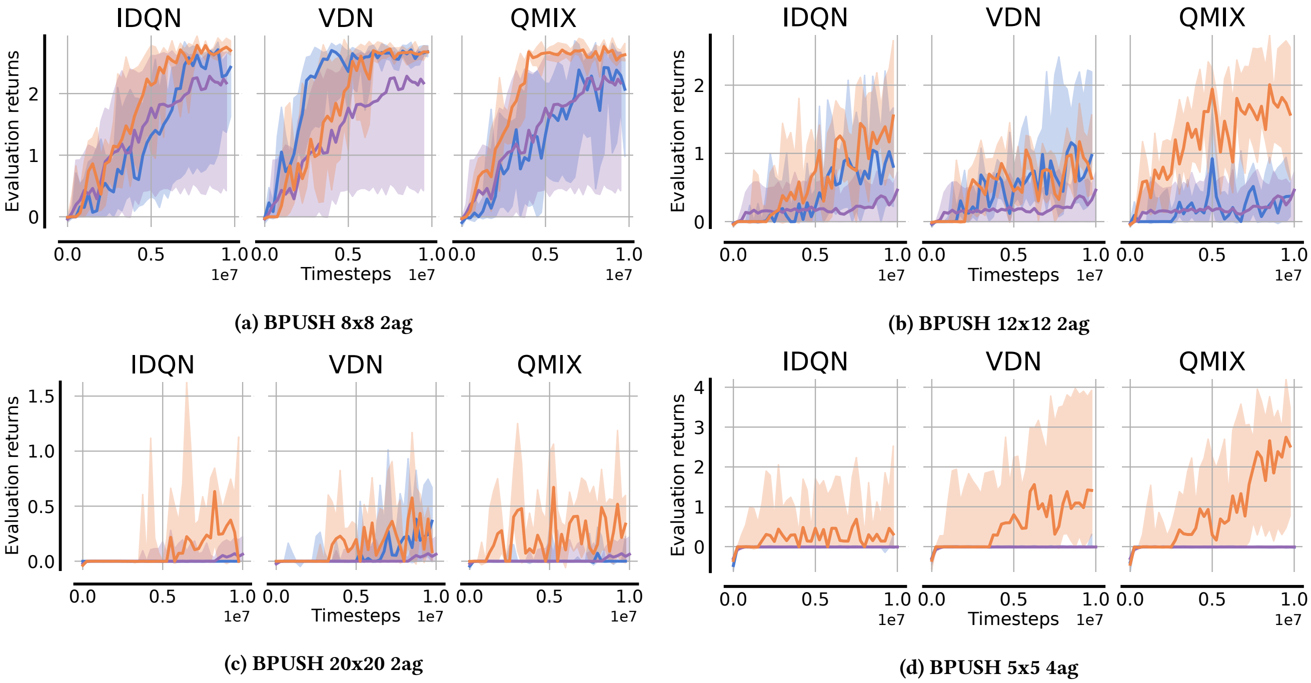


Figure 11: Interquartile mean and 95% confidence intervals of normalised evaluation returns for all algorithms in BPUSH. Evaluation returns are normalised between 0 and 1 for each task with the minimum and maximum achieved evaluation return of any algorithm before computing the interquartile mean and confidence intervals over all tasks and runs.



(a) BPUSH 8x8 2ag

(b) BPUSH 12x12 2ag

(c) BPUSH 20x20 2ag

(d) BPUSH 5x5 4ag

Figure 12: Interquartile mean and 95% confidence intervals of evaluation returns for all algorithms in BPUSH tasks.

C.3 Multi-Robot Warehouse

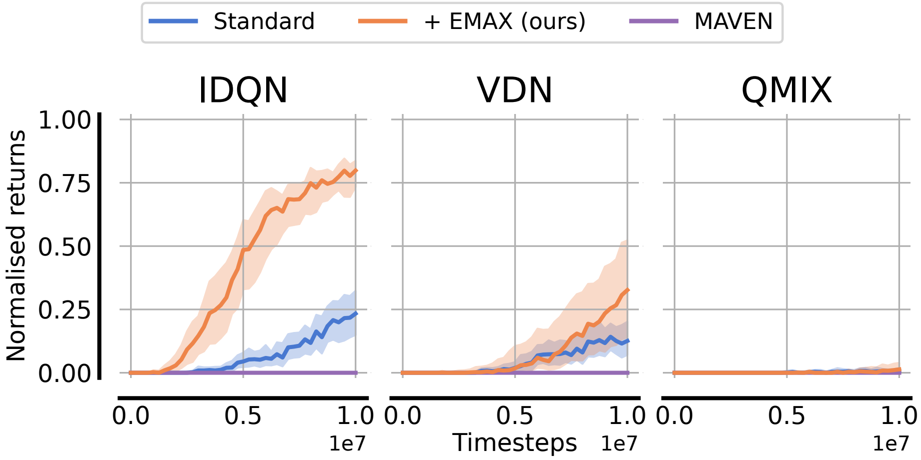
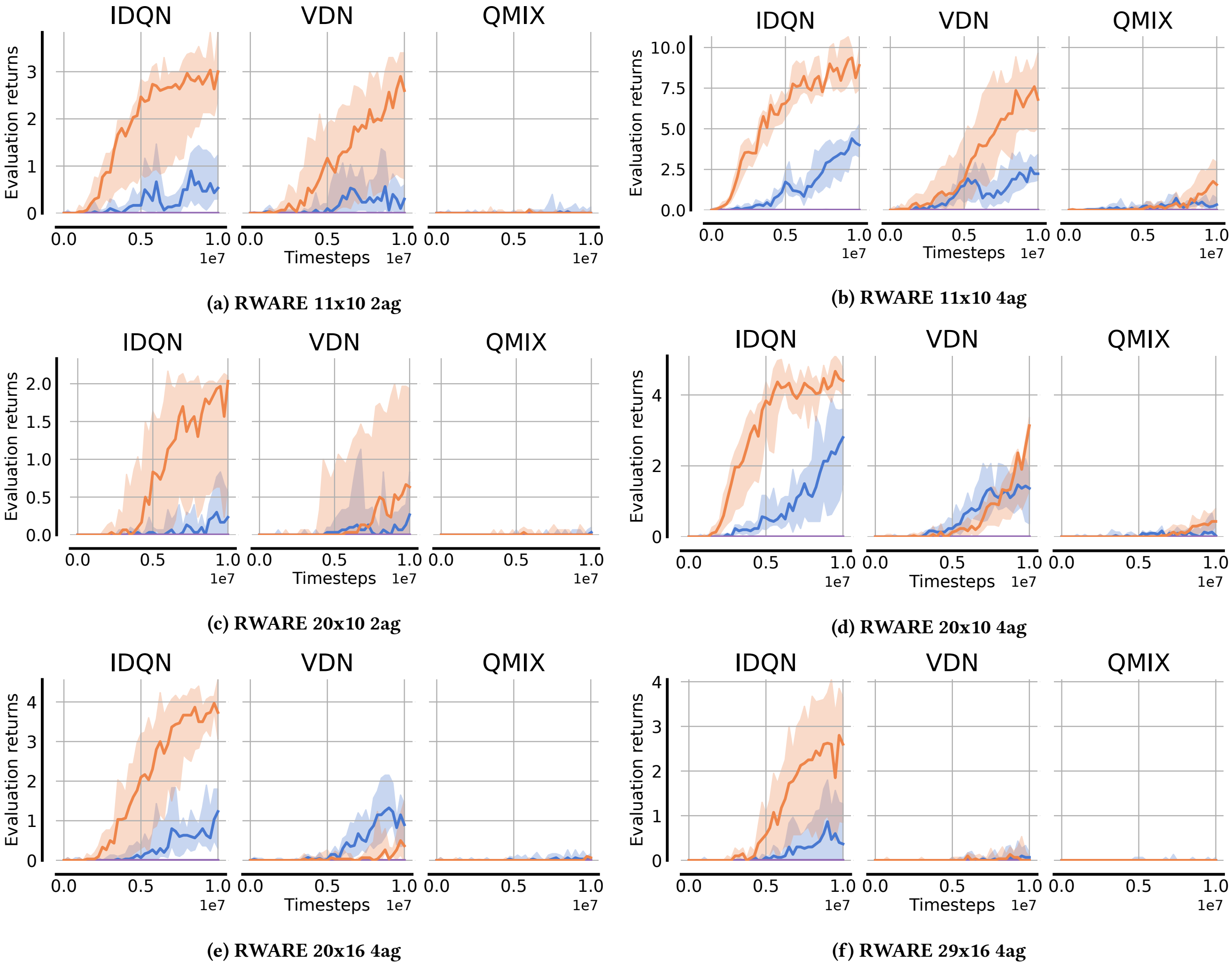


Figure 13: Interquartile mean and 95% confidence intervals of normalised evaluation returns for all algorithms in RWARE. Evaluation returns are normalised between 0 and 1 for each task with the minimum and maximum achieved evaluation return of any algorithm before computing the interquartile mean and confidence intervals over all tasks and runs.



(a) RWARE 11x10 2ag

(b) RWARE 11x10 4ag

(c) RWARE 20x10 2ag

(d) RWARE 20x10 4ag

(e) RWARE 20x16 4ag

(f) RWARE 29x16 4ag

Figure 14: Interquartile mean and 95% confidence intervals of evaluation returns for all algorithms in RWARE tasks.

C.4 Multi-Agent Particle Environment

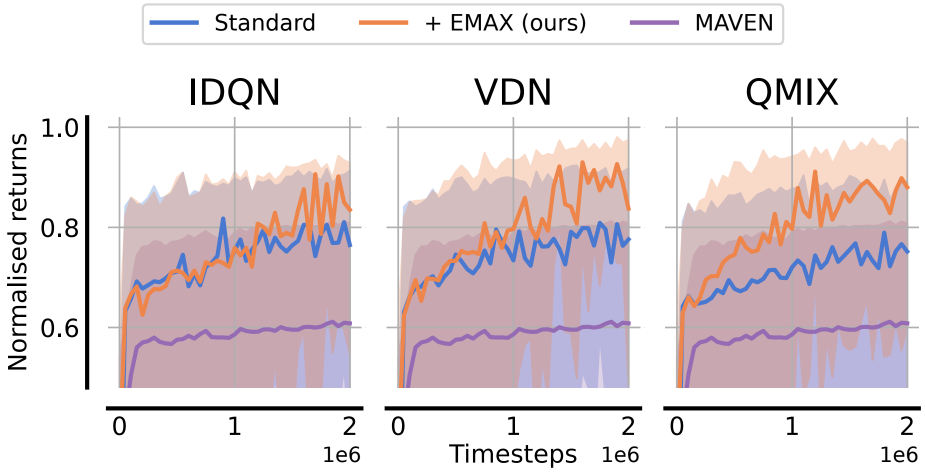
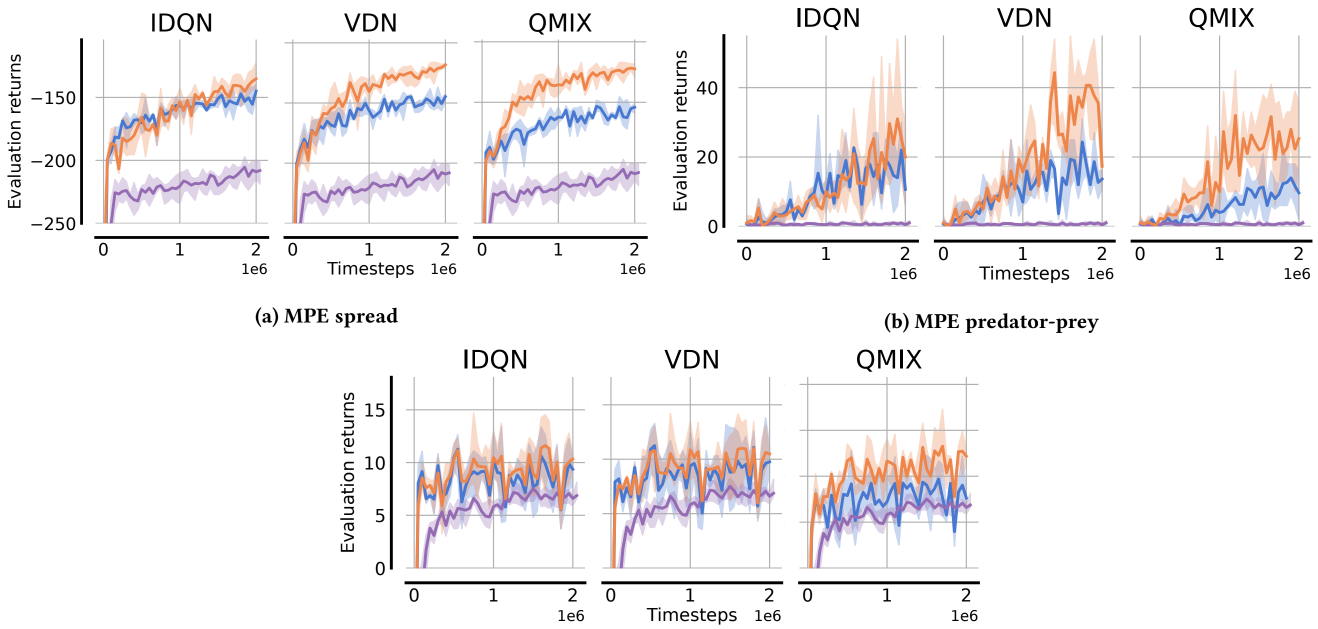


Figure 15: Interquartile mean and 95% confidence intervals of normalised evaluation returns for all algorithms in MPE. Evaluation returns are normalised between 0 and 1 for each task with the minimum and maximum achieved evaluation return of any algorithm before computing the interquartile mean and confidence intervals over all tasks and runs.



(a) MPE spread

(b) MPE predator-prey

(c) MPE adversary

Figure 16: Interquartile mean and 95% confidence intervals of evaluation returns for all algorithms in MPE tasks.

D EVALUATION POLICY ROBUSTNESS

During evaluation, EMAX selects actions by a majority vote across all policies in the ensemble (equation (12)), so any individual policy taking sub-optimal actions does not impact the executed policy as long as the majority of policies agree on the optimal action. Figure 17 shows the evaluation returns of IDQN, VDN, and QMIX with and without EMAX in LBF 10x10-4p-4f-coop. For EMAX, we show the evaluation policy using majority voting (ours) as well as an ablation following the greedy policy with respect to any of the individual value functions within the ensemble (single model). We highlight that no further agents were trained, but we directly extract the individual value functions within the ensemble and evaluate them, so the only difference in the EMAX single policy ablation and ours is the followed policy. This experiment indicates the improved robustness of our majority voting to select actions leading to higher evaluation returns. This is particularly valuable in tasks where multiple agents need to cooperate to pick-up food.

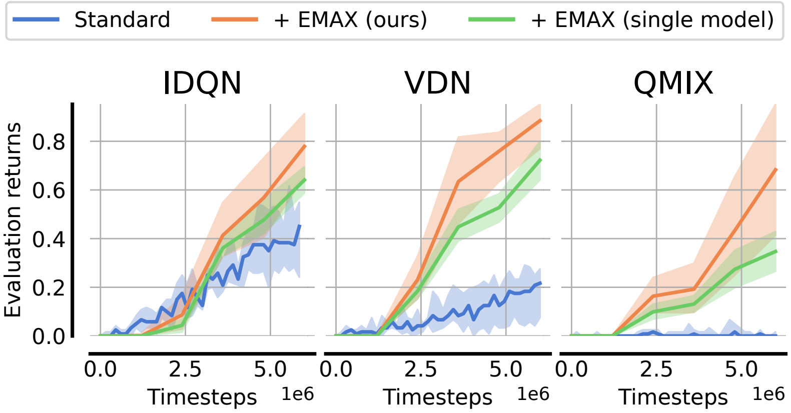


Figure 17: Interquartile mean and 95% confidence intervals of evaluation returns for IDQN, VDN, and QMIX with and without EMAX and an ablation of the evaluation policy in LBF 10x10-4p-4f-coop. For the single model ablation, the agents follow the greedy policy with respect to a single value function within their model instead of computing a majority vote across greedy policies.