Tests on Overcooked-AI: a Multi-Agent Reinforcement Learning environment

Tommaso Perniola
University of Bologna
`tommaso.perniola@studio.unibo.it`

Abstract

In this paper, we present a series of experiments conducted in the Overcooked-AI environment, a high-level benchmark designed to test coordination and cooperation in multi-agent reinforcement learning (MARL). Our objective is to demonstrate that with the correct architectural and training strategies, agents can learn effective cooperative behaviors, generalize across different kitchen layouts, and adapt to sparse and delayed rewards. We leverage a modified implementation of the MAPPO (Multi-Agent Proximal Policy Optimization) algorithm, incorporating reward scheduling, layout curriculum, partner sampling and entropy regulation.

1 Introduction

Cooperative behavior is a central challenge in MARL, particularly in sparse-reward settings such as Overcooked-AI [2], where two agents coordinate to deliver onion soups. Each layout imposes different spatial and coordination constraints, making generalization across environments a key objective. We evaluate whether MAPPO can provide transferable, high-performing cooperative policies in this domain (reward scheme in Table 1).

Event	Reward
Ingredients placement	+3
Picking a dish	+3
Picking a soup	+5
Delivering soup	+20

Table 1: Overcooked-AI rewards.

2 Background

PPO [4] is an on-policy RL algorithm based on clipped policy gradients. MAPPO [7] extends PPO to multi-agent domains via centralized training and decentralized execution (CTDE) [1].

2.1 Why MAPPO

MAPPO employs shared actor and critic networks under the Centralized Training with Decentralized Execution (CTDE) paradigm [1]. A centralized critic receives the joint state of all agents, reducing non-stationarity by providing a stable learning signal for the system as a whole. At execution, each agent acts independently with a shared policy based only on its local observation, which addresses credit assignment while keeping scalability linear in the number of agents. This design balances stability, coordination, and efficiency in cooperative MARL.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

3 Implementation

We implemented a TensorFlow-based version of MAPPO, using a shared actor–critic architecture with centralized training and decentralized execution. To cope with sparse rewards, we integrated reward shaping and entropy regularization. Training focused on five layouts (`cramped_room`, `asymmetric_advantages`, `bottleneck`, `counter_circuit`, and `coordination_ring`), while generalization was encouraged through curriculum learning with layout switching and partner sampling. Evaluation also included an unseen layout, `forced_coordination`.

3.1 Indexing Issue

Role embeddings (index vectors) are often used to specialize agents, but in Overcooked spawn positions are randomized, so agent indices are unstable. Hence, two alternative approaches are possible: (i) *Symmetric policies*, fully parameter-shared without identity; (ii) *Contextual observations*, where role-related features are dynamically extracted from the state (e.g., as in [2]). Both aim to preserve coordination without relying on fixed agent identities.

3.2 Critic Architectures

We evaluated: (i) *Single-head critic*, simpler but blending agent contributions; (ii) *Multi-head critic*, one output per agent, improving credit assignment.

3.3 Implementation Basis and References

Our PPO and MAPPO implementations are inspired by established open-source baselines. Specifically:

The initial PPO structure, including GAE, clipping, and entropy regularization, follows implementation ideas from `PPO-for-Beginners` by Eric Yang[1], which provides a clean educational baseline for single-agent PPO.

The multi-agent extensions, including centralized critic, shared actor, and CTDE structure, are adapted from the `on-policy` MAPPO benchmark repository[2], which offers scalable implementations for MARL research.

3.4 Policy Optimization

The actor is trained with the PPO clipped objective:

$$\mathcal{L}^{\text{CLIP}} = \mathbb{E}_t \left[\min \left(r_t \hat{A}_t,\, \text{clip}(r_t, 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right],$$

augmented with an entropy bonus

$$\mathcal{L}^{\text{ENT}} = -\beta\, \mathbb{E}_t \left[\sum_a \pi(a|s_t) \log \pi(a|s_t) \right],$$

and combined critic loss

$$\mathcal{L}^{\text{Critic}} = \tfrac{1}{N} \sum_t \left(V(s_t) - \hat{R}_t \right)^2.$$

GAE [5] was used for advantage estimation.

3.5 Reward Shaping & Stability

Shaped intermediate rewards were decayed over training to shift focus toward deliveries. Gradient clipping safeguarded stability under multi-agent variance, while advantages were normalized across batches to reduce variance and improve the stability of policy updates.

[1]`https://github.com/ericyangyu/PPO-for-Beginners`

[2]`https://github.com/marlbenchmark/on-policy`

3.6 Dynamic Layout Switching

Training progressed in phases: (i) only `cramped_room`; (ii) probabilistic interleaving of layouts after mastery; (iii) uniform sampling (not reached). This curriculum promoted gradual generalization.

3.7 Population-Based Training

To improve generalization, we adopted *Population-Based Training* (PBT), a technique used in Overcooked [2] which has origin in other domains [3]. During training, agents interact with a pool of partners representing diverse coordination styles rather than relying solely on *self-play*. While self-play ensures stable convergence and a degree of optimality, it may often limits robustness; PBT can mitigate this by exposing agents to varied behaviors, reducing overfitting and improving adaptability to unseen teammates.

3.8 Our Models

We compare two models: (i) **Self-play**, symmetric actor + multi-head critic, trained exclusively via self-play; (ii) **Partner-sampling**, identical architecture but trained with PBT, then fine-tuned with self-play.

3.9 Hyperparameters

Table 2 summarizes key hyperparameters, with fine-tuning values after the semicolon.

Param	**SP+SP**	**PBT+SP**
Rollouts	180; 60	180; 50
Episodes	8	8
Steps/ep	400	400
Epochs/up	6	6
γ	0.99; 0.995	0.99; 0.995
λ	0.90; 0.98	0.95; 0.98
Batch size	64	64
Clip range	0.2	0.2
Entropy coeff.	0.01; 0.02	0.01; 0.02
Actor LR	$5 \cdot 10^{-4}$	$2 \cdot 10^{-4}$
Critic LR	$2 \cdot 10^{-4}$	$1 \cdot 10^{-4}$

Table 2: Key hyperparameters for self-play (SP) and partner-sampling (PBT) models.

4 Results

The results obtained were overall satisfactory. Both models learned quite well the behavioral patterns required by training layout and surprisingly reached the same level of generalization on the unseen layout, but still quite poor. The training of both the models was conducted for a total of 240 rollouts each: 180 during the initial training phase and an additional 60 during fine-tuning, amounting to approximately 1440 epochs and 1920 episodes.

4.1 Training Phases

As shown in Figure 1, that also shows the overlapped fine-tuning performances, the agent successfully learned coordinated behavior in the `cramped_room` layout, reaching a peak average return of approximately 150. However, performance began to slightly decline around rollout number 30—coinciding with the transition to `phase_2` of the layout curriculum, where the training distribution incorporated harder layouts such as `asymmetric_advantages` and `bottleneck`.

The same behavior can be observed also in figure 2, but the agent remains a bit longer in the `phase_1` because we restricted the criteria to switch among phases (the average reward and the mean deliveries now should be consistent across several rollouts).

Notably, the agents were not able to meet the performance threshold required to enter `phase_3`, which involves uniform sampling across all layouts. This suggests that although the policy generalizes reasonably across some layouts, its robustness in more constrained environments is still limited.

The overall training results are shown in table 3.

Layout	SP+SP	PBT+SP
cramped_room	139.2	127.0
asymmetric_advantages	40.81	54.68
bottleneck	10.92	7.840
counter_circuit	0.485	0.515
coordination_ring	17.26	15.56

Table 3: Average return per rollout across layouts for Self-Play and Population-Based Training.

4.2 Evaluation Performance

While training curves provide insight into learning dynamics and curriculum progression, inference results are necessary to evaluate the actual cooperative competence of the trained agents. Inference evaluation removes training-specific aids such as reward shaping and exploration noise, ensuring a fair and realistic comparison between methods.

From the inference results reported in Table 4, it is evident that the `SP+SP` configuration generally outperforms `PBT+SP` across most layouts. In particular, on `cramped_room` and `asymmetric_advantages`, `SP+SP` achieves substantially higher mean rewards (131.4 vs. 98.0 and 81.0 vs. 67.4, respectively), indicating that sequential self-play fosters more stable coordination patterns in relatively structured environments. The difference is even more striking in the `bottleneck` layout, where `SP+SP` achieves nearly five times the reward of `PBT+SP` (37.6 vs. 7.8), highlighting the importance of consistent partner behavior in highly constrained scenarios. On `coordination_ring`, `SP+SP` also shows an advantage, albeit with higher variance, suggesting unstable but occasionally effective coordination. The only case where both methods perform equally poorly is `counter_circuit`, where rewards remain close to zero, indicating that neither approach generalized well to this challenging configuration. Overall, these results confirm that `SP+SP` tends to produce more reliable cooperation at inference time, while `PBT+SP` may introduce noise that hampers coordination in difficult layouts.

Layout	PBT+SP	SP+SP
cramped_room	98.00 ± 18.45	131.40 ± 22.90
asymmetric_advantages	67.40 ± 27.58	81.00 ± 34.69
bottleneck	7.80 ± 2.64	37.60 ± 9.13
counter_circuit	2.20 ± 3.12	2.20 ± 3.12
coordination_ring	19.40 ± 4.41	27.80 ± 16.04

Table 4: Inference results (mean $\pm$ standard deviation) for different layouts.

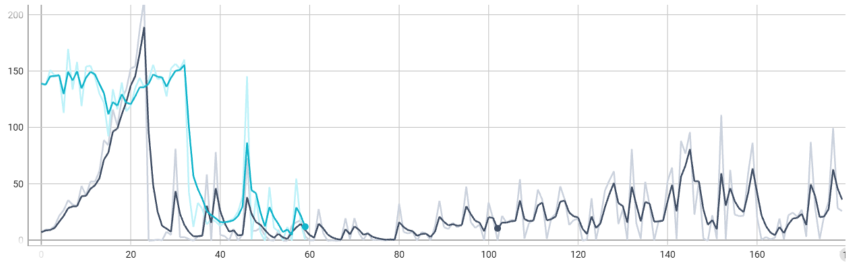

Figure 1: SP+SP model: average return during training across different layouts.

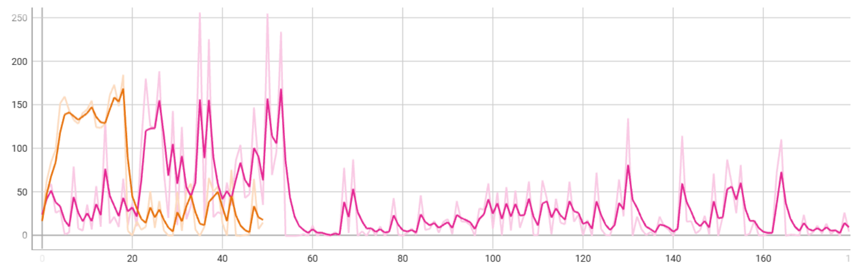

Figure 2: PBT+SP model: average return during training across different layouts.

4.2.1 Generalization abilities

As table 5 shows, both the self-play and partner-sampling (PBT) models achieved identical performance when evaluated on the unseen `forced_coordination` layout. This suggests that, despite their different training regimes, both models reached a comparable level of generalization. PBT introduces extra variance because the agent faces multiple partner behaviors, that leads to broaden experience and can improve robustness, but it also slows down convergence. With limited training time, the benefit of diversity may not fully materialize.

Model	**Mean Reward**	**Std Dev**
Self-Play (SP)	6.850	3.320
Population-Based Training (PBT)	6.850	3.540

Table 5: Evaluation over 20 episodes on the unseen `forced_coordination` layout.

5 Conclusion

In this work, we applied Multi-Agent Proximal Policy Optimization (MAPPO) to the Overcooked-AI environment, a cooperative multi-agent benchmark characterized by sparse rewards and strong coordination requirements. Our implementation integrated several training enhancements, including centralized training with decentralized execution, curriculum-based layout switching, reward shaping, and entropy scheduling, which together enabled stable learning in otherwise challenging settings.

The experimental results indicate that both `SP+SP` and `PBT+SP` agents successfully acquired cooperative behaviors in the training layouts, achieving strong performance in `cramped_room` and moderate results in `asymmetric_advantages`. However, the `bottleneck` and `coordination_ring` layouts remained difficult, with `SP+SP` agents showing more reliable coordination at inference time. In particular, inference evaluation revealed that `SP+SP` consistently outperformed `PBT+SP`, achieving substantially higher rewards and lower variance in most layouts. This confirms that stable partner behavior during training facilitates more effective coordination, whereas partner sampling introduces variance that can hinder performance in highly constrained environments. Notably, both approaches failed to achieve significant progress in the `counter_circuit` layout, underlining the need for stronger exploration or alternative training curricula.

Generalization to unseen layouts remained limited: both methods performed equally on the `forced_coordination` map, highlighting that despite their differing training regimes, the resulting policies converged to a comparable level of robustness. While partner sampling theoretically promotes broader adaptability through exposure to diverse partner strategies, its benefits did not fully emerge within the limited training budget used in this study.

Overall, these findings demonstrate the effectiveness of MAPPO in cooperative multi-agent tasks but also underline its limitations in terms of robustness and generalization.

6 Future Work

Several directions remain open for improving coordination and generalization in Overcooked-AI. First, extending the overall training budget and refining the curriculum to better emphasize constrained layouts could strengthen policy robustness. Beyond this, complementary techniques such as contextual role representations and explicit role conditioning offer promising ways to introduce inductive bias, enrich agent modeling, and support more adaptive cooperation in multi-agent scenarios.

6.1 Contextual Roles

One promising avenue is to augment each agent's observation with contextual information that disambiguates agent identities. This provides a lightweight mechanism for addressing the indexing issue, enabling agents to extract role-aware features and develop more specialized behavior. In this work, we partially implemented an architecture designed to support such contextual information, but it has not yet been trained or systematically evaluated. Future experiments will be needed to assess its effectiveness in promoting stable coordination.

6.2 Role Conditioning

Another key direction is the explicit incorporation of role embeddings into the agents' observations, as suggested by prior work [6]. In the current setup, agents share a policy and lack any identity signal, which may hinder role differentiation—especially when agent order is permuted at every reset. In preliminary experiments, we introduced an 8-dimensional role vector for each agent during training while maintaining dynamic agent indexing. At inference time, replacing this vector with a zero vector (ablation) resulted in a marked performance drop, particularly in complex layouts such as `bottleneck`. This indicates that role information was crucial for specialization and coordination. Future implementations should therefore adopt role conditioning more systematically to promote stable role differentiation and mitigate symmetric coordination failures.

References

[1] Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning, 2024. URL `https://arxiv.org/abs/2409.03052`.

[2] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination, 2020. URL `https://arxiv.org/abs/1910.05789`.

[3] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks, 2017. URL `https://arxiv.org/abs/1711.09846`.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

[5] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL `https://arxiv.org/abs/1506.02438`.

[6] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles, 2020. URL `https://arxiv.org/abs/2003.08039`.

[7] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022. URL `https://arxiv.org/abs/2103.01955`.