Comparative Evaluation of Multi-Agent Deep Reinforcement Learning Algorithms

Georgios Papoudakis * **Filippos Christianos** * **Lukas Schäfer** **Stefano V. Albrecht**

School of Informatics
University of Edinburgh
{g.papoudakis, f.christianos, l.schaefer, s.albrecht}@ed.ac.uk

Abstract

Multi-agent deep reinforcement learning (MARL) suffers from a lack of commonly-used evaluation tasks and criteria, making comparisons between approaches difficult. In this work, we evaluate and compare three different classes of MARL algorithms (independent learners, centralised training with decentralised execution, and value decomposition) in a diverse range of multi-agent learning tasks. Our results show that (1) algorithm performance depends strongly on environment properties and no algorithm learns efficiently across all learning tasks; (2) independent learners often achieve equal or better performance than more complex algorithms; (3) tested algorithms struggle to solve multi-agent tasks with sparse rewards. We report detailed empirical data, including a reliability analysis, and provide insights into the limitations of the tested algorithms.

1 Introduction

Multi-agent reinforcement learning (MARL) addresses learning problems consisting of multiple, concurrently learning agents, that require independent (locally-executable) decision policies. Recent years have seen a plethora of new MARL algorithms which integrate deep learning techniques to aid learning (e.g. see survey [26]). However, comparison of MARL algorithms is difficult due to a lack of commonly-used test environments and evaluation protocols and metrics. While several comparative studies exist for single-agent RL [9, 14, 37], we are unaware of such comparative studies for recent MARL algorithms. The only exception is the study of Albrecht and Ramamoorthy [2] which however focused on classic (non-deep) MARL algorithms and simple matrix games. Such comparisons are crucial in order to understand the relative strengths and limitations of algorithms, which may guide practical considerations and future research.

We contribute a comprehensive empirical comparison of seven MARL algorithms in a diverse set of multi-agent tasks. We focus on three classes of decentralised execution algorithms: independent learning, which applies single-agent RL algorithms for each agent without special modifications to exploit multi-agent structure [35]; centralised training with decentralised execution (CTDE) [12, 20]; and algorithms based on value-decomposition [28, 33]. These algorithm classes are frequently used in the literature either as baselines or building blocks for more complex algorithms [8, 11, 13, 15, 17, 27, 29, 32]. The algorithms are compared across four different multi-agent environments [1] based on which we define a total of 17 different learning tasks. We report standard average training and final returns, as well as several recently-proposed metrics for evaluating robustness and variability [4].

*Authors contributed equally

[1] We provide open-source implementations of two newly developed environments here:
www.github.com/uoe-agents/lb-foraging,
www.github.com/uoe-agents/robotic-warehouse

Our three key observations are: (1) The performance of each algorithm strongly depends on the properties of the environment, and none of the evaluated algorithms manages to learn efficiently across all tasks; (2) The relatively simple class of independent learners often achieves equal or better performance than more complex algorithms across learning tasks; (3) The evaluated algorithms generally struggle to solve environments with sparse rewards, suggesting an open research problem: exploiting structure in multi-agent interaction to achieve efficient exploration under sparse rewards.

2 Algorithms

We consider three classes of MARL algorithms from which we compare seven different algorithms. Table 1 lists algorithms and their properties.

2.1 Independent Learning

A straightforward approach to MARL is independent decentralised training. In these algorithms, each agent is learning independently and perceives the other agents as part of the environment.

IQL: In Independent Q-Learning (IQL) [35], each agent has a decentralised state-action value function that is conditioned only on the local history of observations and actions of each agent. Each agent receives its local history of observations and updates the parameters of the Q-value network [23] using the standard Q-learning optimisation [38].

IA2C: Independent synchronous Advantage Actor-Critic (IA2C) is a variant of the commonly-used A2C algorithm [7, 24] for decentralised training in multi-agent systems. Each agent has its own actor to approximate the policy and critic network to approximate the value-function. Both actor and critic are trained, conditioned on the history of local observations, actions and rewards the agent perceives, to minimise the A2C loss.

2.2 Centralised Training and Decentralised Execution

Centralised training and decentralised execution (CTDE) is a commonly-used approach in MARL. Centralising learning allows sharing of information during the training phase, while learned decentralised policies are only conditioned on the agents' local observations. Each agent consists of an actor and a critic. The actor only receives the local history of the agent's observations as input, while the critic is trained on the joint observation and action space.

MADDPG: Multi-Agent DDPG (MADDPG) [20] is a variation of the standard DDPG algorithm [19] for MARL. Each agent has its own actor and critic. The actor is trained only conditioned on the history of observations to approximate the policy, and the critic is trained on the joint observation and action of all agents to approximate the joint state-action value function. A common assumption of DDPG (and thus MADDPG) is that actions are differentiable with respect to the parameters of the actor, and as a result the action space must be continuous. Lowe et al. [20] apply the Gumbel-Softmax trick [16, 22] in order to learn with discrete action spaces.

COMA: In Counterfactual Multi-Agent (COMA), Foerster et al. [12] propose a modification of the advantage in the actor's loss computation to perform counterfactual reasoning for credit assignment in cooperative MARL. The advantage is defined as the discrepancy given by the value function of each agent when following its taken action and the value function when following the current policy. This modified advantage and the standard policy loss is used to train the actor, while the critic is trained using the TD-lambda algorithm [34]. Due to the architectural design of COMA, the approach is limited to fully-cooperative environments as it assumes the same reward signal for all agents.

Central-V: Central-V is an actor-critic algorithm in which the actor approximates the individual policy and the critic learns a joint state value function (in contrast, the critics in MADDPG and COMA are also conditioned on actions). It extends existing on-policy actor-critic algorithms, such as A2C [7, 24] or PPO [31], by applying centralised critics conditioned on the state of the environment rather than the individual history of observations. In this work, we use A2C for Central-V and the standard A2C loss is minimised during optimisation.

Table 1: Overview of algorithms and their properties.

	Centr. Training	Off-/On-policy	Value-based	Policy-based
IQL	✗	Off	✓	✗
IA2C	✗	On	✓	✓
MADDPG	✓	Off	✓	✓
COMA	✓	On	✓	✓
Central-V	✓	On	✓	✓
VDN	✓	Off	✓	✗
QMIX	✓	Off	✓	✗

2.3 Value Factorisation Learning

Another recent research direction is the factorisation of the joint state-action value function into individual state-action value functions of each agent. These methods fall under the CTDE paradigm as well, but we discuss them separately because they focus on value function factorisation.

VDN: Value Decomposition Networks (VDN) [33] aim to learn a linear factorisation of the joint Q-value. Each agent maintains a deep network to approximate its own state-action values. VDN factorises the joint Q-value into the sum of individual Q-values. The joint state-action value function is trained using the standard DQN algorithm [23, 38]. During training, the gradient of the joint TD loss flows backwards to the Q-network of every agent. Similar to COMA, VDN also assumes a shared reward signal among all agents and is therefore by design limited to fully-cooperative environments.

QMIX: QMIX [28] extends VDN to address a broader class of environments. To represent a more complex factorisation, a mixing network with trainable parameters is introduced to compute the joint Q-value based on each agent's individual state-action value function. A requirement of the mixing function is that the optimal joint action which maximises the joint Q-value is the same as the combination of the individual actions maximising the Q-values of each agent. This assumption is fulfilled by constraining the joint value function to be monotonically increasing with respect to each individual value function ($\frac{\partial Q_{\text{tot}}}{\partial Q^i} > 0$). QMIX is trained to minimise the DQN loss and the gradient is back-propagated to the individual Q-values, similarly to VDN.

3 Environments

We evaluate the algorithms in four multi-agent environments within which we define a total of 17 different learning tasks. We consider both fully cooperative environments in which agents receive identical rewards, and mixed competitive-cooperative environments in which agents work toward a common goal but may have differing individual rewards. Additional environment properties include the degree of observability (whether agents can see the full environment state or only parts of it), reward density (receiving frequent/dense vs infrequent/sparse non-zero rewards), and the number of agents involved. Table 2 lists environments with properties, and we give brief descriptions below. Full details of environments and learning tasks are provided in Appendix A.

MPE: The Multi-Agent Particle Environments (MPE) [25] consists of several two-dimensional navigation tasks. We investigate four tasks that emphasise coordination: Speaker-Listener, Spread, Adversary, and Predator-Prey. Agent observations consist of high-level feature vectors including relative agent and landmark locations, and the actions allow for two-dimensional navigation. Speaker-Listener also requires communication, and allows the speaker to output binary messages.

SMAC: The StarCraft Multi-Agent Challenge (SMAC) [30] simulates battle scenarios in which a team of controlled agents must destroy an enemy team of agents that use fixed policies. Agents observe other agents within a fixed radius, and can move around and select enemies to attack. Rewards are shared among all agents. We consider five tasks in this environment which vary in number of agents in the teams as well as agent types.

LBF: In Level-Based Foraging [3] (LBF), a group of agents must collect items which are spread randomly in a grid-world. Agents and items have skill levels, such that a group of one of more agents can collect an item if the sum of their levels exceed the item's level. We define five tasks in

Table 2: Overview of environments and their properties.

	Type	Observability	Rew. Sparsity	Agents
MPE	Cooperative	Partial / Full	Dense	2-3
SMAC	Cooperative	Partial	Dense	2-9
LBF	Mixed	Partial / Full	Sparse	2-3
RWARE	Collaborative	Partial	Very Sparse	2-4

LBF which vary in world size, number of agents and items, and observability conditions. The main challenge of this environment is its mixed nature requiring agents to switch between cooperating or competing within an episode to achieve their maximum returns.

RWARE: The Multi-Robot Warehouse environment (RWARE) represents a collaborative, partially-observable task with sparse rewards. RWARE simulates a grid-world warehouse in which agents (robots) must locate and deliver requested shelves to work stations and return them after delivery. Agents only observe a 3×3 grid which gives information about the surrounding agents and shelves. We define three tasks which vary in world size and number of agents and shelf requests.

4 Evaluation Protocol and Metrics

4.1 Evaluation Protocol

To account for the improved sample efficiency of off-policy over on-policy algorithms and to allow fair comparisons, we train off-policy algorithms for two million steps and on-policy algorithms for 20 million steps. We evaluate off-policy algorithms every 50,000 steps and on-policy algorithms every 500,000 steps. As a result, a total of 40 evaluations are executed during the training of each algorithm. We train each algorithm with several different hyperparameters using ten different seeds. The implementation details are described in Appendix D.

4.2 Evaluation Metrics

Maximum returns: For each algorithm, we identify the evaluation time step and hyperparameter configuration that achieves the highest average evaluation returns across ten random seeds during training. We then extract the saved models for each of the ten runs with those particular hyperparameters at the identified time step. Using these models, we run 1000 evaluation episodes for each of the ten seeds and compute the average evaluation returns. The reported maximum returns and their deviations as presented in Table 3 correspond to the average and standard deviation across these ten average evaluation returns. Note that in the SMAC we report the average win rate instead of returns.

Average returns: We also report and visualise the average returns achieved throughout all evaluations during training using the same best hyperparameter configuration for each algorithm in Appendix C. Due to this metric being computed over all evaluations executed during training, it considers learning speed besides final achieved returns. To visualise convergence, we show the normalised average returns of all algorithms in the MPE, SMAC and LBF in Figure 1. Returns are normalised across all seeds and tasks of each environment in $[0, 1]$ range.

Given the sensitivity of RL approaches to hyperparameters, seeds [14] and implementation details [10], we also evaluate the reliability of MARL approaches on each environment. For this purpose, we adopt the following metrics suggested by Chan et al. [4] and report rankings of all algorithms for the MPE, SMAC and LBF environments in Figure 2 (see [4] for more details on metrics). We do not report reliability metrics for RWARE since the majority of algorithms did not demonstrate any learning in this environment.

Dispersion across Time (DT): We report the dispersion of returns achieved during training runs. The dispersion refers to the variability of returns and is computed using the inter-quartile range (IQR) of sliding windows of training returns. This metric aims to identify the smoothness of training progress, as high dispersion within each window corresponds to large short-term variability of training returns suggesting noisy training.

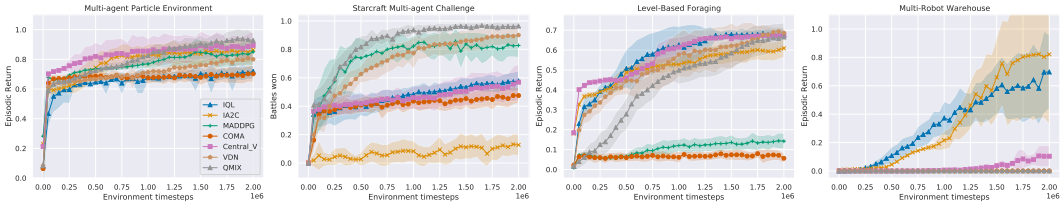


Figure 1: Normalised average returns of all algorithms in all environments.

Short-term Risk across Time (SRT): The short-term risk aims to measure the worst-case drop in evaluation returns from one evaluation to the next during training. For each training run, the conditional value at risk (CVaR) of the difference in evaluation returns among successive evaluations is computed. The CVaR is given by the expected value of the distribution of such differences below its α-quantile. For our experiments, we use $\alpha = 0.05$. Unlike DT, this metric indicates instability in the form of worst, sudden drops in returns rather than general variability throughout training.

Long-term Risk across Time (LRT): To measure long-term risk across training runs, the worst-case drop in evaluation returns during training is reported. Therefore, CVaR is applied to the drawdown [5] rather than adjacent evaluation returns. The drawdown is given by the drop in returns at each evaluation relative to the so-far best evaluation return. Whereas high SRT indicates high, sudden variability during training, this metric gives insight into long-term decay in returns.

Dispersion across Runs (DR): RL is known to be sensitive to random seeds [14]. Therefore, we also report the IQR as a robust metric for dispersion across multiple training runs measuring the variability of final evaluation returns across different random seeds.

5 Results and Analysis

Independent Learning: We find that independent learning approaches such as IQL and IA2C perform competitively in all evaluation metrics on a range of environments despite their simplicity. IA2C with its stochastic policy appears to be particularly effective on MPE tasks, especially predator-prey, spread, and adversary which do not require extensive coordination. The MPE speaker-listener task, for instance, requires significant coordination due to its partial observability and objective such that purely independent learning algorithms appear to be ineffective. While IQL appears less effective in MPE, it achieves high returns on simpler SMAC tasks and both IQL and IA2C achieve high returns on the majority of LBF tasks. In harder SMAC tasks, where agents have to control a variety of melee and ranged units, independent learning appears to suffer from being unable to reason about joint information of all agents which is available during centralised training. IQL ranks low in most reliability metrics, indicating large variability of the performance both within the same run as well as between different runs. Generally, on-policy algorithms such as IA2C appear more reliable than most off-policy approaches, especially in the MPE and LBF. Lastly, we find that independent learning significantly outperforms all other algorithms on RWARE tasks. This environment is challenging due to its very sparse reward signals, and most algorithms fail at learning effective policies for any individual agents enabling them to attempt any form of coordination. Additionally, the environment requires minimal coordination, and despite partial observability, joint information appears to bring little value. In fact, training centralised critics over joint observation and action spaces might even pose a disadvantage due to its increased input size making learning even harder. Latter is particularly relevant in the RWARE environment where observation vectors are large in comparison to other considered environments.

Centralised Training and Decentralised Execution: Centralised training aims to learn powerful critics over the joint observations and actions. This allows for reasoning over a larger information space, but potentially complicates the training process. We find that learning such critics is valuable in tasks which require significant coordination, such as the MPE speaker-listener and harder SMAC tasks due to task objectives or significant partial observability. However, the other MPE tasks (spread, adversary and predator-prey) appear to require less coordination, making effective independent learners competitive compared to this class of algorithms. In the RWARE, centralised training with critics over larger joint observation and action spaces might even make the learning process more difficult without bringing significant value as outlined previously. We find that MADDPG exhibits

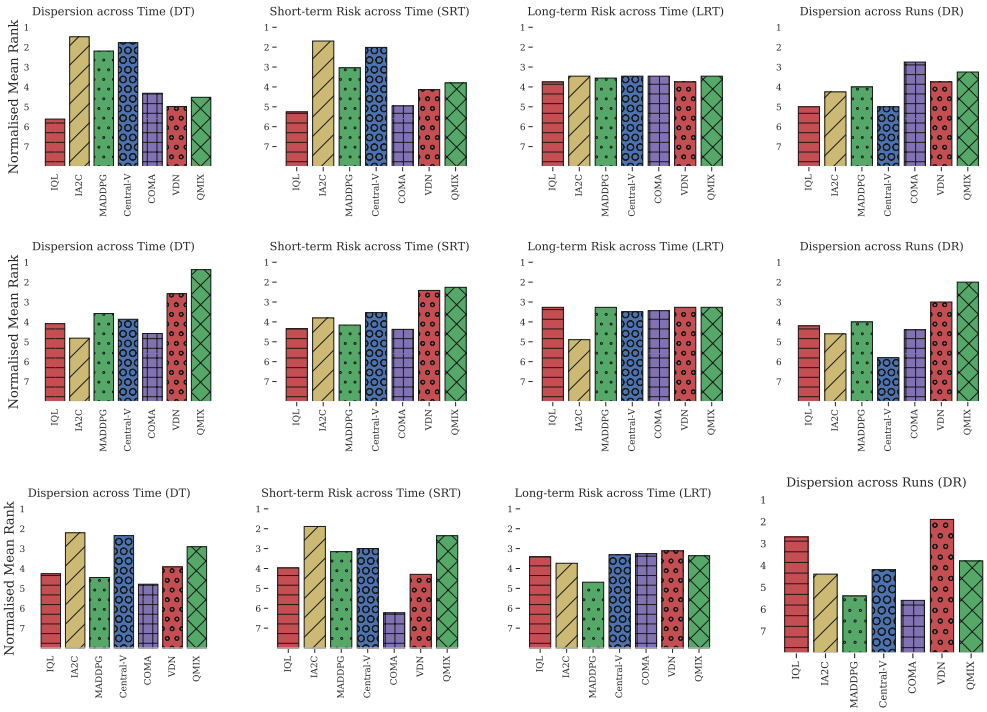


Figure 2: Reliability metrics for MPE (first row), SMAC (second row) and LBF (third row). Bars indicate the ranking, i.e. higher bars correspond to lower dispersion or risk.

particularly high maximum and average returns in MPE and SMAC tasks. However, MADDPG exhibits low returns in discrete grid-world environments such as the LBF and RWARE tasks where Central-V performs more competitively. We believe that this is an effect of the biased categorical reparametarisation using Gumbel-Softmax to apply MADDPG to discrete action spaces. We assume that MADDPG would not suffer such a loss in performance on environments with continuous action dynamics and observation spaces from this effect. In contrast, Central-V suffers compared to MADDPG on SMAC tasks, due to Central-V's critic computing a centralised state value function instead of a centralised state-action value function. As a result, the value function does not take into account the joint action of all the agents limiting its ability to reason over coordinated actions. However, performance is not significantly affected by this in other environments and on the LBF, Central-V even outperforms MADDPG due to the application of a stochastic policy without the bias of the Gumbel-Softmax. With respect to reliability, on-policy Central-V ranks higher than MADDPG which scores particularly low in the SMAC and LBF environments. COMA generally exhibits one of the lowest performances in most environments and only performs competitively in a few MPE and simpler SMAC tasks. Additionally, COMA shows poor reliability with particularly low rankings in DT, SRT and DR indicating high variance and noisy training with consequently large variation across multiple runs. We believe this occurs due to high variance of its counterfactual advantage and consequently large variance critic losses, causing gradients for updates to be unstable. To verify our claim, we conducted further experiments using reward standardisation showing significant improvements across multiple tasks. For more details and analysis, see Appendix B. It should also be noted that reported results in Table 3 and Table 4 for COMA and Central-V on MPE are using such reward standardisation whereas for other environments, results without this modification are reported.

Value Factorisation Learning: Value factorisation is found to be an effective approach for most environments. On the majority of MPE and LBF tasks, VDN and QMIX outperform or at least match the highest returns of any other algorithm. This suggests that VDN and QMIX share the major advantages of centralised training with improved coordination. Additionally on SMAC, their performance is only matched by MADDPG and IQL on simpler tasks, but for harder SMAC tasks value factorisation appears to be very impactful and achieves higher returns than other algorithms. VDN and QMIX generally exhibit similar performance with minor improvements of QMIX over

VDN in many tasks. The biggest disadvantage of VDN is its assumption of linear value function factorisation, which is addressed by the more complex factorisation of QMIX. With its factorisation, QMIX aims to improve learnability, i.e. it simplifies the learning objective for each agent to maximise, while maintaining the factorisation to ensure the global objective is maximised by all agents [1]. This allows QMIX to perform better in complex SMAC tasks while for other environments like LBF, where the global utility can adequately be represented by the sum of individual agents' utilities, little difference between VDN and QMIX can be observed. In the RWARE, VDN and QMIX just as COMA were trained using shared, total rewards due to these approaches assuming fully-cooperative tasks with identical rewards across all agents. This reward scheme introduces the multi-agent credit assignment problem making tasks more difficult. In combination with the sparsity of rewards, this appears to lead to no learning being demonstrated by either VDN or QMIX throughout the entire training. We found that IA2C was able to learn to similar success in the RWARE using these fully-cooperative total rewards, showing that learning is still feasible. Hence, we believe that the lack of learning exhibited by VDN and QMIX in this environment does show a potential reliance on sufficiently dense rewards. VDN is found to be reliable throughout all environments with QMIX ranking slightly higher in almost all cases. However, both algorithms rank lower in training stability metrics DT and SRT compared to on-policy algorithms IA2C and Central-V on the MPE and LBF environments.

Summary: Overall, we observe that there is no single algorithm that performs efficiently across all tasks, and the performance of the algorithms depends strongly on the properties of the task.

1) Independent learning methods are found to achieve competitive returns on a range of environments despite their simplicity.
2) Centralised training is primarily effective in partially-observable environments and when high-degree of coordination is required. On the other hand, the increased complexity of centralised critics can impede training in environments where no significant coordination is needed. This appears to be particularly important when combined with large observation spaces and sparse rewards. One example of this is the IA2C and Central-V comparison in RWARE, where the latter apparently under-performs due to the larger input to the critic network.
3) Training centralised state-action values (Q) is more effective than training centralised state values (V) because the former takes into account the joint action of the agents.
4) Reward standardisation reduces the variance of baselines in actor-critic algorithms and can improve performance significantly. We investigate the impact of such standardisation further at the example of COMA which appears to suffer extremely from high variance of its counterfactual advantage in most tasks. Further analysis can be found in Appendix B.
5) QMIX and MADDPG are known to achieve state-of-the-art performance in SMAC and MPE respectively. In our comparative evaluation on a variety of tasks, we find that both these algorithms perform adequately, but QMIX outperforms MADDPG in harder SMAC and LBF tasks.
6) Gumbel-Softmax seems to be ineffective in grid-worlds with non-continuous dynamics.

6 Conclusion

We evaluated seven multi-agent deep reinforcement learning algorithms in a total of 17 learning tasks, including combinations of cooperative/mixed game types, partial/full observability, sparse/dense rewards, and a number of agents that range from two up to nine. We compared algorithm performance in terms of training curves and final achieved returns, and reported results on several reliability metrics designed to measure the robustness of learning processes. To conclude, while deep RL has improved considerably in recent years, MARL still requires much research effort. Our results in the LBF and RWARE environments in particular suggest future research directions. MARL algorithms appear to suffer from a lack of coordinated exploration which is particularly noticeable in sparsely-rewarded tasks. Also, mixed LBF tasks, requiring agents to cooperate and compete in a single episode, raise the need for role assignment techniques. This work is limited to mostly cooperative environments and commonly-used MARL algorithms. Competitive environments as well as solutions to a variety of MARL challenges such as exploration, communication, and opponent modelling need to be studied separately in the future. We hope that our work sheds some light on the relative strengths and limitations of existing MARL algorithms, to provide guidance in terms of practical considerations and future research.

Table 3: Results in all tasks of four different environments. The maximum returns during evaluation and the standard deviation among different seeds is presented. Ten seeds were used in all tasks and algorithms. The algorithms that achieve the highest returns are presented in bold.

Tasks \Algorithms	IQL	IA2C	MADDPG	COMA	Central-V	VDN	QMIX
MPE Speaker-Listener	-59.01 ± 6.39	-33.19 ± 14.22	-28.04 ± 3.36	-46.63 ± 6.29	-28.86 ± 7.14	-43.83 ± 14.12	$\mathbf{-24.56 \pm 3.23}$
MPE Spread	-642.47 ± 15.81	-397 ± 6.95	$\mathbf{-383.56 \pm 8.15}$	-549.32 ± 6.50	-396.32 ± 2.18	-640.06 ± 8.86	-476.67 ± 23.80
MPE Adversary(PT)	17.05 ± 1.77	$\mathbf{21.85 \pm 0.76}$	$\mathbf{23.02 \pm 2.69}$	18.53 ± 0.62	$\mathbf{22.51 \pm 0.98}$	18.51 ± 2.10	$\mathbf{20.54 \pm 1.80}$
MPE Predator-Prey(PT)	29.73 ± 15.36	$\mathbf{95.06 \pm 12.41}$	42.57 ± 26.46	2.11 ± 0.51	55.69 ± 28.86	61.44 ± 9.33	$\mathbf{88.95 \pm 18.44}$
SMAC 3m	$\mathbf{0.99 \pm 0.01}$	0.67 ± 0.35	$\mathbf{0.98 \pm 0.01}$	0.84 ± 0.10	0.82 ± 0.06	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$
SMAC 8m	0.96 ± 0.03	0.11 ± 0.28	$\mathbf{0.98 \pm 0.02}$	0.95 ± 0.02	0.96 ± 0.03	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$
SMAC 2s3z	0.70 ± 0.13	0.05 ± 0.14	0.96 ± 0.02	0.35 ± 0.13	0.77 ± 0.14	0.96 ± 0.02	$\mathbf{0.98 \pm 0.01}$
SMAC 3s5z	0.12 ± 0.12	0.00 ± 0.00	0.72 ± 0.08	0.01 ± 0.01	0.12 ± 0.18	0.69 ± 0.19	$\mathbf{0.95 \pm 0.03}$
SMAC 1c3s5z	0.19 ± 0.11	0.02 ± 0.07	0.72 ± 0.07	0.26 ± 0.07	0.23 ± 0.23	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.94 \pm 0.04}$
LBF-8x8-2p-3f	0.97 ± 0.02	0.93 ± 0.03	0.47 ± 0.08	0.18 ± 0.04	0.93 ± 0.01	0.97 ± 0.01	$\mathbf{0.99 \pm 0.01}$
LBF-8x8-2p-2f-coop	$\mathbf{0.99 \pm 0.01}$	0.89 ± 0.03	0.007 ± 0.02	0.01 ± 0.02	0.89 ± 0.02	$\mathbf{0.98 \pm 0.02}$	0.82 ± 0.28
LBF-8x8-3p-1f-coop	0.002 ± 0.004	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.001 ± 0.003	0.00 ± 0.00
LBF-10x10-2p-8f	0.53 ± 0.04	0.32 ± 0.04	0.09 ± 0.03	0.09 ± 0.02	0.39 ± 0.05	$\mathbf{0.60 \pm 0.07}$	$\mathbf{0.57 \pm 0.06}$
LBF-8x8-2p-3f, sight=2	0.95 ± 0.02	0.94 ± 0.01	0.16 ± 0.06	0.16 ± 0.07	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.96 \pm 0.02}$	$\mathbf{0.97 \pm 0.02}$
RWARE tiny 2p	12.51 ± 4.76	$\mathbf{17.88 \pm 3.71}$	0.00 ± 0.00	0.000 ± 0.000	4.10 ± 1.45	0.001 ± 0.003	0.000 ± 0.000
RWARE tiny 4p	13.15 ± 7.44	$\mathbf{38.87 \pm 10.09}$	0.00 ± 0.00	0.001 ± 0.003	1.95 ± 0.23	0.001 ± 0.003	0.001 ± 0.003
RWARE small 2p	1.27 ± 0.28	$\mathbf{2.04 \pm 0.46}$	0.00 ± 0.00	0.001 ± 0.003	0.001 ± 0.001	0.001 ± 0.003	0.001 ± 0.003

Acknowledgement

We would like to thank Josiah Hanna, Arrasy Rahman, and Elliot Fosong for providing feedback on early drafts of this work.

References

[1] Adrian K Agogino and Kagan Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *International Conference on Autonomous Agents and Multi-Agent Systems*, 2008.

[2] Stefano V. Albrecht and Subramanian Ramamoorthy. Comparative evaluation of MAL algorithms in a diverse set of ad hoc team problems. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2012.

[3] Stefano V. Albrecht and Subramanian Ramamoorthy. 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*, 2013.

[4] Stephanie CY Chan, Sam Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. *International Conference on Learning Representations*, 2020.

[5] Alexei Chekhlov, Stanislav Uryasev, and Michael Zabarankin. Drawdown measure in portfolio optimization. *International Journal of Theoretical and Applied Finance*, 2005.

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Advances in Neural Information Processing Systems Workshop on Deep Learning*, 2014.

[7] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI baselines. `https://github.com/openai/baselines`, 2017.

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

[9] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 2016.

[10] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *International Conference on Learning Representations*, 2019.

[11] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2016.

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

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

[14] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2018.

[15] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *International Conference on Machine Learning*, 2019.

[16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

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

[19] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

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

[22] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[23] 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. Human-level control through deep reinforcement learning. *Nature*, 2015.

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

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

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

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

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

[29] Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-agent actor-critic with hierarchical graph attention network. *AAAI Conference on Artificial Intelligence*, 2020.

[30] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2019.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[32] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2016.

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

[34] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 1988.

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

[36] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[37] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[38] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 1992.

A Task Specifications

Below, we provide details and descriptions of the environments and tasks used for the evaluation.

A.1 Multi-Agent Particle Environment [25]

This environment consists of multiple tasks involving the cooperation and competition between agents. All tasks involve particles and landmarks in a continuous two-dimensional environment. Observations consist of high-level feature vectors and agents are receiving dense reward signals. The action space among all tasks and agents is discrete and usually includes five possible actions corresponding to no movement, move right, move left, move up or move down. All experiments on this environment are executed with a maximum episode length of 25, i.e. episodes are terminated after 25 episodes and a new episode is started. All considered tasks are visualised in Figure 3.

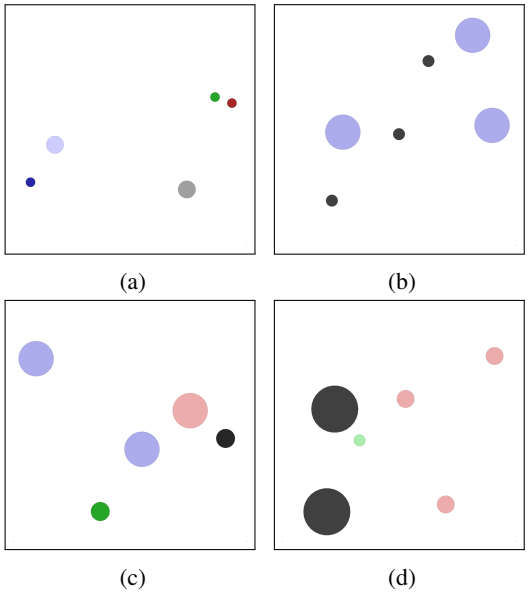


Figure 3: Illustration of Multi-Agent Particle Environment tasks (a) speaker-listener, (b) spread, (c) adversary and (d) predator-prey.

MPE Speaker-Listener: In this fully cooperative task, one static speaker agent has to communicate a goal landmark to a listening agent capable of moving. There are a total of three landmarks in the environment and both agents are rewarded with the negative Euclidean distance of the listener agent towards the goal landmark. The speaker agent only observes the colour of the goal landmark. Meanwhile, the listener agent receives its velocity, relative position to each landmark and the communication of the speaker agent as its observation. As actions, the speaker agent has three possible options which have to be trained to encode the goal landmark while the listener agent follows the typical five discrete movement agents of MPE tasks.

MPE Spread: In this fully cooperative task, three agents are trained to move to three landmarks while avoiding collisions with each other. All agents receive their velocity, position, relative position to all other agents and landmarks. The action space of each agent contains five discrete movement actions. Agents are rewarded with the sum of negative minimum distances from each landmark to any agent and an additional term is added to punish collisions among agents.

MPE Adversary: In this competitive task, two cooperating agents compete with a third adversary agent. There are two landmarks out of which one is randomly selected to be the goal landmark. Cooperative agents receive their relative position to the goal as well as relative position to all other agents and landmarks as observations. However, the adversary agent observes all relative positions without receiving information about the goal landmark. All agents have five discrete movement actions. Agents are rewarded with the negative minimum distance to the goal while the cooperative agents are additionally rewarded for the distance of the adversary agent to the goal landmark. Therefore, the cooperative agents have to move to both landmarks to avoid the adversary from identifying which landmark is the goal and reaching it as well. For this competitive scenario, we use a fully cooperative version where the adversary agent is controlled by a pretrained model obtained by training all agents using the MADDPG algorithm for 25000 episodes.

MPE Predator-Prey: In this competitive task, three cooperating predators hunt a forth agent controlling a faster prey. Two landmarks are placed in the environment as obstacles. All agents receive their own velocity and position as well as relative positions to all other landmarks and agents as observations. Predator agents also observe the velocity of the prey. All agents choose among five movement actions. The agent controlling the prey is punished for any collisions with predators as well as for leaving the observable environment area (to prevent it from simply running away but

learning to evade). Predator agents are collectively rewarded for collisions with the prey. We also employ a fully cooperative version of this task with a pretrained prey agent. Just as for the Adversary task, the model for the prey is obtained by training all agents using the MADDPG algorithm for 25000 episodes.

A.2 StarCraft Multi-Agent Challenge [30]

The StarCraft Multi-Agent Challenge is a set of fully cooperative, partially observable multi-agent tasks. This environment implements a variety of micromanagement tasks based on the popular real-time strategy game StarCraft II[2] and makes use of the StarCraft II Learning Environment (SC2LE) [36]. Each task is a specific combat scenario in which a team of agents, each agent controlling an individual unit, battles against a army controlled by the centralised built-in game AI of the game of StarCraft. These tasks require agents to learn precise sequences of actions to enable skills like *kiting* as well as coordinate their actions to focus their attention on specific opposing units. All considered tasks are symmetric in their structure, i.e. both armies are constructed by the same units. Figure 4 gives renders of each considered task in this environment.

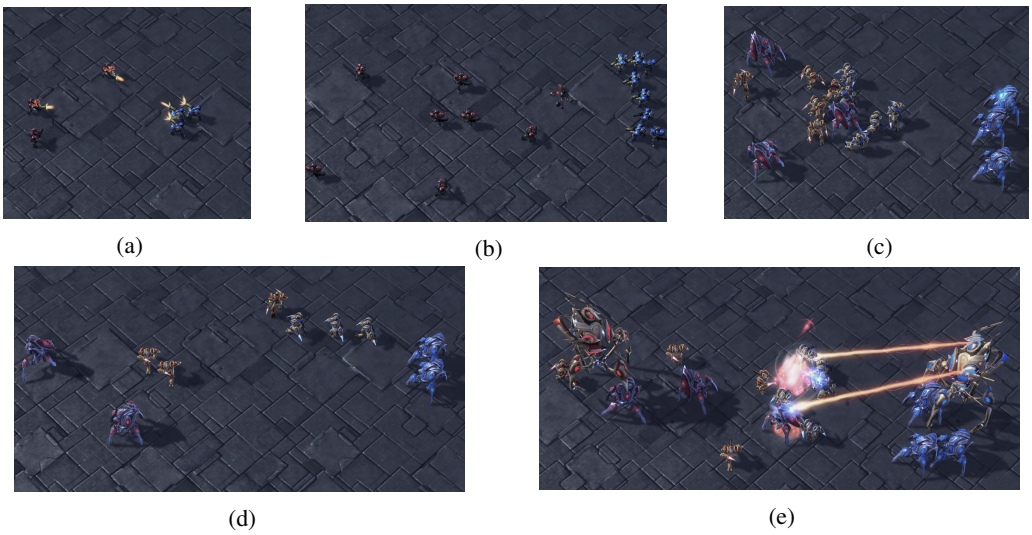


Figure 4: Illustration of StarCraft Multi-Agent Challenge tasks (a) 3m, (b) 8m, (c) 2s3z, (d) 3s5z and (e) 1c3s5z.

SMAC 3m: In this scenario, each team is constructed by three space marines. These ranged units have to be controlled to focus fire on a single opponent unit at a time and attack collectively to win this battle.

SMAC 8m: In this scenario, each team controls eight space marines. While the general strategy is identical to the 3m scenario, coordination becomes more challenging due to the increased number of agents and marines controlled by the agents.

SMAC 2s3z: In this scenario, each team controls two stalkers and three zealots. While stalkers are ranged units, zealots are melee units, i.e. they are required to move closely to enemy units to attack. Therefore, own units still have to learn to focus their fire on single opponent units at a time. Additionally, stalkers are required to learn kiting to consistently move back in between attacks to keep a distance between themselves and enemy zealots to minimise received damage while maintaining high damage output.

SMAC 3s5z: This scenario requires the same strategy as the 2s3z task. Both teams control three stalker and five zealot units. Due to the increased number of agents, the task becomes slightly more challenging.

[2]StarCraft II is a trademark of Blizzard Entertainment™.

SMAC 1c3s5z: In this final scenario, both teams control one colossus in addition to three stalkers and five zealots. A colossus is a durable unit with ranged, spread attacks. Its attacks can hit multiple enemy units at once. Therefore, the controlled team now as to coordinate to avoid many units to be hit by the enemy colossus at ones while enabling the own colossus to hit multiple enemies all together.

A.3 Level-Based Foraging [3]

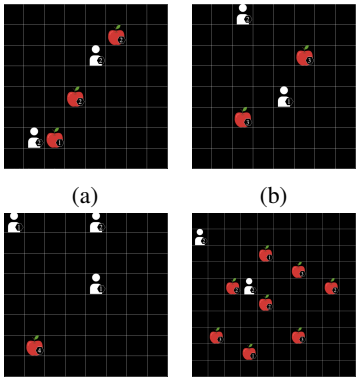


Figure 5: Illustrations of (a) LBF-8x8-2p-3f, (b) LBF-8x8-2p-2f-coop, (c) LBF-8x8-3p-1f-coop and (d) LBF-10x10-2p-8f.

The Level-Based Foraging environment consists of mixed cooperative-competitive tasks focusing on the coordination of involved agents. The task for each agent is to navigate the grid-world map and collect item. Each agent and item is assigned a level and item is randomly scattered in the environment. In order to collect item, agents have to choose a certain action next to the item. However, such collection is only successful if the sum of involved agents' levels is equal or greater than the item level. Agents receive reward equal to the level of collected item. Figure 5 shows the tasks used for our experiments. By default, every agent can observe the whole map, including the positions and levels of all the entities and can choose act by moving in one of four directions or attempt to load an item. In the partially observable version, denoted with 'sight=2', agents can only observe entities in a 5×5 grid surrounding them.

LBF-8x8-2p-3f: An 8×8 grid-world with two agents and three items placed in random locations. Item levels are random and might require agents to cooperate, depending on the level.

LBF-8x8-2p-2f-coop: An 8×8 grid-world with two agents and two items. This is a cooperative version and agents will always need too collect an item simultaneously (cooperate).

LBF-8x8-3p-1f-coop: An 8×8 grid-world with three agents and one item. This is a cooperative version and all three agents will need to collect the item simultaneously.

LBF-10x10-2p-8f: A 10×10 grid-world with two agents and ten items. The time-limit (25 timesteps) is often not enough for all items to be collected. Therefore, the agents need to spread out and collect as many items as possible in the short amount of time.

LBF-8x8-2p-3f, sight=2: Similar to the first variation, but partially observable. The agent's vision is limited to a 5×5 box centred around the agent.

A.4 Multi-Robot Warehouse

The multi-robot warehouse environment is a set of collaborative, partially observable multi-agent tasks simulating a warehouse operated by robots. Each agent controls a single robot aiming to collect requested shelves. At all times, N shelves are requested and each timestep a request is delivered to the goal location, a new (currently unrequested) shelf is uniformly sampled and added to the list of requests. Agents observe a 3×3 grid including information about potentially close agents, given by their location and rotation, as well as information on surrounding shelves and a list of requests. The action space is discrete and contains of four actions corresponding to turning left or right, moving forward and loading or unloading a shelf. While agents have to cooperate to ideally deliver requested shelves, the task is not fully-cooperative as rewards among agents vary. Agents are rewarded once they deliver a requested shelf to a goal position. However, only the delivering agent is rewarded and this is the only timestep of any non-zero

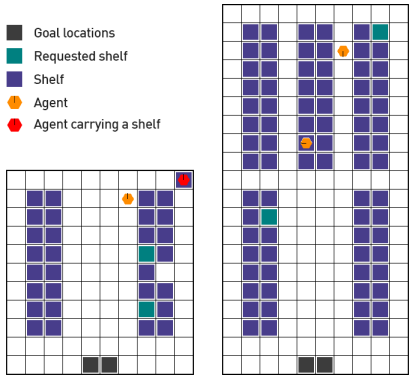


Figure 6: Illustrations of (a) tiny-2ag and (b) small-2ag.

rewards making this environment very sparsely rewarded. We use multi-robot warehouse tasks with warehouses of varying size and number of agents N (which is also equal to the number of requested shelves). Figure 6 illustrates the tiny and small warehouses with 2 agents.

RWARE-tiny-2p: A 10×11 grid-world with two agents.

RWARE-tiny-4p: A 10×11 grid-world with four agents.

RWARE-small-2p: A 10×20 grid-world with two agents. The rewards are much sparser due to the much larger world.

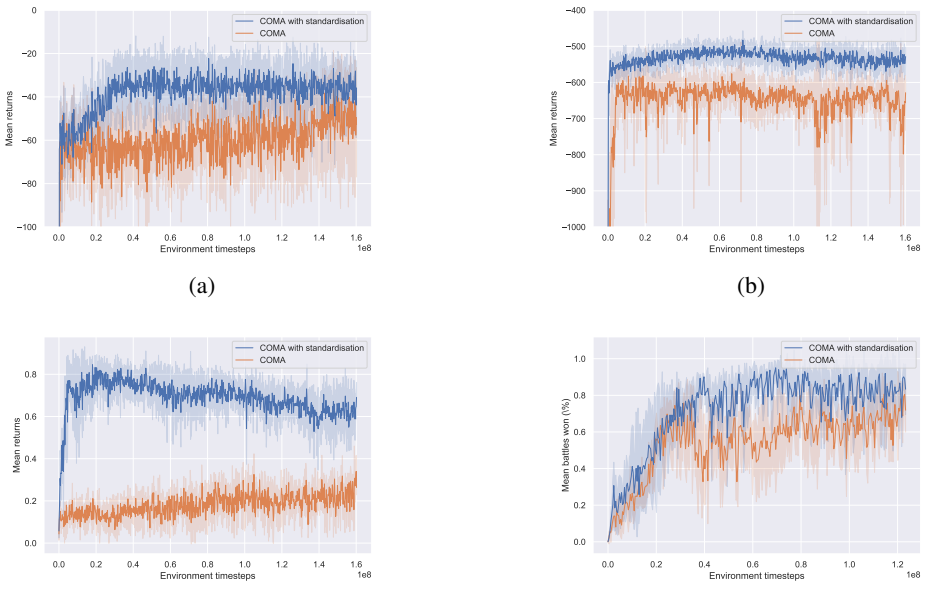
B COMA Baseline Analysis

We conducted further experiments to verify our hypothesis that COMA's counterfactual baseline is ineffective in reducing variance of the reward signal. In order to gain insight on this matter, we conduct experiments on four tasks: MPE Speaker-Listener, MPE Spread, LBF 8x8-2p-3f and SMAC 2s3z. For all these tasks, we ran experiments using five random seeds with the same hyperparameters as specified in Appendix D. Training was extended to 160M environment timesteps for MPE and LBF tasks and about 120M environment timesteps for SMAC 2s3z. We run these experiments once without any modification and once applying standardisation to each batch of reward signals during the optimisation algorithm by computing

$$\bar{r} = \frac{r - \mu(r)}{\sigma(r) + \delta}$$

where r refers to the batch of rewards, $\mu(r)$ and $\sigma(r)$ denote the mean and standard deviation across this reward batch. δ is a small constant introduced for numerical stability (we use $\delta = 1^{-10}$ for these experiments). Given our hypothesis, such standardisation would be expected to reduce the variance of the advantage leading to more stable optimisation. The average evaluation returns for COMA on both MPE and the LBF task as well as the average evaluation win rates for SMAC 2s3z with and without standardisation can be seen in Figure 7. Shading of all plots indicates a single standard deviation.

We find that average returns significantly increase and exhibit improved stability when reward standardisation is applied on all four tasks. This indicates the ineffectiveness of the applied baseline in reducing variance successfully and highlights the consistent benefit of applying such standardisation to rewards for actor-critic algorithms.



(a) (b)

(c) (d)

Figure 7: Average returns for COMA with and without standardised rewards in (a) MPE Speaker-Listener, (b) MPE Spread, (c) LBF 8x8-2p-3f and average win rate in (d) SMAC 2s3z.

C Average Return During Training

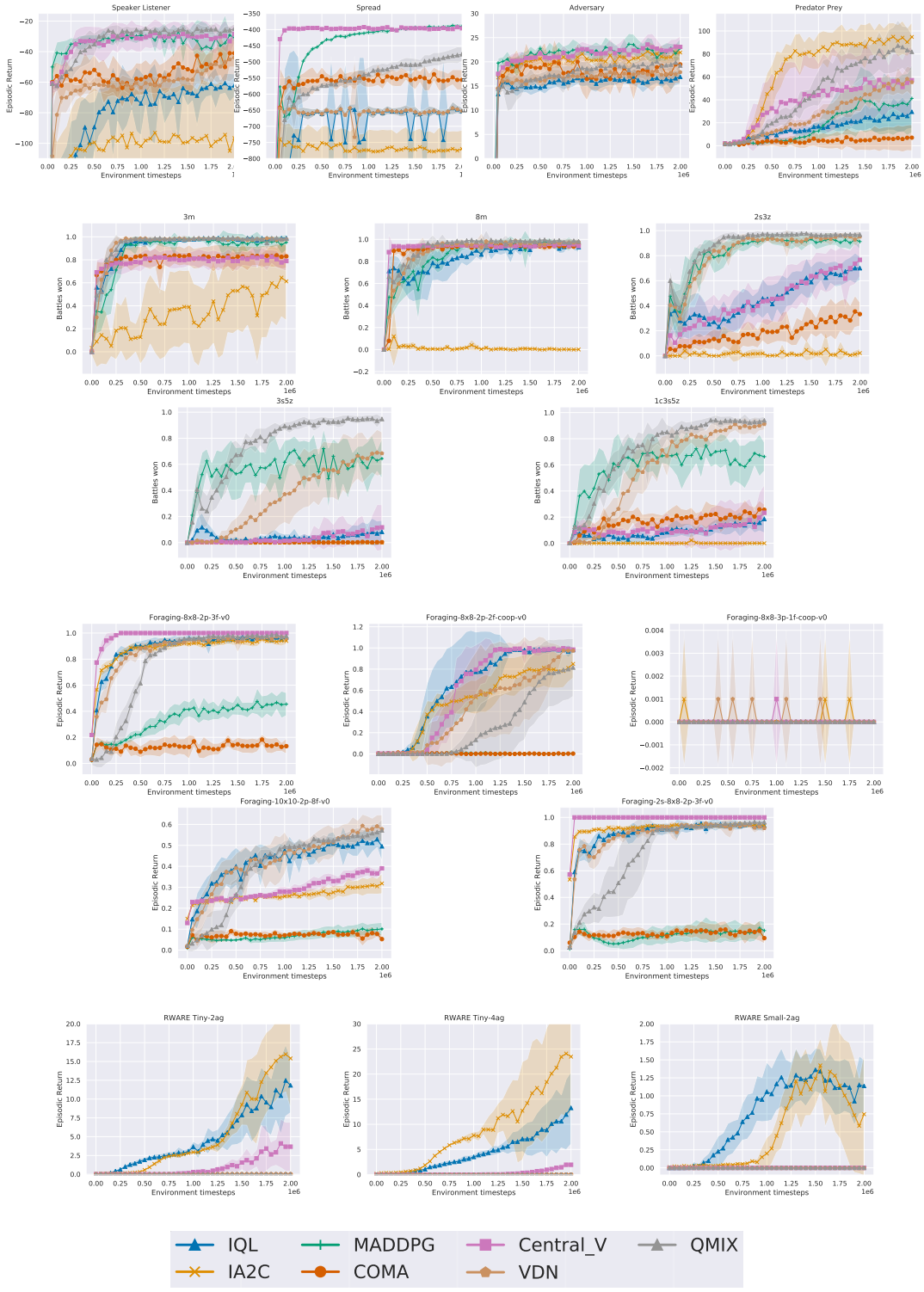


Figure 8: Average episodic returns and standard deviation during training for all seven algorithms in 17 tasks.

Table 4: Results in all tasks of four different environments. The average return and the standard deviation during training among different seeds is presented. Ten seeds were used in all tasks and algorithms.

Tasks \Algorithms	IQL	IA2C	MADDPG	COMA	Central-V	VDN	QMIX
MPE Speaker-Listener	-85.8 ± 59.8	-43.2 ± 62.8	-33.3 ± 72.4	-54.5 ± 5.7	-33.5 ± 80.3	-66.9 ± 56.0	-37.8 ± 57.1
MPE Spread	-704.1 ± 198.0	-402.7 ± 93.9	-438.2 ± 110.3	-557.7 ± 16.5	-396.5 ± 98.0	-667.7 ± 103.1	-558.3 ± 118.2
MPE Adversary (PT)	14.98 ± 7.63	20.32 ± 8.92	21.85 ± 7.68	17.99 ± 1.18	21.60 ± 8.06	16.05 ± 7.71	16.87 ± 8.05
MPE Predator-Prey (PT)	15.92 ± 10.49	68.71 ± 35.13	21.78 ± 21.35	2.57 ± 1.25	39.49 ± 32.88	28.09 ± 20.01	47.00 ± 32.69
SMAC - 3m	0.91 ± 0.19	0.36 ± 0.36	0.89 ± 0.23	0.79 ± 0.16	0.77 ± 0.15	0.92 ± 0.20	0.92 ± 0.19
SMAC - 8m	0.83 ± 0.20	0.01 ± 0.05	0.87 ± 0.2	0.89 ± 0.20	0.92 ± 0.15	0.88 ± 0.21	0.91 ± 0.18
SMAC - 2s3z	0.45 ± 0.21	0.01 ± 0.04	0.83 ± 0.23	0.18 ± 0.13	0.44 ± 0.25	0.81 ± 0.23	0.87 ± 0.22
SMAC - 3s5z	0.04 ± 0.06	0.00 ± 0.00	0.57 ± 0.21	0.00 ± 0.01	0.03 ± 0.08	0.34 ± 0.30	0.75 ± 0.27
SMAC - 1c3s5z	0.09 ± 0.07	0.00 ± 0.01	0.61 ± 0.21	0.17 ± 0.09	0.11 ± 0.13	0.57 ± 0.34	0.71 ± 0.29
LBF-8x8-2p-3f	0.89 ± 0.16	0.89 ± 0.13	0.34 ± 0.14	0.13 ± 0.06	0.91 ± 0.21	0.87 ± 0.18	0.76 ± 0.33
LBF-8x8-2p-2f-coop	0.64 ± 0.43	0.51 ± 0.32	0.003 ± 0.01	0.00 ± 0.01	0.61 ± 0.47	0.44 ± 0.45	0.26 ± 0.38
LBF-8x8-3p-1f-coop	0.00 ± 0.00	0.00 ± 0.00	0 ± 0	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
LBF-10x10-2p-8f	0.42 ± 0.13	0.26 ± 0.04	0.07 ± 0.03	0.07 ± 0.02	0.29 ± 0.06	0.43 ± 0.15	0.40 ± 0.18
LBF-8x8-2p-3f, sight=2	0.88 ± 0.14	0.93 ± 0.07	0.12 ± 0.07	0.13 ± 0.05	0.97 ± 0.26	0.87 ± 0.15	0.74 ± 0.30
RWARE tiny 2p	4.74 ± 4.32	7.81 ± 7.7	0.00 ± 0.00	0.00 ± 0.00	1.85 ± 2.7	0.00 ± 0.00	0.00 ± 0.00
RWARE tiny 4p	4.45 ± 4.62	16.11 ± 14.72	0.00 ± 0.00	0.00 ± 0.00	0.84 ± 0.95	0.00 ± 0.00	0.00 ± 0.00
RWARE small 2p	0.77 ± 0.56	0.93 ± 0.92	0.00 ± 0.00	0.00 ± 0.00	1.67 ± 0.001	0.00 ± 0.00	0.00 ± 0.00

D Implementation details

All algorithms use two hidden layers with ReLU [21] activation function. The number of hidden nodes in each layer range between 64 and 128 for different algorithms and environments. For optimising the objectives we use the Adam optimiser [18] with the learning rate ranging between 0.0001 and 0.0007. COMA, VDN and QMIX use a GRU [6] and share all the parameters between the agents. Experimenting with MADDPG using GRU and shared parameters in the SMAC did not improve the episodic returns. We use different parallel environments [24] in on-policy algorithms and an experience replay [23] for off-policy algorithms to break the correlation between consecutive samples. In IA2C, and Central-V we use 10 parallel environments, while in COMA we use 8 parallel environments. In IA2C, and Central-V we update the parameters using a batch size between 5 and 25. In VDN, and QMIX we update the parameters at the end of each episode, using a batch of 32 episodes. In MADDPG we update the parameters every 100 time steps with a batch size of 1024. In IQL, MADDPG, VDN, and QMIX we use epsilon-greedy policy for exploration. The value of epsilon starts at 1 at the beginning of the episode and we reduce is linearly to 0.05. In IA2C, COMA, and Central-V, the agents explore the environment by sampling actions from the stochastic policy.