MEAL: A Benchmark for Continual Multi-Agent Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Benchmarks play a crucial role in the development and analysis of reinforcement learning (RL) algorithms, with environment availability strongly impacting research. One particularly underexplored intersection is continual learning (CL) in cooperative multi-agent settings. To remedy this, we introduce **MEAL** (**M**ulti-agent **E**nvironments for **A**daptive **L**earning), the first benchmark tailored for continual multi-agent reinforcement learning (CMARL). Existing CL benchmarks run environments on the CPU, leading to computational bottlenecks and limiting the length of task sequences. MEAL leverages JAX for GPU acceleration, enabling continual learning across sequences of up to 100 tasks on a standard desktop PC within a few hours. Evaluating popular CL and MARL methods reveals that naïvely combining them fails to preserve network plasticity or prevent catastrophic forgetting of cooperative behaviors.

1 Introduction

Continual RL has recently attracted growing interest [12, 6, 7, 10], but remains largely unexplored in multi-agent settings [31, 32]. Combining the two introduces unique challenges. In cooperative environments, agents must establish implicit conventions or roles for effective coordination [26]. As tasks or dynamics shift, these conventions can break down, making continual MARL significantly harder than its single-agent counterpart. Forgetting past partners or roles can cause the entire team to fail, amplifying the impact of catastrophic forgetting through inter-agent dependencies. Unlike traditional MARL, CMARL involves non-stationarity not only due to the presence of other learning agents, but also from a shifting task distribution [32]. This dual pressure demands agents that can generalize, adapt, and transfer knowledge more robustly than in standard single-agent continual or static multi-agent settings. This setting is relevant for applications where agents must adapt to evolving environments without forgetting prior coordination strategies. For instance, autonomous vehicles must navigate unseen roads, adapt to new traffic regulations, and interact with unfamiliar human drivers, while occasionally coordinating with other AVs. Similarly, warehouse robots deployed in a new facility must quickly adapt to different layouts and workflows, while preserving established collaborative behaviors.

To analyze how current methods handle the interplay between continual learning and multi-agent coordination, and to drive progress in this domain, we introduce **MEAL**, the first benchmark for CMARL. To the best of our knowledge, MEAL[1] is also the first continual RL library to leverage JAX for end-to-end GPU acceleration. Traditional CPU-based benchmarks are limited to short sequences (5–15 tasks) due to low environment throughput and task diversity [25, 21, 28], making them ill-suited for the computational demands of cooperative continual learning. MEAL's end-to-end

[1]The code and environments are accessible on Anonymous GitHub.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

JAX pipeline removes this barrier, enabling training on up to 100 tasks within a few hours on a single desktop GPU. This unlocks new research directions for scalable, cooperative continual learning in resource-constrained settings.

MEAL is built on Overcooked [5], where agents are known to exploit spurious correlations in fixed layouts, leading to poor generalization even under minor changes [15]. As a result, a task sequence with small layout variations is sufficient to pose a challenging continual learning problem. Successfully learning across such a sequence requires agents to avoid layout-specific overfitting and instead develop robust, transferable coordination strategies. Sequential Overcooked layouts thus offer a controlled and reproducible way to introduce meaningful task diversity.

The **contributions** of our work are three-fold. (1) We introduce MEAL, the first CMARL benchmark, consisting of handcrafted and procedurally generated Overcooked environments spanning three difficulty levels. (2) We leverage JAX to build the first end-to-end GPU-accelerated task sequences for continual RL, enabling efficient training on low-budget setups. (3) We implement six popular CL methods in JAX and evaluate them in various MEALs, revealing key shortcomings in retaining cooperative behaviors and adapting to shifting roles across tasks.

2 Related Work

Continual Reinforcement Learning (CRL) Continual reinforcement learning studies how agents can learn sequentially from a stream of tasks without forgetting previous knowledge. A wide range of methods have been proposed, including regularization-based approaches such as EWC [14], SI [33], and MAS [2]; architectural strategies such as PackNet [18]; and replay-based methods like RePR [3]. More recent works focus on scalability [12], memory efficiency [7], and stability during training [6]. However, these methods are almost exclusively developed for single-agent settings, and their behavior under multi-agent coordination remains largely unexplored.

Multi-Agent Reinforcement Learning (MARL) In MARL, multiple agents learn to act in a shared environment, often with partial observability and either cooperative or competitive goals [13, 20]. A major focus has been on cooperative settings, where agents share a reward function and must learn to coordinate [17, 11]. Popular algorithms include IPPO [8], VDN [27], QMIX [22], and MAPPO [30]. Many benchmarks assume a static environment and fixed task, making them unsuitable for studying continual learning or transfer across environments.

Benchmarks Standard CRL benchmarks include Continual World [29], COOM [28], and CORA [21]. While effective in single-agent settings, they either lack multi-agent capabilities or suffer from slow CPU-bound environments. For MARL, environments like SMAC [24], MPE [19], and Melting Pot [1] are widely used, but are not designed for continual evaluation. Overcooked [5] has emerged as a useful domain for studying coordination, with recent implementations in JAX [23]. Our benchmark builds on Overcooked and introduces procedural variation to create long task sequences for continual MARL.

Overcooked The Overcooked environment [5] is a cooperative multi-agent benchmark inspired by the popular video game of the same name. Agents control chefs in a grid-based kitchen, coordinating to prepare and deliver dishes through sequences of interactions with environment objects such as pots, ingredient dispensers, plate stations, and delivery counters. The environment is designed to require both motion and strategy coordination, making it a standard testbed for evaluating collaborative behaviors.

Compared to the large state spaces and high agent counts in benchmarks like Melting Pot [1] and SMAC [24], Overcooked operates on small grid-based environments with only two agents. However, its complexity arises not from scale but from credit assignment challenges due to shared rewards, and the need for precise coordination, as agents must execute tightly coupled action sequences to complete tasks successfully [13]. However, its fully observable and symmetric setup reduces the need for explicit communication.

Table 1: Comparison of existing Reinforcement Learning benchmarks with MEAL

Benchmark	No. Tasks	Difficulty Levels	GPU-accelerated	Action Space	Multi-Agent	Continual Learning
CORA [21]	31	✗	✓	Mixed	✗	✓
MPE [19]	7	✗	✗	Continuous	✓	✗
SMAC [24]	14	✓	✗	Discrete	✓	✗
Continual World [29]	10	✗	✗	Continuous	✗	✓
Melting Pot [1]	49	✗	✗	Discrete	✓	✗
Google Football [16]	14	✓	✓	Discrete	✓	✗
JaxMARL [23]	33	✗	✓	Mixed	✓	✗
COOM [28]	8	✓	✗	Discrete	✗	✓
MEAL	25	✓	✓	Discrete	✓	✓

3 Preliminaries

Cooperative Multi-Agent MDP We formulate the setting as a fully observable cooperative multi-agent task, modeled as a Markov game defined by the tuple $\langle N, S, {A^i}_{i \in N}, P, R, \gamma \rangle$, where N is the number of agents, S is the state space, A^i is the action space of agent i with joint action space $A = A^1 \times \cdots \times A^N$, $P : S \times A \times S \to [0, 1]$ is the transition function, $R : S \times A \times S \to \mathbb{R}$ is a shared reward function, and $\gamma \in [0, 1)$ is the discount factor. In the fully observable setting, each agent receives the full state $s \in S$ at every time step.

Continual MARL We consider a continual MARL setting in which a shared policy $\pi_\theta = {\pi^i_\theta}_{i \in N}$ is learned over a sequence of tasks $\mathcal{T} = \mathcal{M}_1, \ldots, \mathcal{M}_T$, where each $\mathcal{M}_t = \langle N, S_t, A^i i \in N, P_t, R_t, \gamma \rangle$ is a fully observable cooperative Markov game with consistent action and observation spaces. At training phase t, agents interact exclusively with $\mathcal{M}_t$ for a fixed number of iterations Δ, collecting trajectories $\tau_{t,1}, \ldots, \tau_{t,\Delta}$ to update their policy. Past tasks and data are inaccessible, and no joint training or replay is allowed. The focus of this work is on the task-incremental setting, where the task identity is known during training but hidden at evaluation. The objective is to maximize cumulative performance across all tasks and mitigate forgetting.

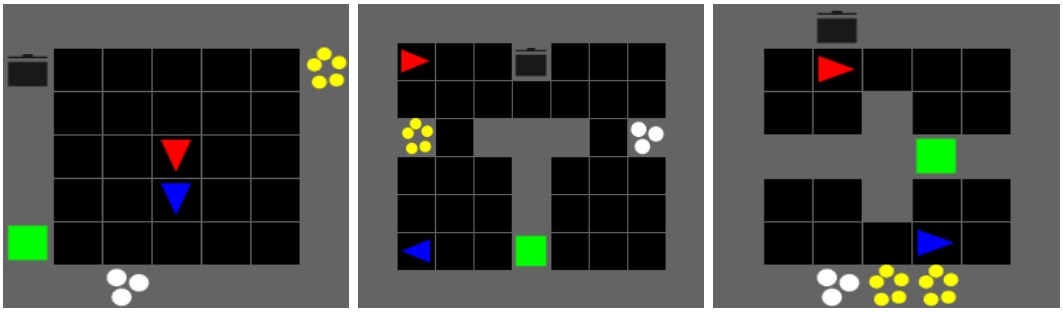
4 MEAL

We introduce MEAL, the first benchmark for CMARL, built on the JaxMARL [23] version of Overcooked. JAX [4] provides just-in-time compilation, automatic differentiation, and vectorization through XLA, enabling high-performance and accelerator-agnostic computation. We incorporate the original five layouts from Overcooked-AI [5] and design 20 additional handcrafted environments.

4.1 Environment Dynamics

Observations Each agent receives a fully observable grid-based observation of shape $(H, W, 26)$, where H and W are the height and width of the environment, and the 26 channels encode entity types (e.g., walls, agents, onions, plates, pots, delivery stations) and object states (e.g., cooking progress, held item). To ensure compatibility across environments in a continual learning setting, we fix H_{max} and W_{max} to the largest layout size and pad smaller layouts with walls. Observations are then standardized to shape $(H_{\text{max}}, W_{\text{max}}, 26)$.

Action Space At each timestep, both agents select one of six discrete actions from a shared action space $\mathcal{A} = \{\texttt{up}, \texttt{down}, \texttt{left}, \texttt{right}, \texttt{stay}, \texttt{interact}\}$. Movement actions translate the agent forward if the target tile is free (i.e., not a wall or occupied), while `stay` maintains the current position. The `interact` action is context-dependent and allows agents to pick up or place items, add ingredients to pots, serve completed dishes, or deliver them at the goal location. Importantly, there is no built-in communication action; all coordination emerges from environment interactions.



(a) Easy layouts are solvable by a single agent as long as key tiles remain accessible. Minimal coordination is required, and navigation is straightforward.

(b) Medium layouts contain bottlenecks that restrict movement and increase the likelihood of deadlocks. Agents must coordinate to avoid obstructing each other.

(c) Hard layouts split the map into disjoint regions, forcing agents to specialize. Solving the task requires deliberate cooperation and division of labor.

Figure 1: MEAL environments are grouped by layout difficulty: **easy** (minimal coordination), **medium** (bottlenecks and deadlocks), and **hard** (specialized cooperation due to partitioned access).

Rewards Agents receive a shared team reward. The primary sparse reward is $+20$ for successfully delivering a completed soup. Optional shaped rewards can be added for partial task completion:

$$r_t = r_{\text{deliver}} + \alpha_1 \cdot \mathbb{1}_{\text{onion\_in\_pot}} + \alpha_2 \cdot \mathbb{1}_{\text{plate\_pickup}} + \alpha_3 \cdot \mathbb{1}_{\text{soup\_pickup}}, \tag{1}$$

where $\alpha_1, \alpha_2, \alpha_3$ are reward shaping coefficients. All rewards are shared, encouraging cooperative behavior.

Score Function Because MEAL environment layouts vary in size, the maximum achievable return differs per task. To ensure consistent comparison across sequences, we normalize returns using a reference performance: the average return of a converged IPPO agent trained from scratch on each environment. This normalizes the baseline score to 1. Note that scores can exceed 1 if a method generalizes or transfers better than the isolated baseline. We refer to this metric as the *IPPO-Normalized Score (INS)*.

4.2 Layout Difficulty

We categorize the handcrafted MEAL layouts into three levels of difficulty to better interpret agent behavior and learning dynamics. Appendix A depicts all the available MEAL layouts in difficulty groups. This grouping disentangles which coordination skills agents can acquire under varying structural constraints. Figure 1 illustrates representative layouts for each tier. In **easy** layouts, a single agent can often complete the task independently as long as key tiles remain unobstructed. **Medium** layouts introduce structural bottlenecks and tighter spatial constraints. Agents must coordinate their movement, such as implicit turn-taking in narrow passages, to sustain task throughput. **Hard** layouts partition the map into disjoint regions, often limiting each agent's access to only a subset of utilities (e.g., one agent sees only plates and onions). This forces agents to specialize and rely on their partner to complete the part of the recipe pipeline. Continual learning becomes especially challenging: agents must infer their new role based on the layout, and adapt strategies without forgetting past roles.

4.3 Task Sequences

Rather than a continuous domain shift, MEAL sequences involves discrete task boundaries, where agents transition between clearly distinct environments. This setup aligns with the task-incremental learning paradigm. We include three task sequence strategies.

Ordered Tasks follow a fixed sequence. This setting enables controlled comparisons and rapid iteration during development, as the order remains constant across runs. Since the fixed task order reduces variance, fewer seeds are needed to draw reliable conclusions.

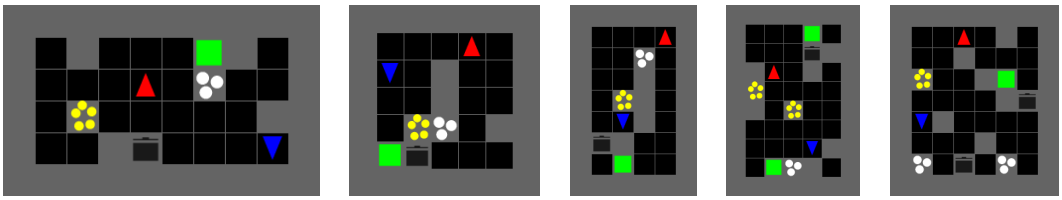


Figure 2: Five randomly generated Overcooked layouts. Each kitchen is guaranteed to be solvable.

Random To evaluate robustness, we sample task sequences randomly without replacement from the available pool. Since tasks differ substantially, the structure of the sequence has a strong impact on learning dynamics and knowledge transfer. Random ordering highlights method sensitivity to transferability between task pairs, and reflects the findings of Tomilin et al. [28] that performance often hinges on the characteristics of the first task and its downstream transfer potential.

Generated To support long sequences and continuous benchmarking, we procedurally generate new Overcooked layouts on the fly. Each layout is guaranteed to be solvable and varies in size, structure, and item placement. Figure 2 shows examples of generated environments. This setting offers a virtually infinite supply of tasks and evaluates true lifelong learning under continual exposure to unseen configurations.

4.4 Evaluation Metrics

We evaluate methods on three core metrics: **Average Performance**, **Forgetting**, and **Plasticity**. Let $s_i(j)$ denote the normalized score (see Section 4.1) on task j after training on task i, and let the task sequence consist of N tasks.

Average Performance We define average performance as the mean normalized score across all tasks at the end of training. This metric captures the balance between forward transfer and retention:

$$\mathcal{A} = \frac{1}{N} \sum_{i=1}^{N} s_N(i) \tag{2}$$

Forgetting Forgetting quantifies the degradation in performance on past tasks due to interference from training on later ones. For each task $i < N$, we compute the difference between the performance immediately after training and at the end of the sequence:

$$\mathcal{F} = \frac{1}{N-1} \sum_{i=1}^{N-1} \left(s_i(i) - s_N(i) \right) \tag{3}$$

Plasticity To evaluate continual training capacity over long task sequences, we measure the model's ability to fit new tasks under continual learning constraints. We compute the average training score (i.e., final performance on the current task right after training) across the entire sequence:

$$\mathcal{P} = \frac{1}{N} \sum_{i=1}^{N} s_i(i) \tag{4}$$

This isolates how quickly and effectively the method learns a new task under capacity constraints, independently of retention. Unlike Average Performance, Plasticity does not require evaluation on previously seen tasks.

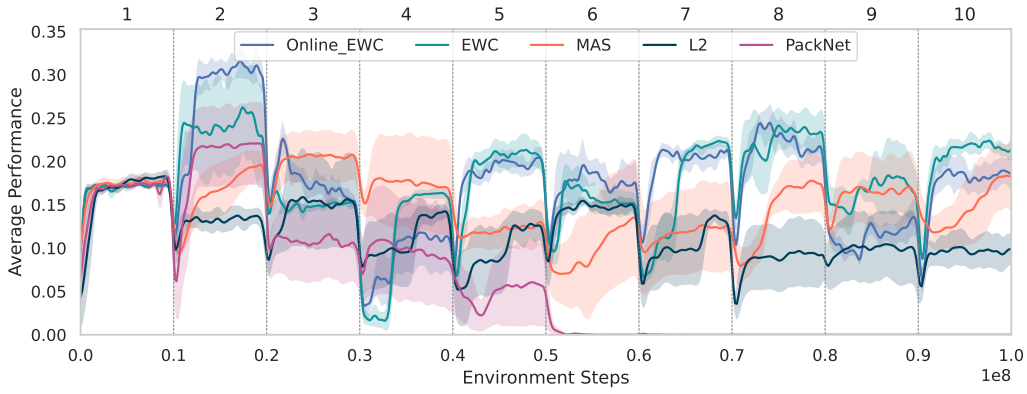


Figure 3: **Average performance** over the course of training on a 10-task sequence using the **Random** sequence strategy. Shaded regions indicate 95% confidence intervals across 5 seeds. Performance is measured as average normalized return across all tasks in the sequence.

5 Experiments

5.1 Setup

The agent is trained on each task $\mathcal{T}_i$ for $\Delta = 10^7$ environment steps on-policy. During training, we evaluate the policy after every 100 updates by running 10 evaluation episodes on all previously seen tasks. To ensure comparability across tasks with different layouts and reward scales, we normalize raw returns using a per-task transformation $f_i(\cdot)$, defined such that $f_i(score) = 0$ corresponds to a random agent and $f_i(score) = 1$ corresponds to a reference policy trained directly on task $\mathcal{T}_i$ until convergence (we use IPPO as the reference). This yields a unified measure of success across tasks.

We run each environment for 10 million environment steps using the random task selection strategy, repeated over five seeds. We leverage JAX to reduce the wall-clock time for training on a single environment to around 5 minutes. All experiments are conducted on a dedicated compute node with a 72-core 3.2 GHz AMD EPYC 7F72 CPU and a single NVIDIA A100 GPU. We adopt many of JAXMarl's default settings for our network configuration, IPPO setup, and training processes. For exact hyperparameters please refer to Appendix B.

5.2 Baselines

We evaluate several continual learning methods. Fine-tune (**FT**) is a naive baseline where the policy is trained sequentially across tasks without any mechanism to prevent forgetting. **L2-Regularization** [14] adds a penalty on parameter changes to encourage stability. **EWC** [14] is a regularization method that penalizes changes to important parameters, with importance measured using the Fisher Information Matrix. **Online EWC** is a variant that maintains a running estimate of parameter importance, making it more suitable for longer sequences. **MAS** [2] computes importance based on how parameters influence the policy's output, rather than gradients. PackNet [18] incrementally allocates parts of the network to each task through pruning and freezing. Finally, Continual Backpropagation (**CBP**) [9] introduces architectural plasticity by periodically replacing parts of the network to preserve adaptability over many tasks. As the MARL baseline, we opt for IPPO [8]. It is a natural choice as it can be seamlessly integrated with all continual learning methods. It has been shown to outperform other MARL approaches on both SMAC [8] and Overcooked [23], making it a strong candidate for evaluating continual multi-agent learning in a fully observable setting.

5.3 Baseline Comparison

Figure 3 compares the performance of several continual learning methods combined with IPPO over a 10-task sequence. None of the methods fully retain prior knowledge, most tasks are completely forgotten once access is lost. Regularization-based approaches like EWC and MAS reduce forgetting to a degree, but their long-term performance gains are limited. PackNet, while somewhat preserving

Figure 4: Comparison of final average performance using MLP vs. CNN encoders for EWC and MAS.

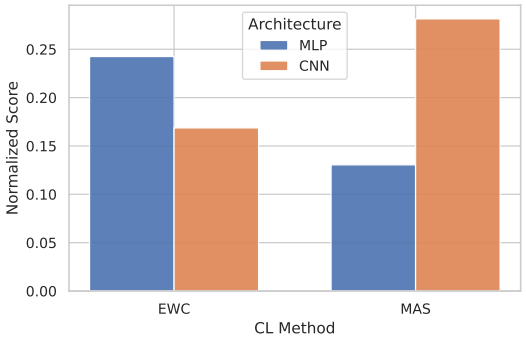


Table 2: Comparison of CMARL performance across CL methods on a 10-task sequence with random order. Results are averaged over 5 seeds. CMARL metrics: $\mathcal{A}$ (avg. performance), $\mathcal{F}$ (forgetting), $\mathcal{P}$ (plasticity).

Method	$\mathcal{A}\uparrow$	$\mathcal{F}\downarrow$	$\mathcal{P}\uparrow$
FT	0.122	0.682	0.794
Online EWC	0.118	0.739	0.870
EWC	0.121	0.710	**0.881**
MAS	**0.131**	**0.294**	0.443
PackNet	0.040	0.318	0.312
L2	0.064	0.695	0.735

earlier tasks, quickly exhausts its capacity and fails to learn anything. Table 2 reports summary metrics over 5 seeds. MAS achieves the best average performance and lowest forgetting, though at the cost of reduced plasticity. In contrast, methods like FT, EWC and Online EWC display high plasticity but struggle with retention, highlighting the inherent stability–plasticity trade-off in CMARL. These results show the difficulty of maintaining both adaptability and memory in cooperative continual multi-agent environments.

5.4 Forgetting

Comparison of CMARL performance across continual learning methods on a 10-task sequence with random order. Results are averaged over 5 seeds. $\mathcal{A}$ measures final average performance, $\mathcal{F}$ captures forgetting, and $\mathcal{P}$ reflects plasticity. MAS achieves the best overall performance and retention, while FT shows high plasticity but suffers from catastrophic forgetting

Figure 5 illustrates the extent of forgetting across tasks for FT, EWC, and MAS. Fine-tune serves as a clear example of catastrophic forgetting. After transitioning to a new task, performance on previous tasks rapidly collapses. In contrast, EWC and MAS manage to retain some knowledge of earlier tasks, particularly the first one, but fail to reach the same training returns on later tasks as FT, demonstrating the trade-off between stability and plasticity.

5.5 Encoder Architecture

In our main experiments, we adopt an MLP encoder due to its simplicity and compatibility with low-dimensional inputs. To explore the effect of encoder choice on CMARL, we evaluate EWC and MAS with a CNN-based encoder. Figure 4 shows the impact of architecture on performance. EWC performs slightly better with an MLP encoder, suggesting that its regularization interacts more favorably with simpler representations. In contrast, MAS exhibits a nearly $2\times$, when paired with a CNN encoder, suggesting that its functional sensitivity estimation benefits from spatial structure and richer features.

6 Conclusion

We introduced MEAL, the first benchmark for continual multi-agent reinforcement learning. By leveraging JAX for efficient GPU-accelerated training and introducing a diverse set of handcrafted and procedurally generated Overcooked environments, MEAL enables the study of long-horizon continual learning in cooperative settings. Our evaluation of six continual learning methods combined with the IPPO algorithm reveals that existing CL techniques struggle to retain cooperative behaviors while maintaining adaptability to new tasks. Regularization-based methods mitigate forgetting but sacrifice plasticity, while parameter-isolation methods fail to scale with longer task sequences. These findings highlight the need for new approaches that can handle the dual challenges of cooperation and non-stationarity in CMARL. We hope MEAL serves as a foundation for advancing this underexplored but important research direction.

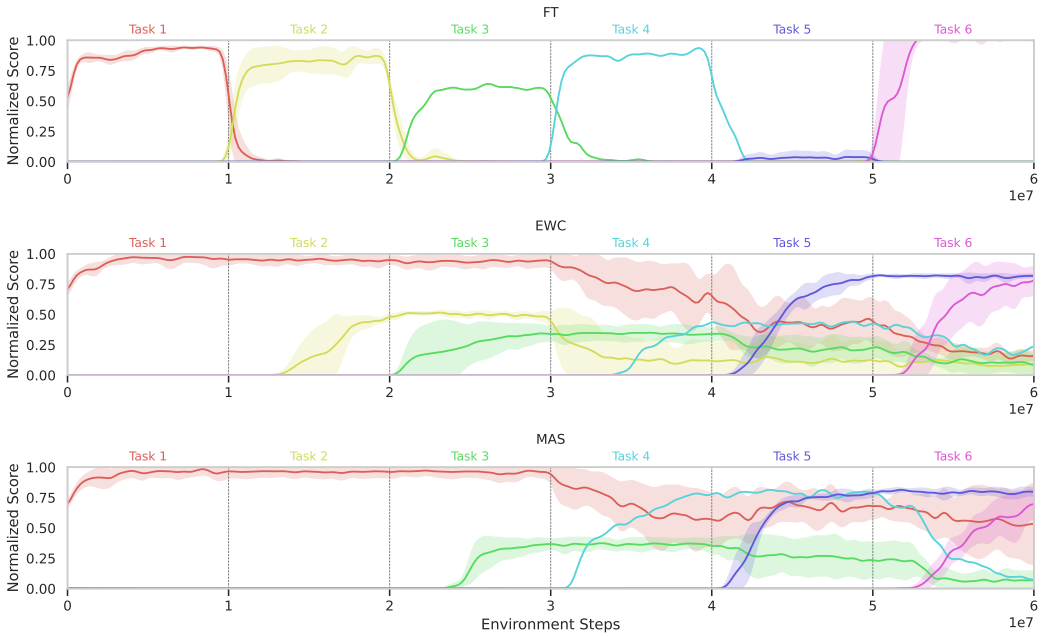


Figure 5: Normalized evaluation score of each task in the 6-task sequence during training.

7 Limitations

While MEAL provides a scalable and diverse testbed for continual multi-agent reinforcement learning, several limitations remain. First, MEAL is restricted to fully observable, two-agent environments with discrete action spaces, limiting its applicability to partially observable or competitive multi-agent settings. Second, while layout diversity is high, the domain itself is narrow. Overcooked dynamics do not capture the full complexity of real-world multi-agent interactions involving language, negotiation, or long-horizon planning. Third, our benchmark only evaluates task-incremental learning. Future work could extend MEAL to other continual learning protocols. Finally, we only consider continual learning in settings where the environment layout changes across tasks, but not the partner agent.

References

[1] John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pages 139–154, 2018.

[3] Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428: 291–307, 2021.

[4] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

[5] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d3db40d65b49f8f9e383ac5dd-Paper.pdf`.

[6] Feng Chen, Fuguang Han, Cong Guan, Lei Yuan, Zhilong Zhang, Yang Yu, and Zongzhang Zhang. Stable continual reinforcement learning via diffusion-based trajectory replay. *arXiv preprint arXiv:2411.10809*, 2024.

[7] Wesley Chung, Lynn Cherif, Doina Precup, and David Meger. Parseval regularization for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 127937–127967, 2024.

[8] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[9] Shibhansh Dohare, Richard S Sutton, and A Rupam Mahmood. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

[10] Zeki Doruk Erden, Donia Gasmi, and Boi Faltings. Continual reinforcement learning via autoencoder-driven task and new environment recognition. In *The Seventeenth Workshop on Adaptive and Learning Agents*.

[11] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[12] Muhammad Burhan Hafez and Kerim Erekmen. Continual deep reinforcement learning with task-agnostic policy distillation. *Scientific Reports*, 14(1):31661, 2024.

[13] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[15] Paul Knott, Micah Carroll, Sam Devlin, Kamil Ciosek, Katja Hofmann, Anca D Dragan, and Rohin Shah. Evaluating the robustness of collaborative agents. *arXiv preprint arXiv:2101.05507*, 2021.

[16] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4501–4510, 2020.

[17] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[18] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[19] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[20] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[21] Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *Conference on Lifelong Learning Agents*, pages 705–743. PMLR, 2022.

[22] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[23] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 2444–2446, 2024.

[24] 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. *arXiv preprint arXiv:1902.04043*, 2019.

[25] Artyom Y Sorokin and Mikhail S Burtsev. Continual and multi-task reinforcement learning with shared episodic memory. *arXiv preprint arXiv:1905.02662*, 2019.

[26] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34: 14502–14515, 2021.

[27] 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. *arXiv preprint arXiv:1706.05296*, 2017.

[28] Tristan Tomilin, Meng Fang, Yudi Zhang, and Mykola Pechenizkiy. Coom: a game benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

[29] Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

[30] 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. *Advances in neural information processing systems*, 35:24611–24624, 2022.

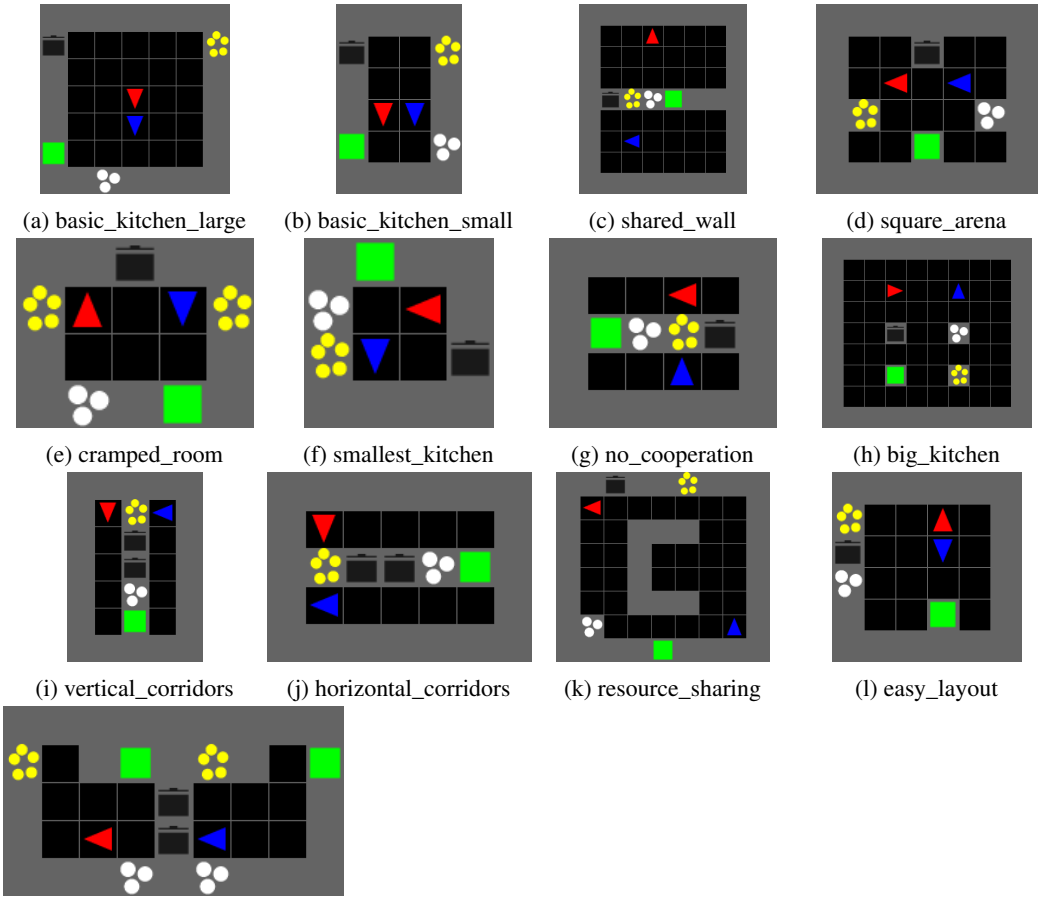
[31] Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. A survey of progress on cooperative multi-agent reinforcement learning in open environment. *arXiv preprint arXiv:2312.01058*, 2023.

[32] Lei Yuan, Lihe Li, Ziqian Zhang, Fuxiang Zhang, Cong Guan, and Yang Yu. Multiagent continual coordination via progressive task contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[33] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR, 2017.

A Environment Layouts

A.1 Easy Layouts



(a) basic_kitchen_large (b) basic_kitchen_small (c) shared_wall (d) square_arena

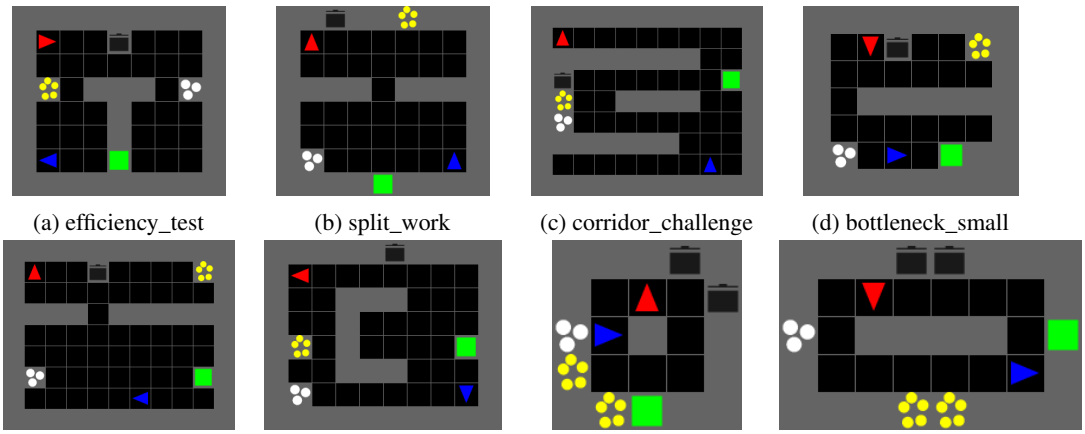
(e) cramped_room (f) smallest_kitchen (g) no_cooperation (h) big_kitchen

(i) vertical_corridors (j) horizontal_corridors (k) resource_sharing (l) easy_layout

(m) asymm_advantages

Figure 6: Easy MEAL layouts (coordination not required).

A.2 Medium Layouts

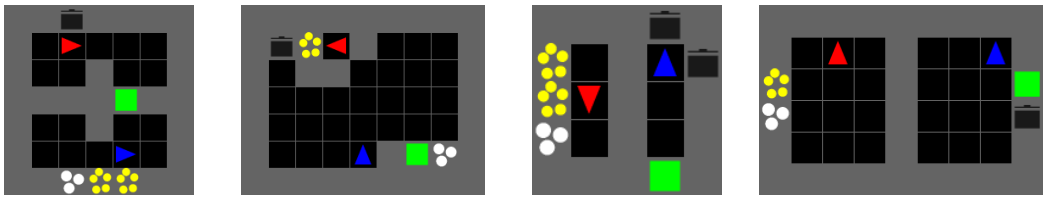


(a) efficiency_test (b) split_work (c) corridor_challenge (d) bottleneck_small

(e) bottleneck_large (f) c_kitchen (g) coord_ring (h) counter_circuit

Figure 7: Medium MEAL layouts (bottlenecks and deadlock risk).

A.3 Hard Layouts



(a) split_kitchen (b) basic_cooperative (c) forced_coord (d) forced_coord_2

Figure 8: Hard MEAL layouts (partitioned regions, specialization needed).

B Hyperparameters

Table 3: Common hyper-parameters for all MEAL experiments. Values are fixed across methods and experiments unless stated otherwise.

Parameter	**Value**
IPPO / optimisation	
Learning rate η	3×10^{-4}
Anneal LR	No (linear schedule optional)
Total env. steps per task Δ	10^7
Num. envs (parallel)	16
Rollout length T	128
Update epochs	8
Minibatches per update	8
Batch size	$16 \times 128 = 2048$
γ	0.99
GAE λ	0.957
Clipping ϵ	0.2
Entropy coef. α_{ent}	0.01
Value-loss coef. α_{vf}	0.5
Max grad-norm	0.5
Continual-learning specifics	
Sequence length $\lvert\mathcal{T}\rvert$	10 tasks (random order)
CL method coefficients λ	1×10^6 (EWC) / 1×10^5 (L2, MAS)
EWC mode / decay	Online / 0.9
Importance episodes / steps	5 / 500
Regularise critic / heads	No / Yes
Misc. settings	
Reward shaping	Yes, anneal to 0 after 2.5×10^6 steps
Evaluation interval	every 100 updates (10 episodes)
Seeds	$\{1, 2, 3, 4, 5\}$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions: MEAL as the first benchmark CMARL benchmark, its GPU-accelerated JAX implementation, and empirical evaluation of six continual learning methods. These claims are supported by the benchmark design and experimental results presented in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in Section 7.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not present any theoretical results or proofs

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All experimental settings, hardware, hyperparameters, and evaluation protocols are described in the main text and appendix. The use of JAX ensures deterministic execution and full reproducibility of results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We included an anonymous GitHub link and will release the full codebase and environment definitions upon publication, including setup instructions, training scripts, and evaluation tools, to ensure faithful reproduction of all experimental results.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant training and evaluation details, including optimizer settings, number of steps, number of environments, regularization parameters, encoder architectures, and evaluation frequency, are provided in the main paper, with a full list of hyperparameters summarized in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 95% confidence intervals across 5 random seeds in relevant plots (Figure 3 and Figure 5).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Explained in Section 5.1

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work adheres fully to the NeurIPS Code of Ethics. It involves no human subjects, no sensitive data, and no foreseeable misuse or dual-use concerns. All used assets are open-source and properly credited.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The authors cannot foresee notable societal impacts from releasing a reinforcement learning benchmark.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All reused assets, mainly including Overcooked-AI and JaxMARL, are properly cited in the paper with corresponding references.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The benchmark code, environment layouts, and evaluation scripts are new assets. They are documented and will be released with accompanying instructions.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The benchmark development does not involve LLMs

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.