Generalization and Cross-Algorithm Cooperation in Cooperative Multi-Agent RL: Evaluating MAPPO and MAA2C

Lorenzo Venturi
University of Bolognaa
lorenzo.venturi14@studio.unibo.it

Abstract

On-policy algorithms in multi-agent reinforcement learning constitute a broad class of methods aimed at directly optimizing agents' policies through continuous interaction with the environment. In this work, two prominent algorithms – Multi-Agent Proximal Policy Optimization (MAPPO) and Multi-Agent Advanced Actor-Critic (MAA2C) – are implemented from scratch and are systematically compared in terms of absolute performance, convergence speed, and generalization across different and unseen layouts on the benchmark Overcooked-AI. Finally, we evaluate an innovative concept of **cross-algorithm cooperation** by pairing MAPPO and MAA2C, each trained on diverse layouts, to tackle a challenging task that requires coordination between heterogeneous learning algorithms.

1 Introduction

Multi-agent reinforcement learning (MARL) is a sub-field of reinforcement leaning that focuses on multiple agents that coexist in a shared environment. The benchmark for this is called Overcooked-AI, based on the popular game Overcooked. In this environment two agents act together in a cooperative settings in order to deliver soups as fast as possible. The focus of this work is to compare the capabilities of two of the most influential contemporary MARL algorithms: Multi-Agent Proximal Policy Optimization (MAPPO) and Multi-Agent Advanced Actor-Critic (MAA2C). To ensure maximum comparability, both algorithms have been implemented from scratch using Pytorch, their networks and their training episodes are the same. Algorithm-specific hyperparameters were chosen based on the best values found for each method. We first evaluate both algorithms on single-layout self-play, to assess their ability to coordinate on a fixed layout. Next, we test their generalization capabilities by training on multiple layouts, with the goal of learning strategies that work across different and unseen environmental configurations. Finally, we explore a cross-evaluation approach, pairing MAPPO agents with MAA2C agents to investigate how heterogeneous algorithms cooperate on challenging tasks, demonstrating the potential for cross-algorithm collaboration in multi-agent environments.

2 Background

2.1 On-Policy Algorithms

The algorithms explored in this work belong to a class called *on-policy* algorithms, a subclass of model-free reinforcement learning alongside off-policy algorithms. On-policy algorithms attempt to improve upon the current behavior policy π that is used to make decisions; therefore, these algorithms learn the value of the policy actually carried out by the agent, typically represented by the state-value function $\hat{v}(s; w)$ or the state-action value function $\hat{Q}(s, a; w)$, where w denotes the parameters of the function approximator.

Off-policy algorithms, in contrast, aim to learn the value of the optimal policy π^*, expressed as $\hat{v}(s; w^*)$ or $\hat{Q}(s, a; w^*)$. Crucially, they can improve a policy that differs from the behavior policy, enabling the reuse of past experience and interaction data generated under alternative policies.

2.1.1 Multi-Agent Advanced Actor-Critic

Multi Agent Advanced Actor-Critic (MAAC) is an extension of the A2C (Advantage Actor-Critic) algorithm for a multi-agent setting. It uses a centralized critic and decentralized actors, the centralized critic has access to the observations and actions of all the agents. The decentralized actors use their individual observations (which can be global or not) independently to choose their own actions. This means that the agents do not consult with each others before taking the actual actions. The critic's role is to learn a centralized value function, which is then used to compute the advantage estimates for each agent's policy update. This structure enables the agents to learn effective cooperative strategies while maintaining the simplicity of independent policy execution.

Advantage Function. A2C, and so MAA2C, use the *advantage function*, which provides a measure of how much better taking a specific action a in state s is compared to the expected value of that state under the current policy:

$$\delta_{ADV} = \hat{Q}(S_t, A_t; w) - \hat{v}(S_t; w)$$

where $\hat{Q}$ and $\hat{v}$ denote parametric estimators (with parameters w) of the state–action value function and the state value function, respectively. A positive advantage indicates that the action a is better than the policy's average behavior at s, whereas a negative advantage indicates the opposite. Using the advantage instead of raw returns helps reduce variance in the gradient estimates, which improves stability during training.

2.1.2 Multi-Agent Proximal Policy Optimization

Multi-Agent Proximal Policy Optimization (MAPPO) is an extension of the widely known Proximal Policy Optimization (PPO) [2] algorithm to multi-agent environments. Similar to MAA2C, MAPPO employs a centralized critic and decentralized actors. This design allows for efficient decentralized execution while still benefiting from the additional stability provided by centralized training.

The core contribution of PPO is its clipped surrogate objective, which constrains the size of policy updates and thereby prevents destructive changes that can arise from high-variance gradient estimates. The clipped objective for a single agent is given by:

$$L(s, a, \theta_t, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)}\delta^{\pi_{\theta_t}}_{ADV},\ clip\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_t}(a|s)}, 1-\varepsilon, 1+\varepsilon\right)\delta^{\pi_{\theta_t}}_{ADV}(s, a)\right)$$

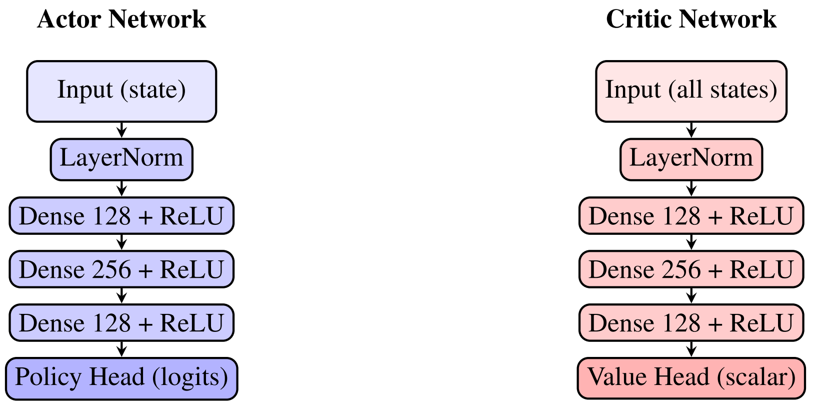
where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio between the updated and the old policy, $\delta^{\pi_{\theta_t}}_{ADV}$ is the advantage estimate, and ϵ is a hyperparameter controlling the update trust region. By clipping the ratio $r_t(\theta)$, PPO prevents large deviations between successive policies, thereby achieving a balance between exploration and stable learning.

3 Implementation Choices and Experiments

In our implementation, both algorithms follow the *centralized training, decentralized execution* (CTDE) paradigm. This approach allows the agents to leverage global information during training while acting independently during execution. To facilitate this, we use a single actor network shared across all agents, reducing model complexity and promoting generalization. The critic, on the other hand, is centralized: it receives the concatenated observations of all agents and estimates the joint state-value function, helping mitigate the non-stationarity inherent in multi-agent environments. We also adopted a modular design, separating the components for agents, trainers, networks, and replay buffers, which allows for flexible experimentation.

3.1 Network Architecture Strategy

For both algorithms, the actor network is a feedforward neural network. Layer normalization is applied at the input to stabilize training, and ReLU activations are used throughout the hidden layers. The actor outputs logits for a categorical distribution over the discrete action space, enabling stochastic action selection. The critic network is similar in structure but instead receives concatenated observations from all agents as input and outputs a single scalar value representing the state-value function. This centralized critic plays a key role in computing the advantage estimates for policy updates.



3.2 MAA2C Implementation Strategy

The Multi-Agent Advantage Actor-Critic (MAA2C) algorithm is implemented as a standard policy gradient method with one update per batch. Trajectories are collected using the current policy, and advantages are computed using Generalized Advantage Estimation (GAE). Each batch results in a single policy update, where the actor loss is defined as the negative expected value of the log probability of actions weighted by their advantages, with an additional entropy term to encourage exploration.

$$L_{\text{actor}} = -\frac{1}{N}\sum_{i=1}^{N} \log \pi_\theta(a_i \mid s_i)\, \delta_{ADV,i} \;-\; \beta\, \mathcal{H}[\pi_\theta(\cdot \mid s_i)]\,.$$

The critic is trained using a mean squared error loss between predicted and target values. Learning rate scheduling is implemented via a step decay schedule to facilitate convergence over time.

3.3 MAPPO Implementation Strategy

For the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm, we adopt a trust-region approach through PPO clipping. Each batch undergoes multiple update epochs, which allows the policy to be refined iteratively. The probability ratio between the new and old policy is clipped to prevent excessively large updates, ensuring stable learning.

3.4 Experiments

3.4.1 Single-layout self-play

We started by comparing the two algorithms in 4 different layouts of the benchmark. Each algorithm was trained using the optimal hyperparameters found, while keeping the same total episodes of training and the same actor and critic networks.

3.4.2 Layout generalization

We then extended the evaluation on multiple layouts using an extended version of the Overcooked environment, which randomly allowed training across different layouts. The process started with a combination of two simple layouts, and progressively included more complex configurations, culminating in a final experiment with seven layouts trained jointly. This setup was created to assess the ability of each algorithm in generalizing the instructions across different spatial patterns with the final goal to generate a sort of all-around agent. As a final test we evaluated the performances of the agents on unseen layouts.

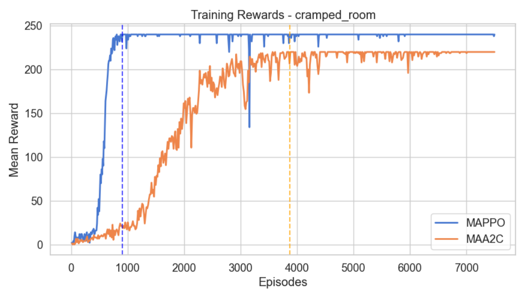
3.4.3 Cross-Algorithm training

To evaluate the robustness and flexibility of our agents, we introduced a cross-algorithm evaluation. In this setting, two agents trained with different MARL algorithms, one using MAPPO and the other using MAA2C, are paired together in a shared environment. The goal is to assess whether heterogeneous agents can cooperate effectively, even though they were never trained together. The procedure proceeds as follows: we first load the actor networks of the two agents, trained either on single or multiple layouts. For each episode, the environment is reset, and both agents select actions based on their individual observations. Optionally, their positions can be randomized at the start of each episode to further evaluate generalization. The environment then progresses one step using the chosen actions, with rewards accumulated until the episode concludes. This process is repeated across multiple episodes and layouts to assess cross-algorithm cooperation.

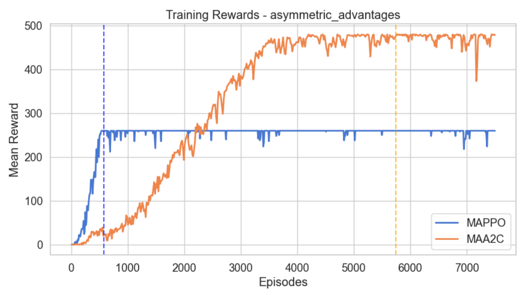
4 Results

4.1 Single-layout self-play

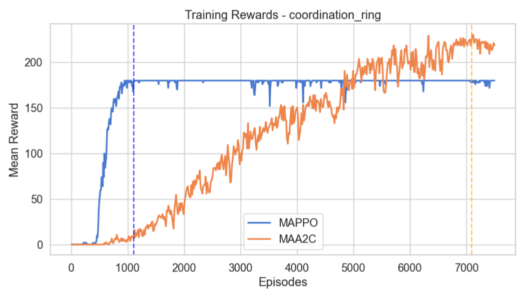
We started by training both algorithm on single layouts for a total of 7500 episodes. In line with the expectations, MAPPO is faster in reaching a plateau and the reward variations in training are more controlled. This behavior can be attributed to the clipping mechanism in its policy update, which effectively constrains policy divergence and stabilizes training. Interestingly, however, the results obtained in MAA2C for two of the four layouts outperforms MAPPO and in asymmetric advantage reaches almost the perfect score, a results consistent with the tests executed in [1]. These results suggest that while MAPPO generally provides faster and more stable training, MAA2C may benefit from extended exploration in environments where coordination and asymmetry play a larger role. Overall, this experiment show how there is no general best algorithm between the two (except for stability where MAPPO dominates), careful tuning and sufficient exploration are therefore crucial to allow each algorithm to reach its optimal performance.



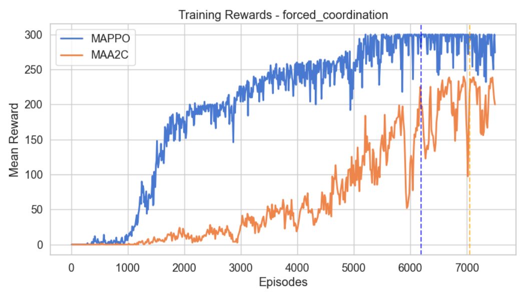
(a) Cramped



(b) Asymmetric



(c) Coordination



(d) Forced

Figure 1: Self-play training reward histories comparison

Table 1: Performance comparison between MAPPO and MAA2C across Overcooked layouts. Green cells indicate better performance.

Layout	**Algorithm**	**Max Reward**	**Episodes to Plateau**	**Time to Plateau (s)**
Cramped Room	MAPPO	240.00	910	204.89
	MAA2C	220.00	3870	655.30
Asymmetric Adv.	MAPPO	260.00	580	152.37
	MAA2C	479.47	5745	1091.39
Coordination Ring	MAPPO	180.00	1110	279.18
	MAA2C	226.13	7095	1281.35
Forced Coord.	MAPPO	300.00	6190	1454.47
	MAA2C	235.47	7050	1193.52

4.2 Multi-layout self-play

We trained both algorithms on different layout combinations, gradually increasing the complexity. The results indicate that our MAA2C implementation generalizes better across layouts and is the only algorithm capable of handling unseen layouts. The table below shows the layout groups used for simultaneous training and the corresponding performance achieved by each algorithm.

Table 2: Performance comparison in multi-layouts settings. Each test performed on 5 episodes.

Test layouts	**MAPPO**		**MAA2C**	
	Reward	Soups	Reward	Soups
Training Group 1: Cramped Room + Asymmetric Advantages				
cramped_room	200	10.0	200	10.0
asymmetric_advantages	260	13.0	260	13.0
Training Group 2: Coordination Ring + Forced Coordination				
coordination_ring	180	9.0	160	8.0
forced_coordination	80	4.0	88	4.4
Training Group 3: Centre Pots + Centre Objects				
centre_pots	200	10.0	176	8.8
centre_objects	0	0	136	6.8
Training Group 4: Multi-layout Mix				
cramped_room	220	11.0	220	11.0
coordination_ring	0	0	156	7.8
forced_coordination	0	0	76	3.0
large_room	200	10	176	8.8
five_by_five	152	7.6	220	11
centre_pots	188	9.4	180	9.0
centre_objects	0	0	164	8.2
simple_o (unseen before)	0	0	8	0.40
simple_o_t (unseen before)	0	0	12	1.80
simple_tomato (unseen before)	0	0	16	2.40
asy_adv (unseen before)	0	0	0	0

4.3 Cross-algorithm multi-layout generalization

We evaluated the interaction between two agents, each trained independently on the same single or multiple layouts, without any knowledge of the other agent's training. By pairing the agents in shared environments, we assessed how well each algorithm adapts to the presence of a differently trained teammate, capturing both cooperative behavior and robustness. The experiments also included randomly shuffles of which agent controls which player in which position in each episode, and vice versa; for example in episode 1 Agent1 might be Player 0, Agent2 Player 1; in episode 2 Agent2 might be Player 0 and Agent1 Player 1. The results shows the algorithms' ability to cooperate effectively, revealing interesting insights into their robustness and adaptability in mixed-algorithms agents settings.

Table 3: Cross-algorithm test results for single and multi-layouts agents, with fixed and randomized agent role. Each tested on 5 episodes.

Layout Type	**Fixed**		**Random**	
	Reward	**Soups**	**Reward**	**Soups**
Single: Cramped room	16	0.8	12	0.6
Single: Asymm. Adv.	160	8.0	28	1.4
Single: Coord. Ring	56	2.8	20	1.0
Single: Forced Coord.	116	5.8	36	1.8
Group 4: Cramped	220	11.0	20	1.0
Group 4: Large Room	188	9.4	12	0.6

5 Conclusion

This work presented a systematic comparison between two state of the art on-policy multi-agent reinforcement learning algorithms, MAPPO and MAA2C, within the Overcooked-AI benchmark. By implementing both algorithms from scratch under a shared CTDE framework, we ensured a fair and transparent evaluation across three axes: absolute performance, generalization, and cross-algorithm cooperation.

Our findings highlight a fundamental trade-off. MAPPO exhibits faster convergence and greater stability, owing to its clipped policy updates, making it more reliable in environments where rapid learning and stable performance are critical. However, MAA2C demonstrates superior performance in asymmetric or highly coordination-dependent layouts, and more importantly, achieves stronger generalization to unseen environments when trained on different layout groups. This suggests that exploration and flexible value estimation play a decisive role when transferring policies across heterogeneous environments.

The cross-algorithm experiments revealed that heterogeneous agents, trained independently with different learning paradigms, are capable of meaningful cooperation. The results indicate that cross-algorithm cooperation is feasible, particularly under fixed role assignments. Under random role assignments the task become much harder but the algorithms still produce solutions. This opens promising avenues for real-world MARL applications, where agents developed with distinct learning processes or by different teams may still need to collaborate effectively.

In summary, there is no universally superior algorithm: MAPPO is preferable when stability and training efficiency are paramount, whereas MAA2C is better suited when generalization and adaptability to new environments are critical. Our results also demonstrate the potential of cross-algorithm cooperation, pointing to future research on communication mechanisms, joint training protocols, robustness and adaptive teaming strategies between heterogeneous agents.

References

[1] George Papadopoulos et al. "An Extended Benchmarking of Multi-Agent Reinforcement Learning Algorithms in Complex Fully Cooperative Tasks". In: *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '25. Detroit, MI, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2025, pp. 1613–1622. ISBN: 9798400714269.

[2] Chao Yu et al. "The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games". In: *CoRR* abs/2103.01955 (2021). arXiv: 2103.01955. URL: https://arxiv.org/abs/2103.01955.