# 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, and entropy regulation.

## 1 Introduction

Cooperative behavior remains one of the key challenges in Multi-Agent Reinforcement Learning (MARL), especially in environments characterized by sparse and delayed rewards. One such environment is Overcooked-AI [2], where two agents must coordinate in a shared kitchen to prepare and deliver onion soups across various layouts of increasing difficulty.

Each layout introduces unique spatial and coordination constraints, pushing the agents to adopt versatile and generalized cooperative policies. Our goal is to investigate whether current state-of-the-art MARL algorithms—specifically MAPPO—can be effectively employed in this setting to learn transferable and high-performing behaviors across different layouts.

The Overcooked-AI environment assigns sparse rewards, as shown in Table 1, making the credit assignment problem non-trivial and requiring careful exploration and scheduling strategies.

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

Table 1: Reward values for different events in the Overcooked-AI environment.

## 2 Background: PPO and MAPPO

Proximal Policy Optimization (PPO) [3] is an on-policy reinforcement learning algorithm that has demonstrated strong empirical performance in various domains. MAPPO [6] extends PPO to multi-agent settings by integrating centralized training with decentralized execution (CTDE), a paradigm particularly suited for cooperative environments.

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

The MAPPO formulation assumes shared actor and critic networks among agents. The centralized critic receives the joint state of all agents, while each agent acts based on its own local observation.

# 3 Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) considers multiple interacting agents learning concurrently in a shared environment. In cooperative MARL, the agents share a common reward signal and are required to learn policies that contribute to a joint objective.

A particularly successful strategy for cooperative MARL is Centralized Training with Decentralized Execution (CTDE) [1]. Under CTDE:

- **Training is centralized:** the critic is fully observable and receives the global state composed of joint observations from all agents.
- **Execution is decentralized:** each agent acts independently based on its own observation using a shared policy.

# 4 Implementation

We implemented a modified version of the MAPPO algorithm in TensorFlow, available in `utils.py`. The agent uses a shared actor-critic architecture with centralized training and decentralized execution. To improve learning under sparse rewards, we introduced reward shaping, entropy regularization, and a curriculum learning strategy via layout switching. We carry on a training process that only considers three layouts, `cramped_room`, `asymmetric_advantages` and `bottleneck`. We will also test performances on unseen layouts.

## 4.1 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.

Our codebase reworks these core ideas into a TensorFlow-based, Overcooked-AI–compatible architecture, integrating reward scheduling and layout-specific curriculum learning as described in the following sections.

## 4.2 Policy Optimization with PPO

The actor is trained using the clipped surrogate loss defined in Proximal Policy Optimization (PPO) [3]. Given an old policy $\pi_{\theta_{\text{old}}}$ and the current policy $\pi_\theta$, the objective is:

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

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio,
- $\hat{A}_t$ is the advantage estimate,

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

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

- $\epsilon$ is the PPO clipping threshold (initialized to 0.15 and decayed linearly to 0.05 during fine-tuning).

This objective prevents large deviations from the previous policy by penalizing updates where $r_t$ departs too much from 1.

### 4.3 Hyperparameters

Table 2 summarizes the key hyperparameters used during training, including values overridden during the fine-tuning phase.

| Parameter | Base Value | Fine-Tuned Value |
|---|---|---|
| Timesteps per batch | 4096 | 4096 |
| Steps per episode | 400 | 400 |
| Discount factor $\gamma$ | 0.99 | 0.99 |
| GAE parameter $\lambda$ | 0.90 | 0.90 |
| Epochs per update | 4 | 6 |
| Batch size | 512 | 256 |
| Initial PPO clip range | 0.15 | 0.15 |
| Final PPO clip range | 0.05 | 0.05 |
| Initial entropy coeff. | 0.01 | 0.03 |
| Actor learning rate | $5 \times 10^{-4}$ | $2 \times 10^{-4}$ |
| Critic learning rate | $2 \times 10^{-4}$ | $1 \times 10^{-4}$ |

Table 2: Comparison of base and fine-tuned hyperparameters used during MAPPO training.

### 4.4 Entropy Regularization

To promote exploration, an entropy bonus is added to the actor loss:

$$\mathcal{L}^{\text{ENT}}(\theta) = -\beta \cdot \mathbb{E}t\left[\mathcal{H}[\pi\theta(\cdot|s_t)]\right] = -\beta \cdot \mathbb{E}t\left[\sum_a \pi\theta(a|s_t) \log \pi_\theta(a|s_t)\right], \tag{2}$$

where:

- $\mathcal{H}$ is the policy entropy,
- $\beta$ is the entropy coefficient, linearly annealed from 0.03 to 0 during fine-tuning to reduce exploration over time.

The final actor loss becomes:

$$\mathcal{L}^{\text{Actor}} = \mathcal{L}^{\text{CLIP}} + \mathcal{L}^{\text{ENT}}. \tag{3}$$

### 4.5 Advantage Estimation

Advantages are computed using Generalized Advantage Estimation (GAE) [4]:

$$\hat{A}t = \delta_t + (\gamma\lambda)\delta t + 1 + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{4}$$

with:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{5}$$

### 4.6 Critic Loss

The critic is trained to minimize the squared error between estimated returns and value predictions:

$$\mathcal{L}^{\text{Critic}} = \frac{1}{N} \sum_{t=1}^{N} \left( V_\theta(s_t) - \hat{R}_t \right)^2, \tag{6}$$

where $\hat{R}_t = \hat{A}_t + V(s_t)$.

### 4.7 Reward Shaping

To accelerate learning in the presence of sparse rewards, we implemented a reward shaping mechanism. Intermediate rewards (such as placing ingredients or picking up dishes) were weighted using a time-dependent scaling factor that gradually decays during training. This decay begins once agents demonstrate sufficient proficiency on the initial layout. The goal is to incentivize exploration early on while ensuring that, over time, the agents focus on achieving the sparse but meaningful reward signal—namely, delivering dishes. This scheduling strategy was crucial to transitioning agents from local, shaped behaviors to globally coordinated ones that generalize across layouts.

### 4.8 Gradient Clipping

To ensure stability during training—especially under the variability introduced by multi-agent interactions—we applied gradient clipping to both the actor and critic networks. This technique constrains the magnitude of gradient updates during backpropagation, preventing sudden and destabilizing shifts in the policy or value estimates. Specifically, each gradient is individually scaled if its norm exceeds a predefined threshold. This safeguard proved effective in maintaining smooth learning dynamics and was particularly beneficial when training on more complex layouts

### 4.9 Dynamic Layout Switching

Training progresses through three phases using `dynamic_layout_switching()`:

1. Phase 1: only `cramped_room`.
2. Phase 2: probabilistic interleaving of layouts after mastering cramped room.
3. Phase 3: uniform random sampling when all layouts reach target thresholds.

This progressive exposure helps the policy generalize across layouts.

### 4.10 Fine-Tuning and Curriculum Adaptation (Summary)

During fine-tuning, we progressively reduce exploration and policy update aggressiveness to improve generalization. This is achieved by linearly decaying the PPO clipping threshold ($\epsilon$) and entropy coefficient ($\beta$) over 200 rollouts. Simultaneously, we use a specialized layout sampler that shifts training focus from easier (e.g., `cramped_room`) to more complex layouts (e.g., `bottleneck`), enabling gradual adaptation and robust policy generalization.

## 5 Results

The results obtained were overall satisfactory. Training was conducted for a total of 500 rollouts: 300 during the initial training phase and an additional 200 during fine-tuning, amounting to approximately 2400 epochs. Each rollout consisted of 4096 timesteps per agent.

### 5.1 Initial Training Phase

As shown in Figure 1, the agent successfully learned coordinated behavior in the `cramped_room` layout, reaching a peak average return of approximately 70. However, performance began to slightly decline around 700k timesteps—coinciding with the transition to `phase_2` of the layout curriculum,

where the training distribution incorporated harder layouts such as `asymmetric_advantages` and `bottleneck`.

Notably, the agent was 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.

**Evaluation Performance**

During evaluation (i.e., deterministic inference without exploration), the trained policy achieved the following average returns, over 20 episodes:

- `cramped_room`: 51
- `asymmetric_advantages`: 160
- `bottleneck`: 3

These results indicate a strong performance in the `asymmetric_advantages` layout, which provides more maneuvering space and relatively simpler coordination opportunities. However, upon inspecting the generated rollout visualizations (e.g., in GIF format), we observed that the high return was largely due to one agent consistently performing the majority of the tasks—such as picking, cooking, and delivering—while the other agent exhibited suboptimal or idle behavior. This suggests that the policy learned an imbalanced but effective strategy in which cooperation was minimal and one agent effectively carried the load. While this may suffice in some layouts, it highlights a lack of robust coordination and balanced role allocation between agents. On the other hand, the poor performance in `bottleneck` highlights the difficulty in learning fine-grained coordination in narrow, constrained environments. This suggests that either additional curriculum shaping or specialized coordination priors may be required for such layouts.

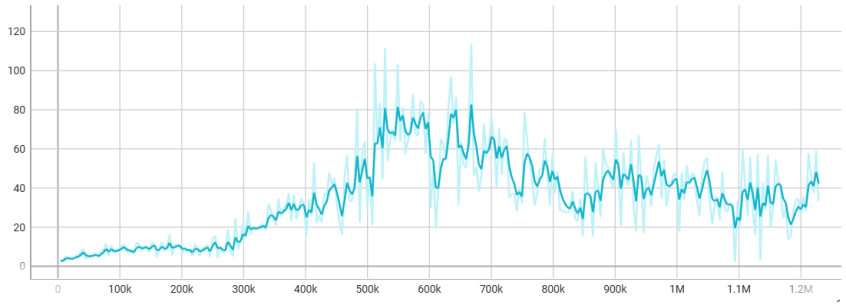


Figure 1: Average return during training across different layouts.

## 5.2 Fine-tuning

The second training phase (fine–tune) yielded a marked performance lift on *all* three layouts when compared with the policy that ended Phase 1. Indeed, as we can see from figure 2, the average among is quite higher than the previous one.

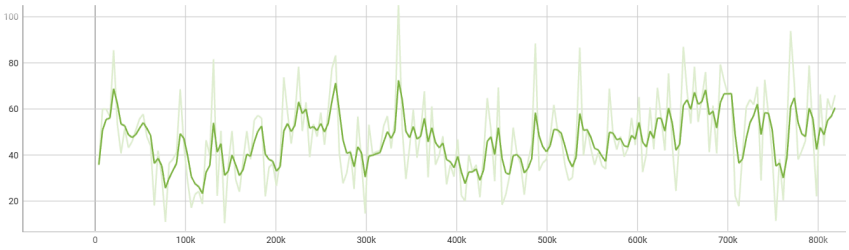


Figure 2: Average return during training across different layouts.

Using the mean episodic returns logged during training, *cramped_room* increased from **33.7** to roughly **52–54**, *asymmetric_advantages* from **38.9** to **44.8**, and *bottleneck* remains the same. Absolute gains are therefore similar (around 10-15 reward points) while relative gains are largest on the asymmetric layout, suggesting that the higher-entropy, curriculum-guided fine-tune phase especially helped the helper–leader role specialization required there. Variance remains **high** but the minimum return per rollout rose substantially, indicating a more robust policy. These trends confirm that the entropy schedule, learning-rate bump, and reward-shaping floors introduced in Phase 2 succeeded in translating exploratory behaviour into stable return improvements without sacrificing the progress made during the initial phase.

# 6 Conclusion

In this work, we explored the application of Multi-Agent Proximal Policy Optimization (MAPPO) to the Overcooked-AI environment, a challenging cooperative MARL benchmark with sparse rewards and high coordination demands. By incorporating centralized training with decentralized execution, curriculum-based layout switching, reward shaping, and entropy scheduling, our implementation demonstrated promising performance across multiple layouts.

The results show that agents are capable of learning effective cooperative strategies, particularly in spacious layouts like `asymmetric_advantages`. However, the bottleneck layout remained challenging, highlighting limitations in coordination under spatial constraints. The fine-tuning phase contributed significantly to robustness and generalization, with consistent improvements across all layouts. Unfortunately, performances on unseen layouts like `forced_coordination` are poor, delivering very bad results (around 3–6 average reward), highlighting how generalization ability can be further enhanced.

Overall, our approach validates the effectiveness of MAPPO in structured MARL environments, and opens the door for further enhancements to drive specialization and coordination in more complex settings, as outlined in the next section.

# 7 Future Work

Several enhancements were identified but not fully explored in this work. Among them, two stand out for their potential impact on coordination and generalization.

## 7.1 Role Conditioning

A key direction is incorporating explicit role embeddings into the agent observations, as observed in [5]. In a preliminary experiment, we assigned each agent a fixed 8-dimensional role vector during training. At inference time, replacing this vector with a zero vector (ablation) led to a significant drop in performance—particularly in complex layouts like `bottleneck`. This suggests that roles were essential for agents to specialize and coordinate effectively. Future implementations should adopt role conditioning to promote stable role differentiation and reduce symmetric failures.

## 7.2 Agent Indexing

Currently, agents share a policy and receive no identity signal. This can hinder role learning, especially when agent order varies. Including an agent index (e.g., one-hot or learned ID) could improve behavior consistency and allow agents to specialize even when observations are identical. Combined with role conditioning, this could improve coordination and generalization across layouts.

Overall, both mechanisms would add valuable inductive bias, enabling richer agent modeling and better adaptation in multi-agent scenarios.

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

[4] 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.

[5] 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.

[6] 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.