CoDreamer: Communication-Based Decentralised World Models

Edan Toledo
e.toledo@instadeep.com
InstaDeep

Amanda Prorok
asp45@cam.ac.uk
University of Cambridge

Abstract

Sample efficiency is a critical challenge in Reinforcement Learning. Model-based RL has emerged as a solution, but its application has largely been confined to single-agent scenarios. In this work, we introduce CoDreamer, an extension of the Dreamer algorithm for multi-agent environments. CoDreamer leverages Graph Neural Networks for a two-level communication system to tackle challenges such as partial observability and inter-agent cooperation. Communication is separately utilised within the learned world models and within the learned policies of each agent to enhance modelling and task-solving. We show that CoDreamer offers greater expressive power than a naive application of Dreamer, and we demonstrate its superiority over baseline methods across various multi-agent environments.

1 Introduction

Reinforcement Learning (RL) has become a leading paradigm for creating autonomous control systems. Despite recent successes (Degrave et al., 2022; Luo et al., 2022; Roy et al., 2021; Mirhoseini et al., 2021; Schrittwieser et al., 2020), these achievements often require vast data and computational resources. While computational power is likely to increase, data availability remains a constraint. As we tackle more complex problems requiring expensive simulations, improving the sample efficiency of RL methods is critical.

Model-based RL (Sutton & Barto, 2018) offers a promising solution to this issue. Recent algorithms like EfficientZero (Ye et al., 2021) and Dreamer (Hafner et al., 2020; 2021; 2023) show that high performance can be achieved utilising far less data. These methods develop a world model, a learned simulation of the agent's environment, to generate synthetic data and/or plan actions based on future predictions. However, their effectiveness in multi-agent settings is limited by the accuracy of these models and their single-agent design.

Transitioning to multi-agent systems introduces new challenges such as partial observability and non-stationarity, complicating the use of many single-agent RL algorithms (Nguyen et al., 2020). Sample efficiency issues become more pronounced when multiple agents need to learn a shared set of policies. This work aims to adapt the success of single-agent model-based RL to multi-agent environments.

We propose Communicative Dreamer (CoDreamer), an enhancement to the Dreamer algorithm, addressing challenges like partial observability and non-stationarity in multi-agent environments. CoDreamer employs a two-level communication system: agents communicate within their world models to better model their environment, and within their policies to enhance cooperation and performance. This dual-tiered strategy aims to overcome the limitations of existing methods and advance model-based RL in multi-agent scenarios.

As agents use their world models during execution, decentralised communication is essential. Recent research (Li et al., 2020; Tolstaya et al., 2020; Prorok, 2018; Kortvelesy & Prorok, 2022) shows that

Graph Neural Networks (GNNs) can enhance communication and performance in multi-agent systems within the CTDE framework. CoDreamer utilises GNNs for a learned communication strategy, allowing agents to collaboratively generate synthetic trajectories, thereby improving performance given a limited sample budget.

This work makes several key contributions: the implementation and evaluation of the DreamerV3 (Hafner et al., 2023) algorithm in a multi-agent independent learning setting, termed IDreamer; the development of CoDreamer, an enhanced version of IDreamer that uses GNNs for decentralised communication among agents' world models and policies, addressing issues like non-stationarity and partial observability; and a comprehensive evaluation of CoDreamer across various environments, demonstrating superior performance and more accurate world models in environments with inter-agent dependencies.

2 Methodology

2.1 Independent Dreamer

To thoroughly examine the effects of communication within world models, we first assess the performance of fully decentralised non-communicative world models. This leads to the creation of distinct Dreamer agents, each using its own separate world model. We introduce IDreamer, which implements DreamerV3 (Hafner et al., 2023) for the multi-agent setting.

Like most independent multi-agent RL (MARL) algorithms, each IDreamer agent treats other agents as part of the environment, with no explicit communication or opponent modelling. Each agent trains solely on its own experiences and does not observe other agents during training or execution.

During actor-critic network training, agents use only their own observations to start trajectory imagination. Due to the substantial size of the world model networks (16M+ parameters), we use parameter sharing (Gupta et al., 2017) to expedite learning. Although parameter sharing allows the world model to train on all agents' experiences, it is still conditioned on individual agents' observations, without additional information for trajectory imagination or observation encoding.

Given the homogeneous nature of the agents and parameter sharing, the world model may struggle to distinguish which agent it is being used by. To address this, we concatenate agent indices as a one-hot encoding to all observations in non-visual environments, enabling the world model to differentiate between agents. We hypothesise that in visual environments, there is sufficient information for the model to distinguish agents.

Apart from these multi-agent specific changes, the implementation details of IDreamer remain unchanged from DreamerV3.

2.2 CoDreamer: Communicative World Models

Using world models independently can exacerbate the challenges of multi-agent systems. In direct-style model-based methods like IDreamer, each policy is trained solely with data from its agent's world model, making policy performance entirely dependent on the model's accuracy. Issues like non-stationarity, partial observability, and cooperation are worsened if the model fails to capture other agents' actions. Independent world models struggle as the environment dynamics seem to constantly change, and single-agent observations and actions are insufficient to infer future information, potentially hindering accurate model development.

Moreover, if the environment's reward and transition functions are highly inter-agent dependent, learning through imagination becomes impossible because the imagined trajectories will be inaccurate. To address these issues while staying within the CTDE framework, we propose CoDreamer.

CoDreamer enhances IDreamer by incorporating two levels of communication. The first level, used and learned by the world models, improves trajectory imagination among agents, helping with non-

stationarity and partial observability. This communication is independent of actor-critic learning and focuses solely on better state representations and environment modelling. The world models learn communication grounded in the specific environment rather than any agent's policy.

The second level of communication is dedicated to the actor and critic networks trained during imagination, enabling agents to share action and value prediction information. This level is consistent with communication in other MARL methods, focusing on action-relevant information.

For both communication levels, we use GNNs due to their applicability in both centralised and decentralised contexts. GNNs, in our case the GAT V2 architecture (Brody et al., 2022), enable k-hop aggregation of nodes, allowing all agents to share information as long as each can communicate with at least one other agent.

2.2.1 Communication

As is consistent with the literature (Kortvelesy & Prorok, 2022; Li et al., 2020; Blumenkamp & Prorok, 2021), to learn communication between agents using GNNs, we model the inter-agent communication with a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where each node $i \in \mathcal{V}$ represents an individual agent $i \in n$, and each edge $e^{ij} \in \mathcal{E}$ represents a communication link between agents i and j. The adjacency matrix $\mathbf{A}$ establishes the edge set for each agent, which is contingent on the agent's communication range C set by the environment. In this work, adjacency matrices are constructed using Euclidean distance. Specifically, given two agents i and j with positions p^i and p^j, an edge is created if $||p^i - p^j|| \leq C$. This communication range is dependent on the environment and other limiting factors. As the current state $s \in \mathcal{S}$ changes over time, $\mathbf{A}$ adapts dynamically based on the changing positions of the agents. The set of neighbouring agents $\mathcal{N}$ that are capable of communicating with agent i is defined as $\mathcal{N}^i = \{v^j | e^{ij} \in \mathcal{E}\}$. Additionally, the Euclidean distance of each edge e^{ij} is stored in a matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ that can be utilised by the GNNs.

2.2.2 World Models

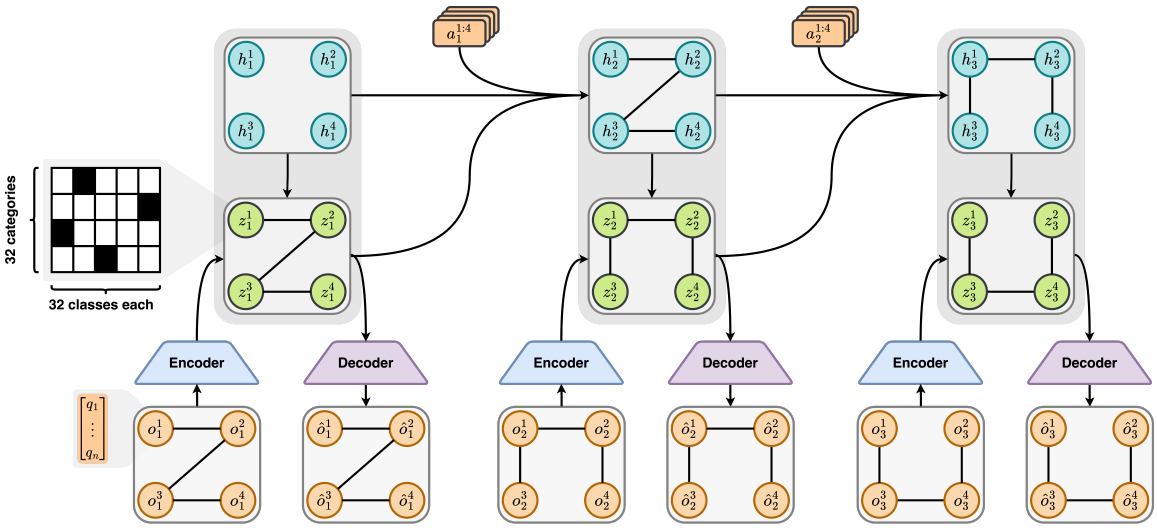

Figure 1: **Training process of CoDreamer world model:** CoDreamer trains in a similar manner to IDreamer (see Figure 21). However, CoDreamer operates over graphs of observations G_t^o and produces graphs of stochastic posterior G_t^z and prior $\hat{G}_t^z$ states. Additionally, recurrent states are also represented as graphs G_t^h. This allows communication to happen at any level within the world model architecture.

In CoDreamer, a unified world model can be used independently by clusters of agents during execution. All elements from IDreamer remain unchanged except for the addition of k GNN layers within

the Recurrent State-Space Model (RSSM) and prediction heads to facilitate communication. Specifically, only the reward and terminal state prediction heads use communication, while the decoder head remains independent. Each GNN layer performs node updates and neighborhood aggregation, concatenating the original node features to the output to help the world model distinguish agents and prevent over-smoothing (Li et al., 2018).

Although we focus on environments formalised as Dec-POMDPs, CoDreamer is adaptable to various multi-agent environments and formalisms, such as Markov Games (Shapley, 1953), due to each agent's ability to independently use their prediction heads for local environment estimates.

Unlike IDreamer, CoDreamer's training involves graphs representing all agents' experiences rather than individual experiences. Depending on the adjacency matrix $\mathbf{A}$, agents can be processed independently, effectively transforming CoDreamer into IDreamer for those agents. All graphs have edge features representing the relative distances between agents, which are used with observations in the GAT V2 to calculate attention coefficients for neighborhood aggregation. Beyond these and architectural modifications, CoDreamer remains consistent with IDreamer.

Figure 1 shows CoDreamer unrolling its world model on a sequence of graphs where each node represents an agent's observation.

2.2.3 Behaviour Learning

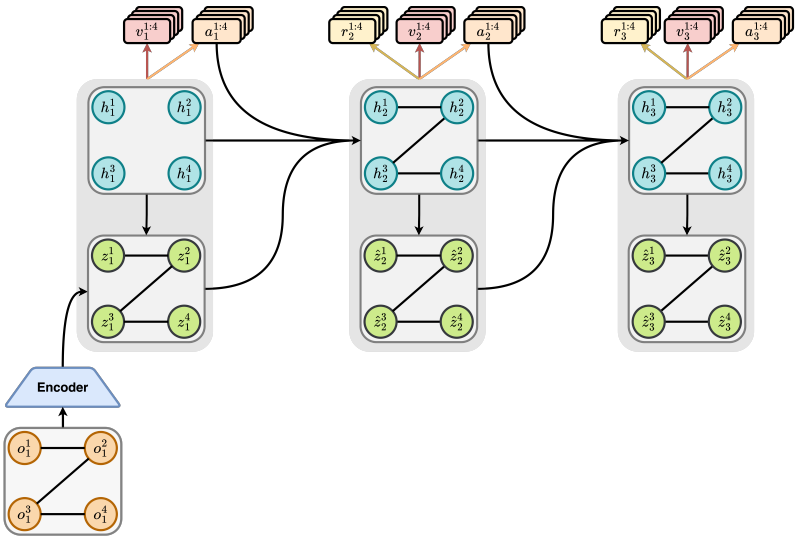

Figure 2: **Training process of CoDreamer Actor-Critic:** To train the actor-critic networks in CoDreamer, trajectories of compact world model state graphs $G^{z||h}_{t:H}$ are produced. As CoDreamer does not predict new adjacency matrices over the *imagined* trajectories, the adjacency matrix of the starting graph is used for all subsequent graphs. We posit that, as the most relevant agents for short-term future predictions are those nearby at the start, this is sufficient to create more consistent trajectories.

CoDreamer modifies the behavior learning process similarly to its training procedure. The world model generates trajectories of graphs, with each agent's compact world model state as node features. These imagined trajectories use the adjacency matrix and edge features of the original graph that started the trajectory imagination. We believe the starting graph's adjacency matrix suffices for necessary communication over the imagination horizon, as relevant agent information for future prediction likely depends on agents within the initial communication range. However, original edge features may become outdated over time. Updating these features every step with the GNN could be a future improvement. Each agent uses the predicted graphs to train their actor and critic networks. Figure 2 illustrates CoDreamer's modifications to IDreamer's behavior learning procedure.

For CoDreamer's synthetic data generation at training time, following the CTDE framework, we could use fully connected adjacency matrices during the imagination rollout. However, since the second level of communication is trained on the imagined graphs, realistic communication topologies must be maintained. Otherwise, at execution time, the actor-critic networks won't effectively communicate relevant information with a non-fully connected adjacency matrix.

In addition to the state information communicated by the world model, agents also communicate relevant information to learn their policy and value functions. Both actor and critic networks, operating on compact world model state graphs, have distinct GNN layers acting as a torso for the respective networks. Figure 3 illustrates how the compact world model graphs are used by the policy and value functions during training and execution. Beyond the architectural changes, the actor-critic networks are trained in the same way as IDreamer.

Through the utilisation of an intra-model communication layer, CoDreamer is afforded a higher level of expressivity in modelling various environments, compared to IDreamer. Unlike IDreamer, which relies on an independent modelling approach, CoDreamer overcomes the significant limitation of not being able to capture inter-agent dependent transition dynamics and reward functions. CoDreamer is strictly more expressive in that it can model the true underlying reward and transition functions of Dec-MDPs (Bernstein et al., 2002), a claim that IDreamer cannot guarantee.

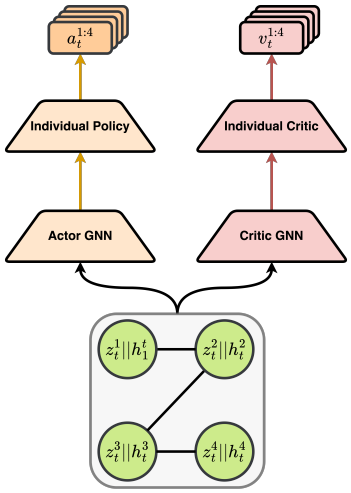

Figure 3: **Illustration of CoDreamer Actor-Critic:** The CoDreamer actor-critic networks operate over the *imagined* graphs of world model states. This second level of communication is trained to maximise the reward and thus exchange relevant information that is not specific to environment modelling.

3 Results

To evaluate the efficacy of our proposed approach, we select two distinct environment suites that present diverse challenges. Firstly, we utilise VMAS (Bettini et al., 2022) to examine the performance of our methods with lower dimensional vector-based observations. Secondly, we use Melting Pot (Agapiou et al., 2022) to investigate the efficacy of our methods in higher-dimensional visual observation settings. We evaluate a variety of separate tasks within each environment suite and utilise the evaluation methodology outlined in by Agarwal et al. (2021), and further explained in section D.1, to produce aggregated results.

3.1 Vectorised Multi-Agent Simulator

Vectorised Multi-Agent Simulator (VMAS)[1](Bettini et al., 2022) is an open-source, 2D physics simulation platform designed to assess various MARL algorithms across multi-agent coordination problems. VMAS offers diverse, challenging scenarios requiring varying degrees of individual skill and collaboration. The observation spaces are vector-based, simulating real-world robotic sensing systems like LIDAR, and the action spaces can be continuous or discrete. In this work, we evaluate our method on three VMAS scenarios: Flocking, Discovery, and Buzz Wire, all with discrete action spaces.

[1]Found at https://github.com/proroklab/VectorizedMultiAgentSimulator

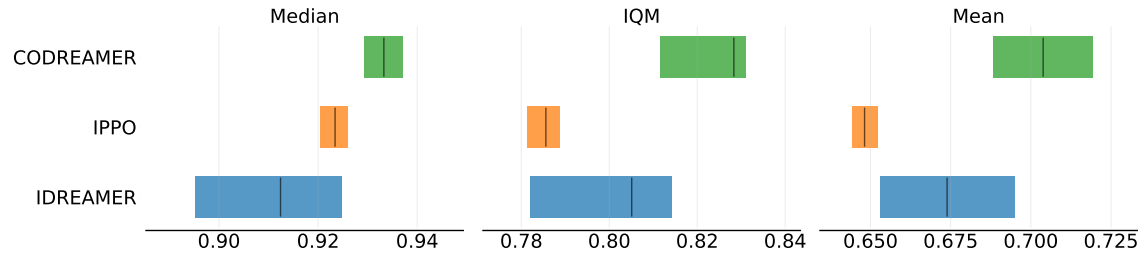

Figure 4: **Aggregate metrics on VMAS Tasks** with 95% CIs. Reported from left to right are: Median, IQM, Mean.

In VMAS, CoDreamer shows superior performance in all point estimate metrics. Figure 4 illustrates that while CoDreamer's point estimates surpass both IPPO and IDreamer, there is some overlap in their IQM and mean Confidence Interval, indicating minor statistical uncertainty. However, the overlap is minor, suggesting with confidence that CoDreamer leads to improved results. To understand performance nuances beyond point estimates, we examine additional metrics.

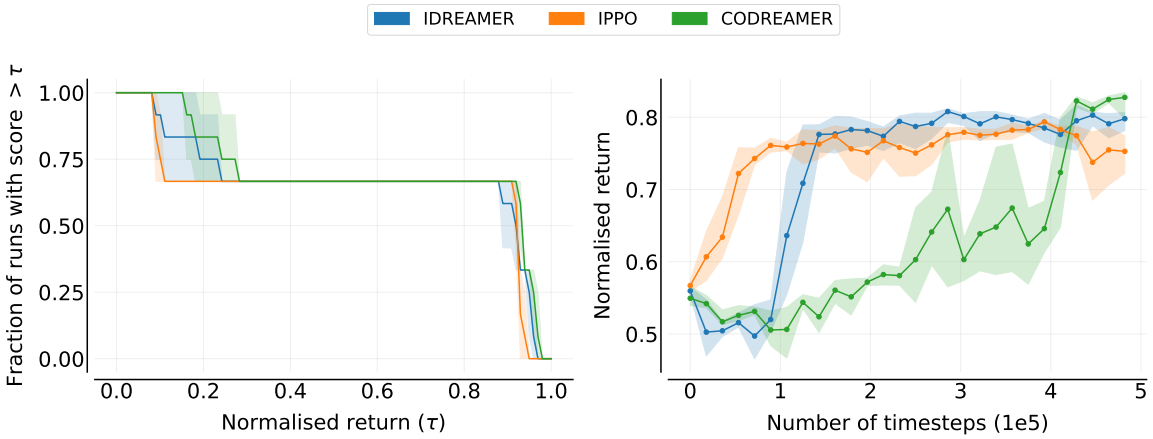

Figure 5: **VMAS Evaluation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right:** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all agents. For both plots, the shaded regions show 95% CIs.

The performance profile in Figure 5 shows that CoDreamer consistently matches or exceeds IDreamer and IPPO across all training runs, suggesting that CoDreamer's communication improves performance. However, the sample efficiency plot reveals a marginal trade-off, with CoDreamer's final performance accompanied by reduced initial sample efficiency. IPPO and IDreamer reach their final performance with less data. This trade-off is expected, as learning communication protocols adds complexity. Both IDreamer and CoDreamer need to learn an accurate environment model before achieving performance gains. While IPPO is initially more sample efficient, IDreamer quickly surpasses it after 100,000 environment steps.

In conclusion, both CoDreamer and IDreamer statistically significantly outperform IPPO, with CoDreamer showing the best results among all algorithms. However, the performance improvements are relatively marginal.

3.2 Melting Pot

Melting Pot[2](Agapiou et al., 2022) benchmarks MARL by evaluating agents' ability to generalise and adapt to unfamiliar environmental and social contexts. It includes tasks that assess various social interactions such as cooperation, competition, and trust. Agents are trained in specific games and assessed in unique test scenarios to evaluate their generalisation.

In this work, we focus on cooperative performance rather than social generalisation. We train and evaluate agents only on the original games, excluding Melting Pot's unique test scenarios. Melting Pot's observation space is pixel-based and partially observable, with each agent having a distinct field of view. This setup demonstrates our method's capability to model complex, high-dimensional observations and highlights the benefits of communication in prediction and policy learning. We select four scenarios: Daycare, Cooperative Mining, and two variants of Collaborative Cooking. Each scenario is transformed into a Dec-POMDP by using a single global reward, which is the sum of individual rewards.

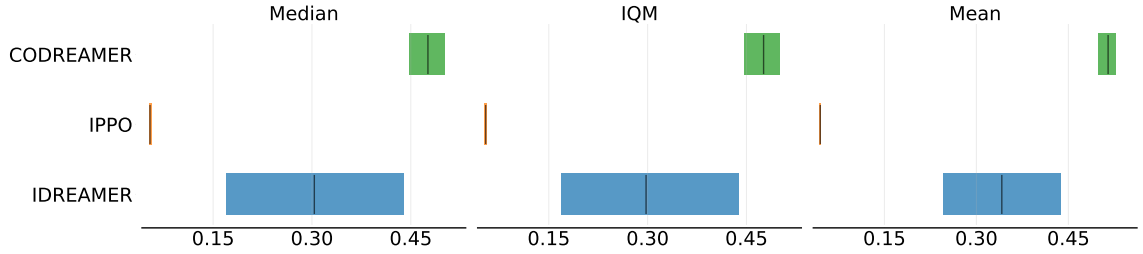

Figure 6: **Aggregate metrics on Melting Pot Tasks** with 95% CIs. Reported from left to right are: Median, IQM, Mean.

In Melting Pot tasks, both IDreamer and CoDreamer significantly outperform IPPO, as shown in Figure 6. IPPO consistently fails to achieve high scores, which aligns with expectations given the sample inefficiency of pixel-based environments. As a model-free on-policy algorithm, IPPO struggles with the limited data (500,000 steps) to learn adequate visual features and high-level strategies. In contrast, CoDreamer outperforms IDreamer in all aggregate metrics with higher normalised scores and less variance, and the results show no CI overlap, indicating high certainty.

The performance profile in Figure 7 shows that IDreamer and CoDreamer significantly outperform IPPO, with IPPO failing to score above 0.1 in any task or run. CoDreamer consistently outperforms IDreamer with low CI overlap, indicating stochastic dominance. Additionally, CoDreamer achieves near-normalised scores of 1.0 in 25% of runs, showing its capability to consistently achieve maximum rewards. Despite the added complexity of communication, CoDreamer demonstrates higher performance with less data, proving the benefit of increased expressivity in learning.

4 Related Work

4.1 MARL With Learned Communication

End-to-end learning of communication protocols has been used to create efficient communication mechanisms without needing significant domain expertise (Sukhbaatar et al., 2016; Foerster et al., 2016; Peng et al., 2017; Kong et al., 2017; Jiang & Lu, 2018).

GNNs have become popular for facilitating communication between agents in various MARL tasks, including cooperative navigation, traffic control, and robotic swarms (Li et al., 2020; Nishi et al., 2018; Tolstaya et al., 2020). These methods efficiently propagate relevant information without breaking the decentralised nature of the system and handle dynamic and heterogeneous environments

[2]Found at https://github.com/google-deepmind/meltingpot

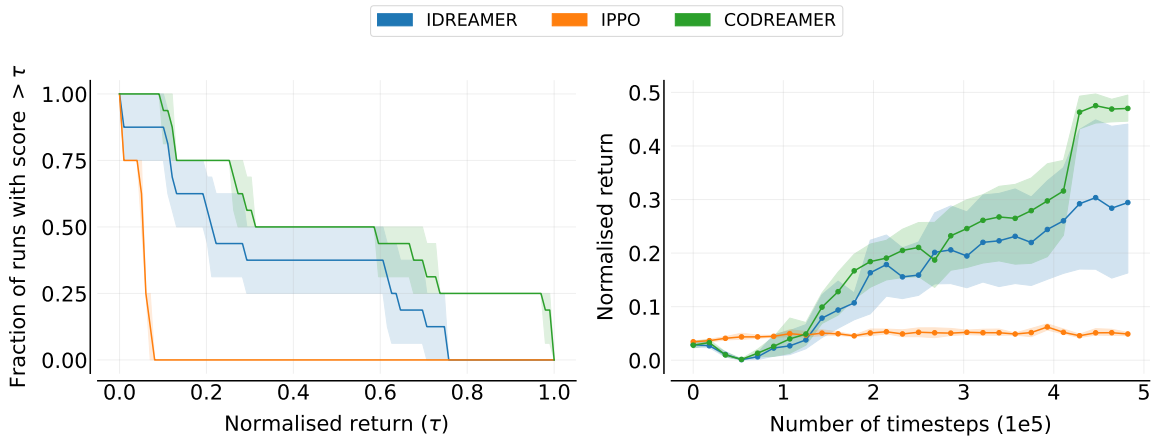

Figure 7: **Melting Pot Evaluation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right:** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all agents. For both plots, the shaded regions show 95% CIs.

(Bettini et al., 2023; Li et al., 2020). However, traditional GNNs have not been used within world models for model-based MARL, setting the stage for CoDreamer.

4.2 Multi-Agent Model-Based RL Methods

Dyna-style methods, like Multi-Agent Model-Based Policy Optimisation (MAMBPO) (Willemsen et al., 2021) and Adaptive Opponent-wise Rollout Policy Optimisation (AORPO) (Zhang et al., 2021), use data from both the real environment and learned models to train agent policies. While some approaches benefit from centralisation, CoDreamer offers a fully decentralised framework. Unlike AORPO, CoDreamer does not need to explicitly model all other agents, as each agent's world model implicitly captures information about others.

Direct-style methods, such as Multi-Agent Model-Based Approach (MAMBA) (Egorov & Shpilman, 2022), learn models from environment data and update agent policies accordingly. CoDreamer incorporates a two-level communication system, utilising GNNs for graph-like structures, and integrates communication into the world models' computation. MAMBA extends DreamerV2 (Hafner et al., 2021) with alterations such as using MAPPO to train policies.

Communication-based methods, like Intention Sharing (IS) (Kim et al., 2021) and Multi-Agent Communication through Imagination (MACI) (Pretorius et al.), use environment models to convey agents' long-term future predictions. While CoDreamer does not specifically use a world model for this purpose, it effectively facilitates communication to create the agents' current state representation for each timestep.

5 Conclusion

In this work, we tackle the issue of performance given a limited sample budget in MARL by exploring model-based RL as a solution. We adapt the single-agent DreamerV3 algorithm (Hafner et al., 2023) into an independent multi-agent variant, IDreamer. Recognizing limitations like partial observability, non-stationarity, and inter-agent dynamics, we introduce CoDreamer, which enhances IDreamer with a two-level GNN-based communication system to improve global observability, handle non-stationarity, and model inter-agent dependencies.

Our evaluations show statistically significant improvements of CoDreamer over IDreamer and the well-known MARL algorithm IPPO, especially in environments with high-dimensional visual obser-

vations. These results demonstrate the potential of CoDreamer and similar model-based MARL methods for sample-efficient learning. The issues it addresses are relevant to real-world applications like multi-robot systems and on-robot learning. We see CoDreamer as a promising step toward scalable, communicative model-based MARL, with potential for significant real-world impact.

References

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.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. *arXiv preprint arXiv:2301.07137*, 2023.

Jan Blumenkamp and Amanda Prorok. The emergence of adversarial communication in multi-agent reinforcement learning. In *Conference on Robot Learning*, pp. 1394–1414. PMLR, 2021.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, February 2022. doi: 10.1038/s41586-021-04301-9. URL https://doi.org/10.1038/s41586-021-04301-9.

Rotem Dror, Segev Shlomov, and Roi Reichart. Deep dominance-how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2773–2785, 2019.

Vladimir Egorov and Aleksei Shpilman. Scalable multi-agent model-based reinforcement learning. *arXiv preprint arXiv:2205.15023*, 2022.

Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning, 2022. URL https://arxiv.org/abs/2212.07489.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Ghost Town Games. Overcooked, 2016. URL https://store.steampowered.com/app/448510/Overcooked/.

Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl. *Advances in Neural Information Processing Systems*, 35:5510–5521, 2022.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pp. 66–83. Springer, 2017.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023.

Auke Jan Ijspeert, Alcherio Martinoli, Aude Billard, and Luca Maria Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11:149–171, 2001.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018.

Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*, 2021.

Xiangyu Kong, Bo Xin, Fangchen Liu, and Yizhou Wang. Revisiting the master-slave architecture in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.07305*, 2017.

Ryan Kortvelesy and Amanda Prorok. Qgnn: Value function factorisation with graph neural networks. *arXiv preprint arXiv:2205.13005*, 2022.

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, pp. 4501–4510, 2020.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Haim Levy. Stochastic dominance and expected utility: Survey and analysis. *Management science*, 38(4):555–593, 1992.

Qimai Li, Zhichao Han, and Xiao-ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32 (1), Apr. 2018. doi: 10.1609/aaai.v32i1.11604. URL https://ojs.aaai.org/index.php/AAAI/article/view/11604.

Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. Graph neural networks for decentralized multi-robot path planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11785–11792. IEEE, 2020.

Jerry Luo, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, et al. Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357*, 2022.

Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pp. 50–60, 1947.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839, 2020.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 16828–16847. PMLR, 2022.

Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 877–883, 2018. doi: 10.1109/ITSC.2018.8569301.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Arnu Pretorius, Scott Cameron, Andries Petrus Smit, Elan van Biljon, Lawrence Francis, Femi Azeez, Alexandre Laterre, and Karim Beguir. Learning to communicate through imagination with model-based deep multi-agent reinforcement learning.

Amanda Prorok. Graph neural networks for learning robot team coordination. *arXiv preprint arXiv:1805.03737*, 2018.

Jenny C. A. Read, Shah Farzana Begum, Alice McDonald, and Jack Trowbridge. The binocular advantage in visuomotor tasks involving tools. *i-Perception*, 4(2):101–110, January 2013. doi: 10.1068/i0565. URL https://doi.org/10.1068/i0565.

Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 25–34, 1987.

Rajarshi Roy, Jonathan Raiman, Neel Kant, Ilyas Elkin, Robert Kirby, Michael Siu, Stuart Oberman, Saad Godil, and Bryan Catanzaro. Prefixrl: Optimization of parallel prefix circuits using deep reinforcement learning. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 853–858. IEEE, 2021.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, December 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038/s41586-020-03051-4.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on robot learning*, pp. 671–682. PMLR, 2020.

Daniël Willemsen, Mario Coppola, and Guido CHE de Croon. Mambpo: Sample-efficient multi-robot reinforcement learning using learned world models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5635–5640. IEEE, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Weinan Zhang, Xihuai Wang, Jian Shen, and Ming Zhou. Model-based multi-agent policy optimization with adaptive opponent-wise rollouts. *arXiv preprint arXiv:2105.03363*, 2021.

A Additional Results

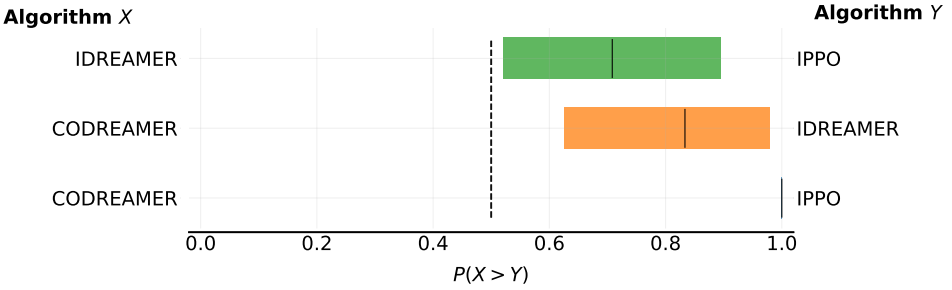

Figure 8: **Probability of Improvement on VMAS Tasks.** Each row shows the probability of improvement, with 95% CIs, that algorithm X outperforms algorithm Y.

Figure 8 shows that CoDreamer has a 100% probability of improvement over IPPO and over 80% probability of outperforming IDreamer. IDreamer also has a high probability of outperforming IPPO. Since all lower bounds of CIs are above 0.5 and the upper bounds are above 0.75, the results are statistically significant and meaningful (Agarwal et al., 2021).

Figure 9 shows similar results in the probability of improvement metrics. IDreamer has a high probability of improvement over IPPO, and CoDreamer has a 100% probability of outperforming IPPO. CoDreamer also has a higher probability of outperforming IDreamer in Melting Pot tasks. Although CoDreamer's performance improvement is larger in magnitude in Melting Pot compared to VMAS, the lower probability of improvement and CI indicates slightly greater uncertainty. However, all probabilities are statistically significant and meaningful.

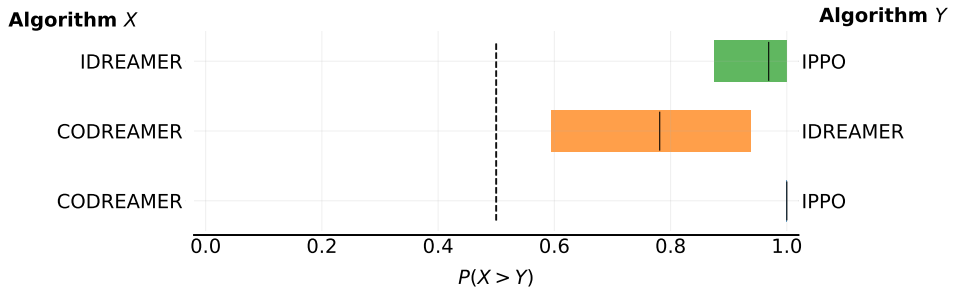

Figure 9: **Probability of Improvement on Melting Pot Tasks.** Each row shows the probability of improvement, with 95% CIs, that algorithm X outperforms algorithm Y.

B Empirical Validation of Modelling Expressivity

To empirically validate our expressivity claims, we present a modified version of the Estimate Game (Kortvelesy & Prorok, 2022) known as the Sequential Estimate Game. This environment extends the original game into a multi-step Partially Observable Markov Game (POMG), which cannot be accurately represented solely based on the information of individual agents.

Commonly used environments (Samvelyan et al., 2019; Ellis et al., 2022; Kurach et al., 2020; Papoudakis et al., 2021) in MARL literature suggest that cooperation and coordination are required to achieve an optimal policy. However, this is often simply conjectured and generally remains unknown. Due to this fact, the original variant of the Estimate Game was proposed to serve as a stress test for multi-agent cooperative systems. The environment is inherently simple yet provably requires coordination through communication to solve by explicitly constructing a dependency for agents to utilise non-local information.

In the original Estimate Game, all n agents are assigned a local state $s^i \in [0, 1]$ where each agent's observation is directly their state $o^i = s^i$. Concurrently, a random adjacency matrix $\mathbf{A}$ is constructed, using an edge density ρ, that represents agent connectivity and defines which agents are interdependent.

The explicit interdependence on non-local information is constructed through the reward function which is defined as follows for agent i at timestep t:

$$y_t^i = 2 \cdot \left(\eta \cdot (s_t^i - 0.5) + (1 - \eta) \cdot \frac{1}{\mathcal{N}^i} \sum_{j \in \mathcal{N}^i} (s_t^j - 0.5) \right) + 0.5$$

$$r_t^i = -max(|\frac{a_t^i}{|\mathcal{A}|} + \frac{1}{2 \cdot |\mathcal{A}|} - y_t^i|) - \frac{1}{2 \cdot |\mathcal{A}|}, 0)$$

where $\mathcal{N}^i$ is agent i's neighbours defined by $\mathbf{A}$.

In simpler terms, this reward function defines the task of agent i predicting the constructed target y^i which is dependent on its neighbours $\mathcal{N}^i$. To make the action space discrete, each agent has a predetermined number of actions $|\mathcal{A}|$ that correspond to equally spaced intervals between 0 and 1. The original Estimate game utilises a global reward to be represented as a Dec-MDP which is the minimum of all local rewards. However, we train agents using their local rewards in order to thoroughly evaluate CoDreamer's ability to learn more independent reward functions that are still dependent on non-local information. Thus, the Sequential Estimate Game can be viewed as a POMG with agent observations being equal to agent local states.

Our extension to the Estimate Game involves the construction of a simple transition function:

$$s_{t+1}^i = \frac{1}{2} \cdot \cos(a_t^i + \eta * s_t^i + (1-\eta) \cdot \frac{1}{\mathcal{N}^i} \sum_{j \in \mathcal{N}^i} s_t^j) + \frac{1}{2}$$

This transition function allows agents to retain unique states without the risk of all agents converging to the same value due to neighbourhood aggregation. Additionally, as it is constructed similarly to the reward function, it introduces an inter-agent dependence. In the sequential variant of the Estimate Game, the randomly generated adjacency matrix $\mathbf{A}$ is held constant for the entirety of an episode thus giving certain agents complete independence in their reward and transition functions. The use of a static adjacency matrix per episode is done to evaluate each method's ability to both model independent agents and inter-dependent agents simultaneously. We list the specific values used in our implementation of the Sequential Estimate Game in Table 1.

Name	**Symbol**	**Value**
Number of Agents	n	4
Number of Timesteps	T	5
Observation Size	$\lvert s_t^i \rvert$	1
Initial State	s_0^i	U(0, 1)
Number of Actions	$\lvert \mathcal{A} \rvert$	4
Edge Density	ρ	0.6
Local State Percentage	η	0.3

Table 1: Specific instantiation of the Sequential Estimate Game evaluated

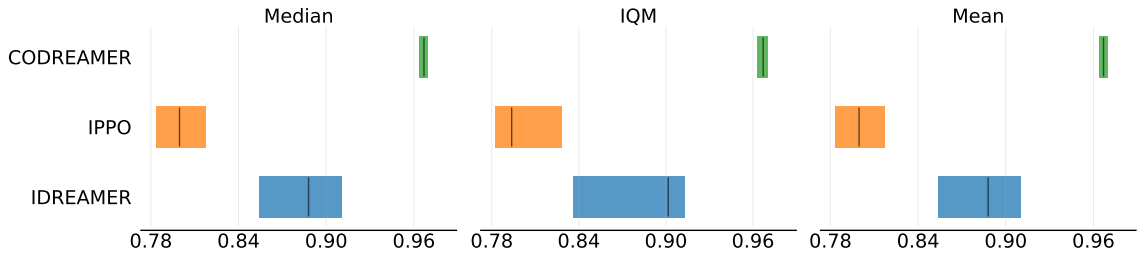

Figure 10: **Aggregate metrics on Sequential Estimate Game** with 95% CIs. Reported from left to right are: Median (↑), IQM (↑), Mean (↑). We use the (↑, ↓) notation to indicate whether higher or lower scores are desired.

We present the final aggregated results in Figure 10. As expected, CoDreamer achieves near-optimal performance with very low variance between runs. Notably, IDreamer outperforms IPPO. We hypothesise that due to IDreamer learning the dynamics for all agents, it is likely aware of the episodes where an agent is independent thus allowing it to learn policies specifically for these agents when possible.

Figure 11 shows the performance profile and sample efficiency of each algorithm. The performance profile confirms our aggregated metrics and shows that CoDreamer has very low variance in its performance over all runs. Additionally, we see that all methods converge to their final performance at around 100,000 - 200,000 timesteps. Interestingly, we see that IDreamer achieves a significant increase in performance at around 300,000 timesteps. This further gives evidence to our hypothesis that, given enough data, IDreamer manages to learn the optimal policy for independent agents specifically. It is possible that through the use of a recurrent IPPO baseline, the same policy would be found.

Although not conventional, we present the relevant loss curves in Figure 12 to further evaluate the world modelling differences between CoDreamer and IDreamer. In the context of the Sequential

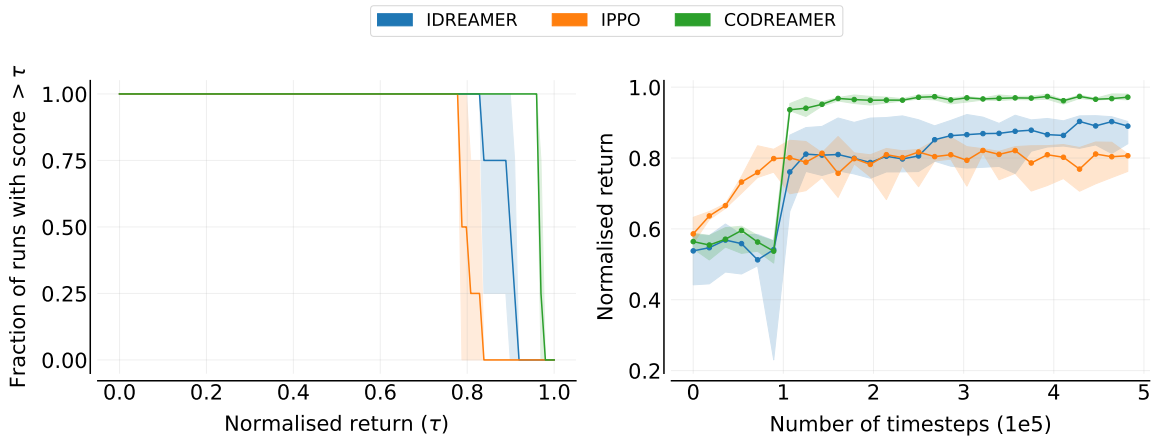

Figure 11: **Sequential Estimate Game Evaluation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right.** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all the agents. For both plots, the shaded regions show 95% CIs.

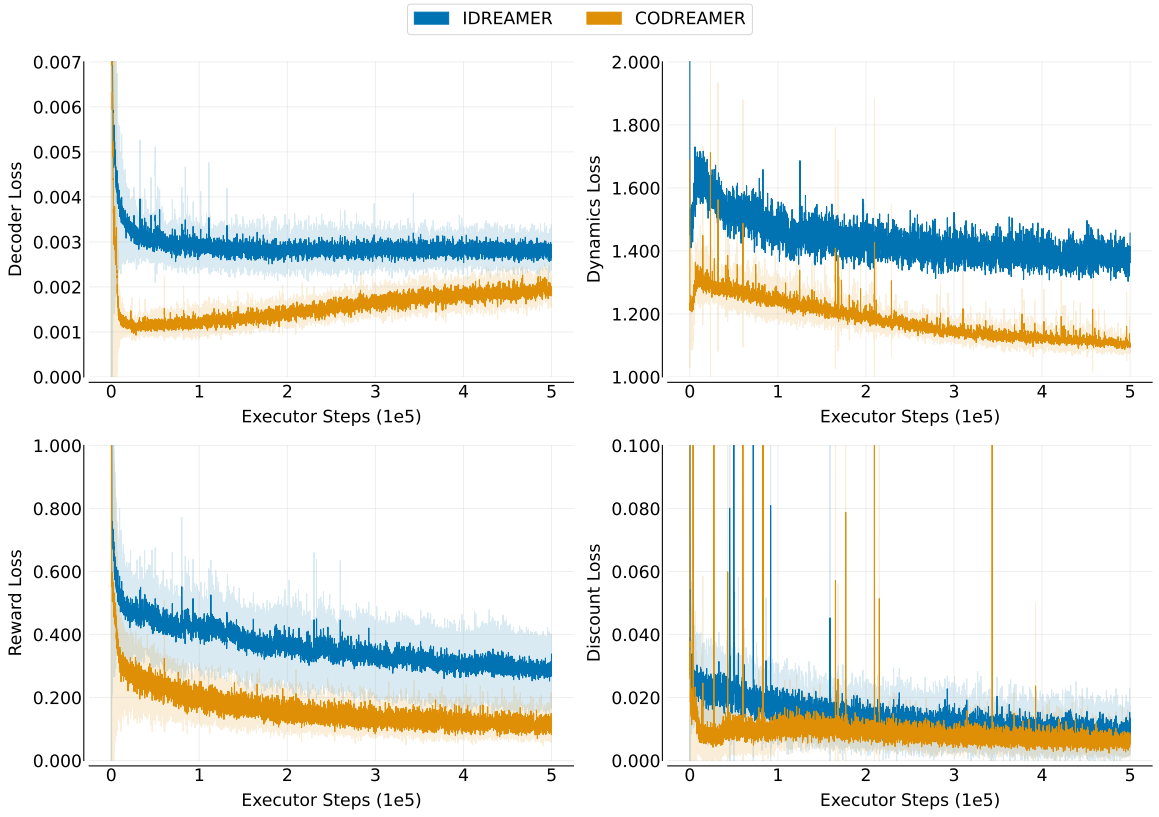

Figure 12: **Mean prediction loss curves** over all Sequential Estimate Game training runs. CoDreamer significantly achieves lower loss values in predicting each respective environment quantity.

Estimate Game, an environment crafted to explicitly have a high degree of inter-agent dependence, we see that CoDreamer achieves much lower losses over the course of training. As each loss directly correlates to prediction accuracy, it is fair to claim that the modelling of the environment is much more accurate using CoDreamer.

Although these results confirm our initial claims, we acknowledge that the Sequential Estimate Game is a highly limited environment that might not be representative of more realistic MARL use cases.

C Ablation

Following our initial set of experiments, we sought to isolate the individual impact of each distinct level of communication within CoDreamer. Consequently, we evaluate a separate set of experiments where only a single level of communication is utilised either within the world model or actor-critic. Particularly, when limiting CoDreamer to use communication exclusively within the world model, our objective is to investigate if the learnt state representation holds enough information for independent actor-critic networks to achieve high performance. For the ablation results, we term the separate levels of communication as WM Comm for the world model and AC comm for the actor-critic.

C.1 Estimate Game

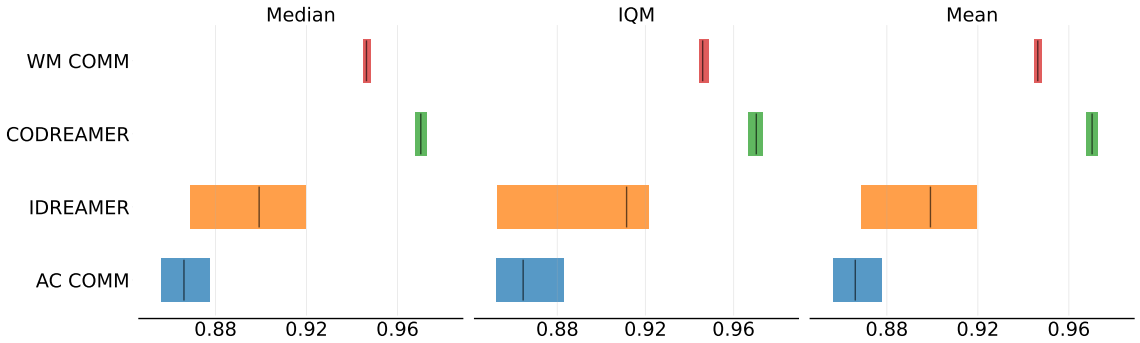

Figure 13: **Aggregate metrics for Sequential Estimate Game Ablation** with 95% CIs. Reported from left to right are: Median (↑), IQM (↑), Mean (↑). We use the (↑, ↓) notation to indicate whether higher or lower scores are desired.

Interestingly, we observe in Figure 13 that IDreamer, which utilises no communication, outperforms AC Comm. This is seemingly due to how Direct-style model-based methods train. If the world model is unable to capture the dynamics and reward functions of the environment, then the learnt representations of the state will be largely incorrect. As this is the case, using communication on top of the inaccurate state representations as well as generating and training on synthetic data will lead to sub-optimal policies. This would naturally make one assume that the performance of IDreamer and AC Comm would perform the same, however, the added complexity of communication most likely inhibits learning from even performing well on more independent agents. Remarkably, we see that utilising communication in the world model exclusively i.e. WM Comm, allows the independent actor-critic networks to perform significantly better than IDreamer. This indicates that there can be a large degree of overlap in the information that is relevant for the world model and that which is relevant for the actor-critic networks. However, we do note that in the particular instance of the Sequential Estimate Game, this overlap of information is highly evident by the construction of the environment.

The performance profiles and sample efficiency plots in Figure 14 give results consistent with those in Section B. We see that all methods converge quickly to their final performance. Both IDreamer and AC Comm have a much higher degree of variance in their performance compared to CoDreamer and WM Comm. This further confirms our hypothesis that both IDreamer and AC Comm obtain their performance through modelling the agents that are seemingly independent, which change every episode thereby introducing potential high variance in returns.

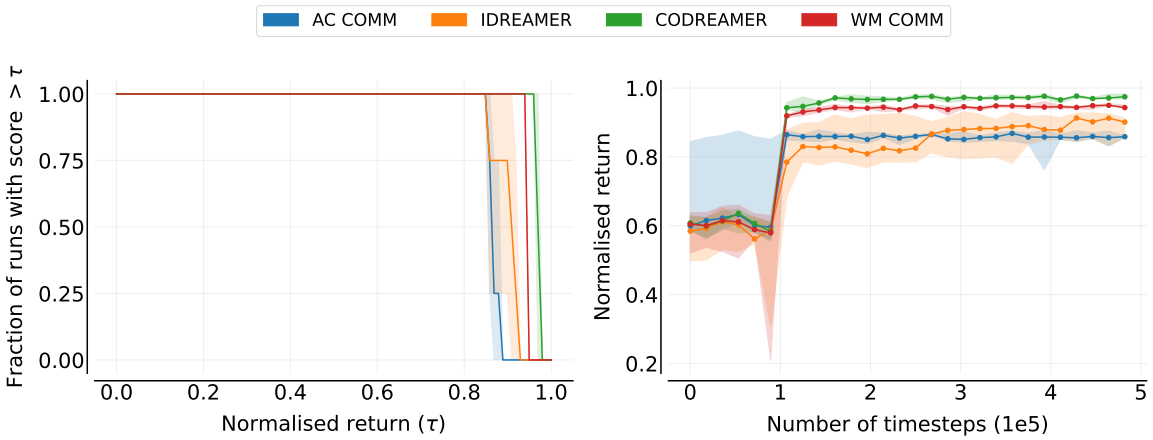

Figure 14: **Sequential Estimate Game Ablation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right.** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all the agents. For both plots, the shaded regions show 95% CIs.

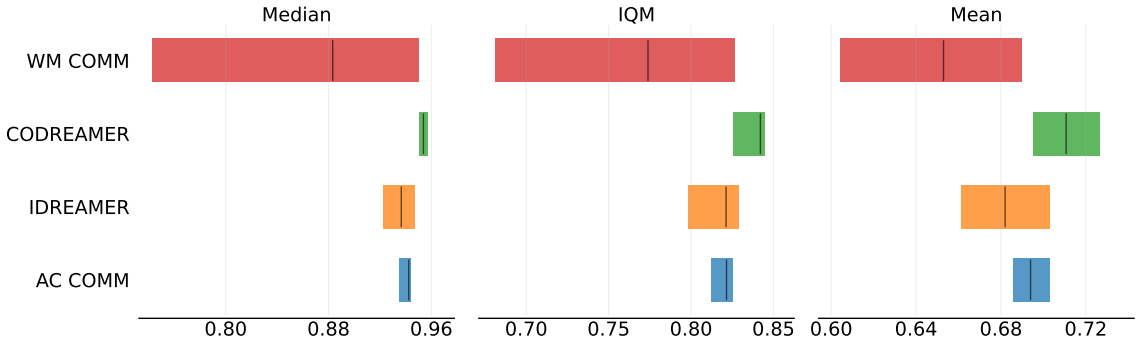

Figure 15: **Aggregate metrics for VMAS Ablation** with 95% CIs. Reported from left to right are: Median (↑), IQM (↑), Mean (↑). We use the (↑, ↓) notation to indicate whether higher or lower scores are desired.

C.2 VMAS

Looking at the aggregate metrics in Figure 15, we see that when exclusively utilising communications within the world model, performance significantly decreases. These results are counter-intuitive as we would expect that a similar pattern to the estimate game would emerge where a more shared state representation could assist the independent actor-critic networks. This seems to indicate that the evaluated VMAS tasks' reward and transition functions are not highly inter-agent dependent thereby not benefitting from the introduced expressivity. However, we do see that when adding communication in the actor-critic networks, performance gains are had yet the results are still close in magnitude to IDreamer. As noted before, CoDreamer outperforms IDreamer which has no communication suggesting that the combination of both levels of communication can yield a unique improvement that no individual part can give. Ultimately, we see from the confidence intervals that IDreamer and CoDreamer are similar in performance and improvements can be due to experiment stochasticity.

The sample efficiency plots presented in Figure 14 demonstrate that IDreamer and AC Comm exhibit similar levels of sample efficiency. We see the same observation for CoDreamer and WM Comm. These findings suggest that communication within the world model acts as the limiting factor in

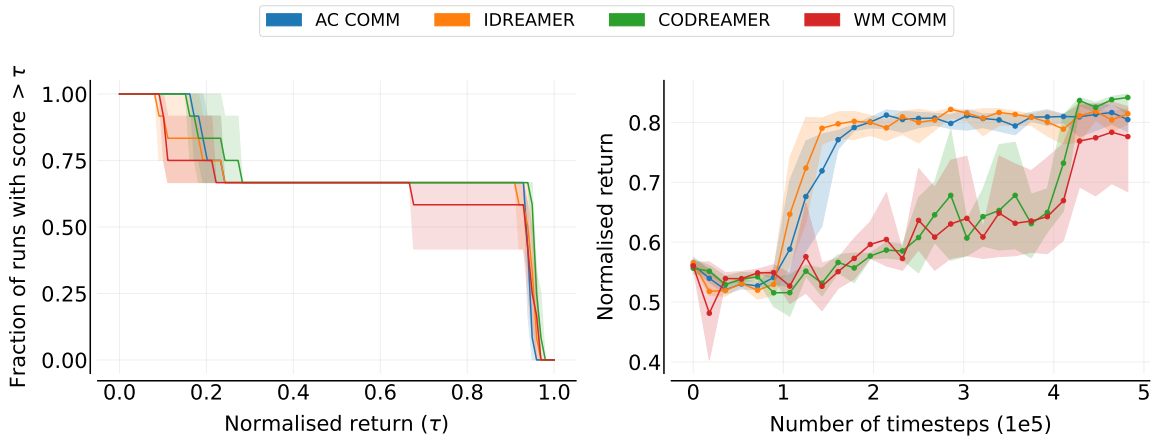

Figure 16: **VMAS Ablation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right.** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all the agents. For both plots, the shaded regions show 95% CIs.

sample efficiency, as increased expressivity negatively impacts early performance during the training process.

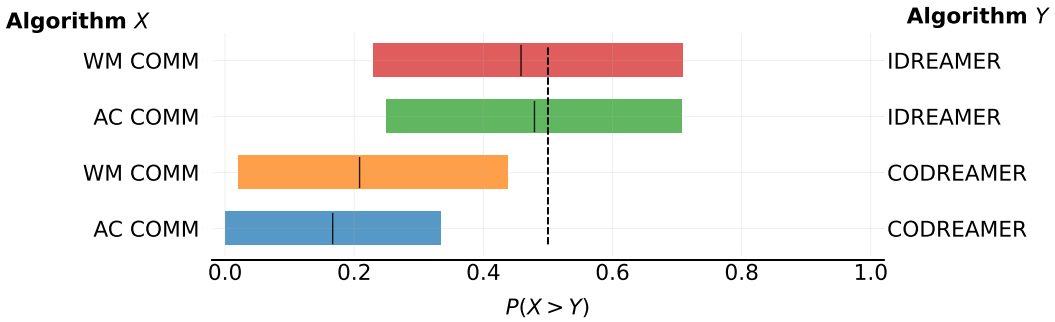

Figure 17: **Probability of Improvement for VMAS Ablation.** Each row shows the probability of improvement, with 95% CIs, that algorithm X outperforms algorithm Y.

In Figure 17, we observe that both AC Comm and WM Comm have a lower probability of improvement compared to IDreamer. However, these results lack statistical significance or meaning as the CIs lower and upper bound do not surpass 0.5 and 0.75 respectively. Thus, we cannot say with certainty that given a random task that IDreamer will perform better. Additionally, we see that both communication levels exhibit a lower probability of improvement compared to CoDreamer, but these differences are statistically significant and meaningful.

C.3 Melting Pot

We see when looking at the results presented in Figure 18, that CoDreamer outperforms all other methods in every point estimate. This indicates that the combination of both levels of communication provides consistent improvements across all tasks in Melting Pot, with no specific outliers. Interestingly, closely following CoDreamer's performance is WM comm and then AC Comm informing us that communication in the world model provided the largest performance gains. WM Comm's CI's have a large overlap with CoDreamer indicating that there is uncertainty in the improvements provided by the second level of communication. This seems to suggest that the state representation

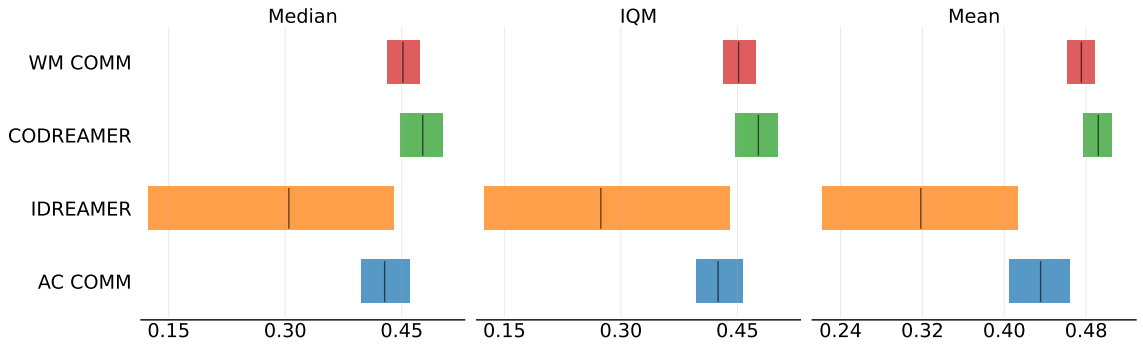

Figure 18: **Aggregate metrics for Melting Pot Ablation** with 95% CIs. Reported from left to right are: Median (↑), IQM (↑), Mean (↑). We use the (↑, ↓) notation to indicate whether higher or lower scores are desired.

learnt is beneficial and provides enough shared information for the independent actor-critic networks to exploit and cooperate without needing their own form of explicit communication. We posit that the Melting Pot environments, due to their visual nature, give each agent's state representation enough information for an independent actor-critic network to know what their teammates are planning on doing, thus simply utilising communications in the world model is enough to improve cooperation.

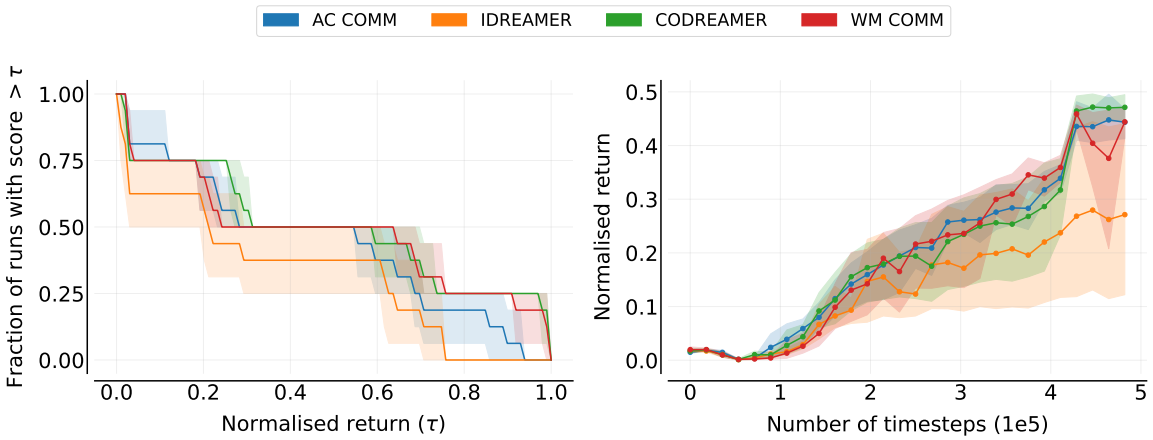

Figure 19: **Melting Pot Ablation. Left:** Performance profiles indicating the percentage of runs that scored above a certain normalised return. **Right.** IQM min-max normalised scores as a function of environment timesteps. This measures the sample efficiency of all the agents. For both plots, the shaded regions show 95% CIs.

The performance profiles in Figure 14 show us that although AC Comm has a slightly higher lower bound in performance, both CoDreamer and WM Comm have higher upper bounds indicating that certain environments do greatly benefit from the communicative world models whereas other environments can be negatively impacted due to the increased complexity albeit marginally. Furthermore, we observe stochastic dominance of IDreamer by all communicative methods. Lastly, We observe in the sample efficiency plots that all communicative methods experience roughly the same sample efficiency with no method being clearly superior.

In the probability of improvement, presented in Figure 20, we see interesting results. Unlike in VMAS, both AC Comm and WM Comm have a probability greater than 0.5 of improving upon IDreamer performance with both methods achieving statistical significance and meaning. When

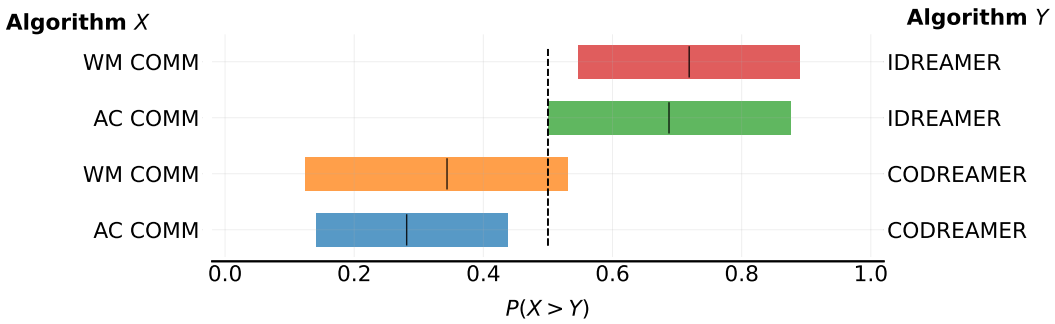

Figure 20: **Probability of Improvement for Melting Pot Ablation.** Each row shows the probability of improvement, with 95% CIs, that algorithm X outperforms algorithm Y.

compared to CoDreamer, we see the opposite relationship with both AC Comm and WM Comm being below the 0.5 value. However, in this instance, we see that WM Comm's result is not statistically significant. What we can interpret from these results is that, regardless of the magnitude of improvement, in Melting Pot tasks, communication in either level of CoDreamer offers improvements, however, we cannot say with statistical certainty that the use of communication within the actor-critic networks, in addition to the world model, improved performance as CoDreamer and WM Comm perform similarly.

D Experimental Details

D.1 Evaluation Methodology

In recent times, there has been a noticeable trend towards evaluating RL algorithms on larger suites of tasks with comparisons being made on aggregate performance point estimates, such as mean or median, for each task independently. However, these measures often neglect the inherent statistical uncertainty that comes with evaluations based on a limited number of training runs and seeds. Furthermore, a significant portion of research has started to focus on highly computationally-intensive benchmarks whereby each training run can last from a few hours to several weeks. Due to this, it is computationally impractical to evaluate a large number of runs per task thereby further increasing the statistical uncertainty associated with the reported metrics. To mitigate this uncertainty and obtain more statistically validated results, we adopt the evaluation methodology presented in Agarwal et al. (2021) and Gorsane et al. (2022) to assess the performance of IDreamer and CoDreamer as well as a model-free baseline IPPO.

For each algorithm, we conduct evaluations across M tasks within a specific environment suite, utilising $N = 4$ independent training runs per task $m \in M$. During each training run $n \in N$, we assess algorithm performance over $E = 32$ distinct evaluation episodes at intervals of 10,000 environment timesteps. At each interval i, we compute the mean return $G^i_{m,n}$ per agent, over the E episodes. Additionally, at each evaluation interval, we employ model checkpointing to save the best-performing model. The model with the largest mean return achieved over all intervals is retained for a subsequent final evaluation.

Upon completing a training run $n \in N$, we evaluate the best model, discovered during interval evaluations, over $10 \times E = 320$ episodes. This evaluation process yields normalised scores $x_{m,n}$, for $m = 1, \ldots, M$ and $n = 1, \ldots, N$, acquired by scaling each per-task score based on the minimum and maximum scores observed throughout all training runs in the specific task. Consequently, we obtain a set of normalised scores $x_{1:M,1:N}$ for each algorithm. To aggregate the performance of an algorithm, we map the set of normalised scores into a singular scalar point estimate i.e. $x_{1:M,1:N} \rightarrow \bar{x}$.

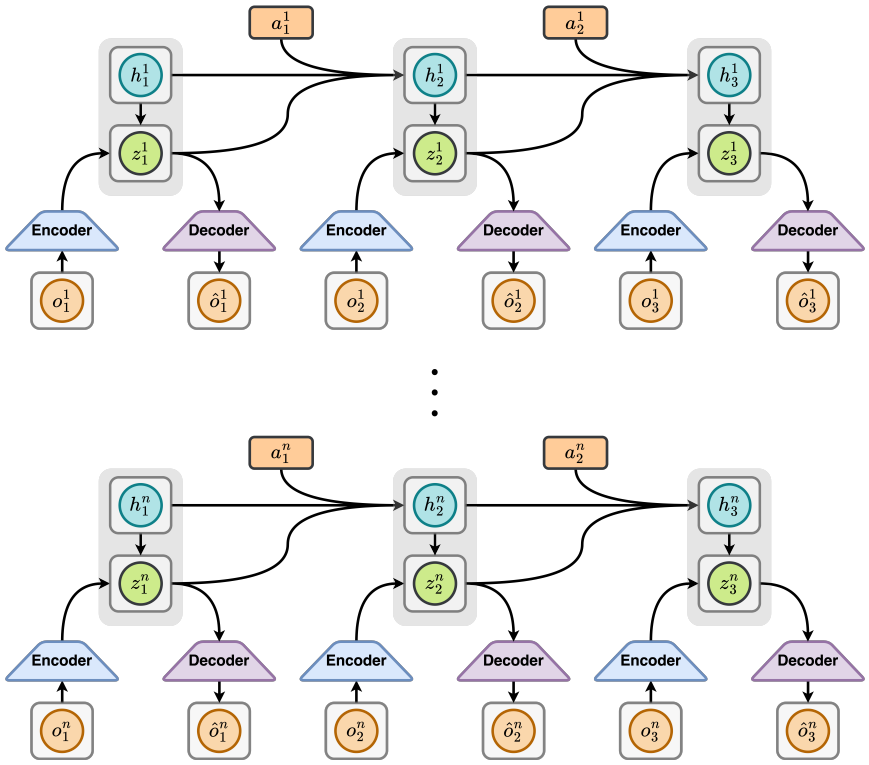

Figure 21: **Training process of IDreamer world model:** Each agents' world model encodes their respective observation o_t^i and RSSM recurrent state h_t^i into the discrete stochastic state z_t^i known as the posterior state. Additionally, at each step, the world model for agent i produces a prior state $\hat{z}_t^i$ that is predicted solely by the RSSM recurrent state h_t^i and trained towards z_t^i. Given h_t^i, z_t^i, and the action a_t^i, the next recurrent state is produced h_{t+1}^i. Each agent's posterior state z_t^i is used to reconstruct the observation o_t^i in order to learn a better representation. The sequence of observations $o_{t:T}^i$ is unrolled over time.

By following this methodology, we consolidate all independent tasks M and training runs N of an environment suite into a single comprehensive score and utilise a 95% Confidence Interval (CI) obtained from stratified bootstrapping over all $M \times N$ experiments treated as *random samples*. This ensures that although a large number of runs cannot be obtained for each task on its own, by performing statistical analysis over all runs of all tasks, we emulate the statistical confidence obtained when conducting a large number of runs over a single task whilst mitigating a lack of task diversity. Consequently, we report the results for each environment suite as a whole, rather than the tasks independently.

D.2 Metrics

Using the set of normalised scores, in addition to traditional aggregate point estimates such as median and mean, we employ the following metrics to measure algorithmic performance:

- **Interquartile Mean (IQM):** We utilise the IQM, which ignores both the lower 25% and upper 25% of all runs, calculating the mean normalised score of the central 50% ($\frac{N \cdot M}{2}$). This metric is more robust to outliers in training runs than a conventional *mean* and has less bias when compared to *median*. Additionally, it has been shown that IQM has greater statistical efficiency than *median* and is thereby able to detect algorithmic improvements using fewer training runs (Agarwal et al., 2021).

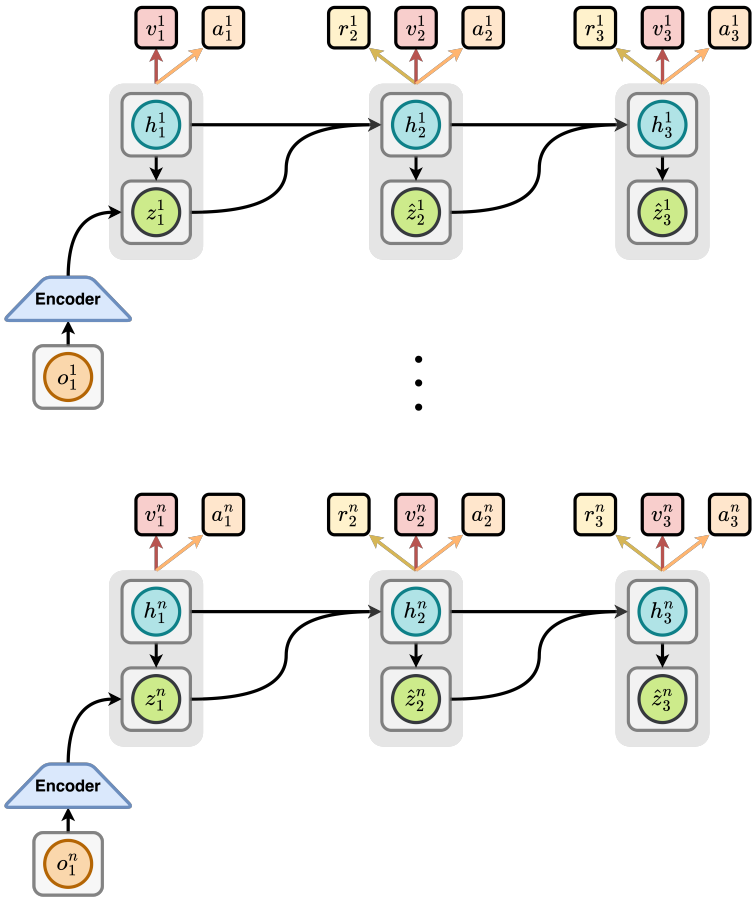

Figure 22: **Training process of IDreamer Actor-Critic:** Each agents' world model generates synthetic trajectories by auto-regressively predicting the next discrete $\hat{z}_t^i$ and recurrent state h_t^i. These states form the compact world model state for each agent's actor-critic networks.

- **Probability of Improvement:** The probability of improvement quantifies the likelihood of algorithm X outperforming algorithm Y on a **randomly** chosen task m. This metric is useful in quickly identifying how robust the improvement an algorithm brings. This metric uses the Mann-Whitney U-statistic (Mann & Whitney, 1947) for its computation.

  To formally elaborate, we give the following definitions:

$$Pr(X > Y) = \frac{1}{M} \sum_{m=1}^{M} Pr(X_m > Y_m)$$

  where $Pr(X_m > Y_m)$ is the probability of algorithm X performing better on a **specific** task m.

  Furthermore, the metric $Pr(X_m > Y_m)$ is defined as:

$$Pr(X_m > Y_m) = \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K} S(x_{m,i}, y_{m,j})$$

  where

$$S(x, y) = \begin{cases} 1, & \text{if } y < x \\ \frac{1}{2}, & \text{if } y = x \\ 0, & \text{if } y > x \end{cases}$$

When interpreting the probability of improvement metric, as per the Neyman-Pearson statistical testing criterion outlined by Bouthillier et al. (2021), if the **lower** bound of the CI is greater than the null hypothesis of $Pr(X > Y) = 0.5$, then the result is statistically significant. In addition to statistical significance, if the **upper** bound of the CI exceeds the recommended threshold of 0.75, then the result is statistically meaningful.

Additionally, we present algorithm performance using performance profiles, which visually display the entire set of normalised scores $x_{1:M,1:N}$ and provide enhanced insights into performance variability across tasks compared to interval estimates of aggregate metrics. Specifically, the performance profile reports the fraction of runs that obtained a normalised score above a certain threshold. This allows us to easily perform a visual comparison of methods and determine if one method stochastically dominates another. In the context of performance profiles, if one curve is strictly positioned above another, it is interpreted as having stochastic dominance[3] over the other (Levy, 1992; Dror et al., 2019; Agarwal et al., 2021). The performance profiles also allow us to better identify the empirical lower and upper bounds on the performance of an algorithm. Finally, to evaluate sample efficiency, we compute the IQM score of each interval evaluation and plot it against the number of environment steps taken.

D.3 Implementation Specifics

Each algorithm is trained for 500,000 environment timesteps collected by 8 independent workers. All hyperparameters and network sizes are listed in Section D.4 and D.5 in the appendix. Additionally, for both IDreamer and CoDreamer, we perform 500 training steps of the world model before any behaviour is learnt. This form of *pre-training* is performed in order to avoid negative performance effects due to a primacy bias, a common RL flaw examined by Nikishin et al. (2022), when learning on inaccurate world model states.

As an on-policy algorithm, IPPO is typically considered to be less sample efficient compared to its off-policy counterparts. However, PPO makes use of off-policy correction techniques to enable the reuse of recently collected data for multiple training steps. To enhance sample efficiency, we increase the number of epochs and mini-batches within the algorithm, enabling more training steps per batch of collected data. The specific values used can be found in Section D.5.

D.4 Model Sizes

Dimension	**Size**
GRU recurrent units	512
CNN multiplier	32
Dense hidden units	512
MLP layers	2
RSSM GNN Layers	1
Reward & Cont GNN Layers	1
Visual Environment Parameters	**22M**
Vector Environment Parameters	**16M**

Table 2: World Model Size

We list the model sizes used in Tables 2, 3, and 4.

Dimension	**Size**
Dense hidden units	512
MLP layers	2
GNN Layers	1
Actor Parameters	**1.6M**
Critic Parameters	**1.7M**

Table 3: Actor-Critic Model Size

Dimension	**Size**
Dense hidden units	512
MLP layers	2
Actor Parameters	**1.6M**
Critic Parameters	**1.7M**

Table 4: PPO Model Size

Name	**Symbol**	**Value**
General		
Replay capacity (FIFO)	-	10^6
Batch size	B	16
Batch length	T	64
Activation	-	LayerNorm + SiLU
World Model		
Number of latents	-	32
Classes per latent	-	32
Reconstruction loss scale	β_{pred}	1.0
Dynamics loss scale	β_{dyn}	0.5
Representation loss scale	β_{rep}	0.1
Learning rate	-	10^{-4}
Adam epsilon	ϵ_a	10^{-8}
Gradient clipping	-	1000
Actor Critic		
Imagination horizon	H	15
Discount Factor	γ	0.997
Return lambda	λ	0.95
Target Critic Polyak Averaging Step	-	0.02
Actor entropy scale	η	$3 \cdot 10^{-4}$
Learning rate	-	$3 \cdot 10^{-5}$
Adam epsilon	ϵ_a	10^{-5}
Gradient clipping	-	100

Table 5: Hyperparameters used in all experiments for both CoDreamer and IDreamer.

Name	Symbol	Value
General		
Queue capacity (FIFO)	-	1000
Batch size	B	64
Batch length	T	64
Activation	-	ReLu
Algorithm		
Learning rate	-	$10^{-3} - 5 \cdot 10^{-5}$
Learning rate Schedule	-	Linear (10,000 training steps)
Adam epsilon	ϵ_a	10^{-5}
Gradient clipping	-	1.0
Number of Epochs	-	30
Number of Minibatches	-	32
Discount Factor	γ	0.997
Clipping Epsilon	ϵ_c	0.3
Actor entropy scale	η	10^{-2}
Value loss coefficient	-	0.5
GAE lambda	λ	0.95

Table 6: Hyperparameters used in all experiments for IPPO.

D.5 Hyperparameters

We list the hyperparameters used in Tables 5 and 6.

D.6 Evaluation Details

As recommended by Agarwal et al. (2021), for our aggregate point estimates confidence intervals, we use stratified bootstrapping with 50,000 bootstrap replications. For our probability of improvement and performance profiles, we use 2000 bootstrap replications. Lastly, for our sample efficiency plots, we use 5000 bootstrap replications.

Environment	Task	Minimum Observed Return	Maximum Observed Return
Estimate Game	Sequential	-3.13	0.00
VMAS	Flocking	-37.65	2.58
VMAS	Discovery	0.00	14.20
VMAS	Buzz Wire	-29.97	1.81
Melting Pot	Daycare	0	189.00
Melting Pot	Cooperative Mining	8.00	1003.00
Melting Pot	Collaborative Cooking: Asymmetric	0.00	2514.00
Melting Pot	Collaborative Cooking: Forced	0.00	49.00

Table 7: Observed Min-Max scores used for normalisation

We list the minimum and maximum scores observed and used for normalisation in Table 7.

Environment	Task	Observation Spec.	Action Spec.	No. of Agents	Avg. Time Horizon
VMAS	Flocking	Vector (18)	Discrete (5)	4	500
VMAS	Discovery	Vector (21)	Discrete (5)	5	500
VMAS	Buzz Wire	Vector (8)	Discrete (5)	2	500
Melting Pot	Daycare	Pixels (64, 64, 3)	Discrete (8)	2	1000
Melting Pot	Cooperative Mining	Pixels (64, 64, 3)	Discrete (8)	4	1500
Melting Pot	Collaborative Cooking: Asymmetric	Pixels (64, 64, 3)	Discrete (8)	2	1000
Melting Pot	Collaborative Cooking: Forced	Pixels (64, 64, 3)	Discrete (8)	2	1000

Table 8: External Environment Summary

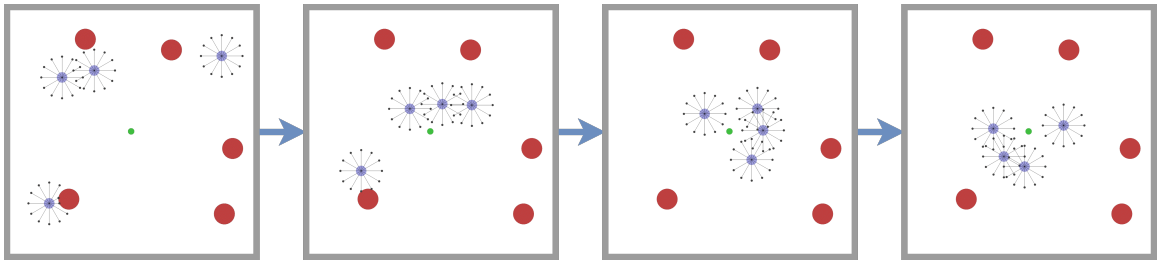

Figure 23: **VMAS Flocking** environment. Example illustration showcasing 4 agents surrounding a target position whilst avoiding obstacles

D.7 Evaluation Environments Description

D.7.1 Flocking

The Flocking task (see Figure 23) generates an open environment containing M randomly positioned obstacles. N agents are situated within this environment, tasked with encircling a moving target whilst avoiding collisions with the obstacles as well as one another. Flocking has been a long-standing benchmark in the field of robotic coordination (Reynolds, 1987) and serves as an ideal challenge due to the complexity of coordinating multiple agents. We specify our instantiation of the Flocking task in Table 9.

Name	Symbol	Value
Number of Agents	N	4
Number of Obstacles	M	5

Table 9: Specific instantiation of VMAS Flocking

D.7.2 Discovery

The Discovery task, drawing inspiration from the Stick Pulling experiment (Ijspeert et al., 2001), places N agents in an open setting with M objectives. The agents are assigned to cover as many objectives as they can while avoiding collisions. An objective is considered satisfied when K agents are within a predetermined distance D from it. After a goal has been covered by K agents, each of them is rewarded. It has been shown that the performance can be significantly improved through communication when the number of agents, N, is less than the number of goals, M. We specify our instantiation of the Discovery task in Table 10.

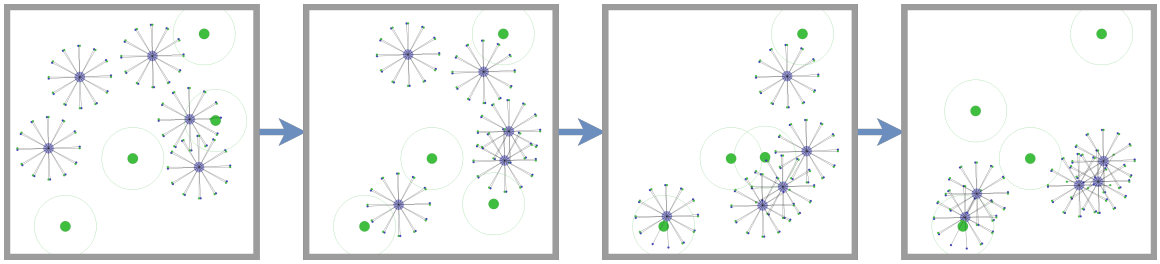

Figure 24: **VMAS Discovery** environment. Example illustration showcasing 5 agents covering 7 objectives.

Name	Symbol	Value
Number of Agents	N	5
Number of Objectives	M	7
Coverage Requirement	K	2
Coverage Distance	D	0.25

Table 10: Specific instantiation of VMAS Discovery

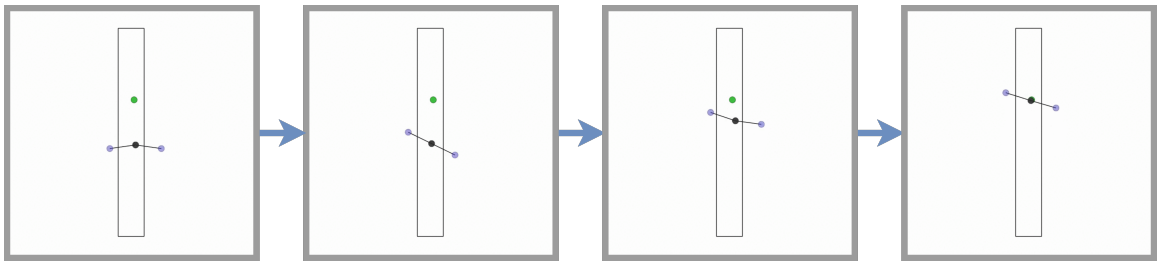

Figure 25: **VMAS Buzz Wire** environment. Example illustration showcasing 2 agents successfully reaching the target.

D.7.3 Buzz Wire

The Buzz Wire task, based on the popular "Wire Loop" game (Read et al., 2013), requires two agents to steer a central mass via attached linkages through a straight corridor to a designated target. Both agents are unable to touch the borders of the corridor otherwise they fail the task and the episode ends, introducing a fair level of difficulty. This task necessitates a high level of awareness and coordination between the agents, as a lack of synchronised movement can result in one agent pulling the other into the borders thereby causing failure.

D.7.4 Daycare

Daycare is a straightforward two-player game in which agents occupy one of two roles: *parent* or *child*. In Daycare, two types of fruit can grow on either shrubs or trees. The parent can pick and consume any fruit from a tree or shrub whereas the child can only pick fruits on shrubs and only consumes one specific type of fruit. Each agent is rewarded equally for consuming fruit. If the child does not consume a fruit within 200 timesteps, it throws a "tantrum" and is temporarily removed from the game for 100 timesteps. Whilst the child is removed, the parent cannot gain any reward, thus, two challenges exist: First, the parent must take advantage of its unique affordances and assist

[3] A random variable X is defined as having stochastic dominance over another random variable Y if $P(X > \tau) \geq P(Y > \tau)$ for all τ, and for at least some τ there is $P(X > \tau) > P(Y > \tau)$. Intuitively this means that if we sample a random value from X, it is likely to be larger than a random value sampled from Y.

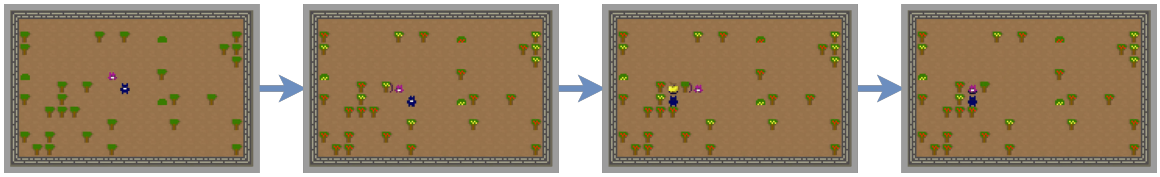

Figure 26: **Melting Pot: Daycare** environment. Example illustration the parent and child agents collecting food.

the child to pick the specific type of fruit for it from trees. Second, the child needs to convey its fruit preference to the parent as it can change from episode to episode.

D.7.5 Cooperative Mining

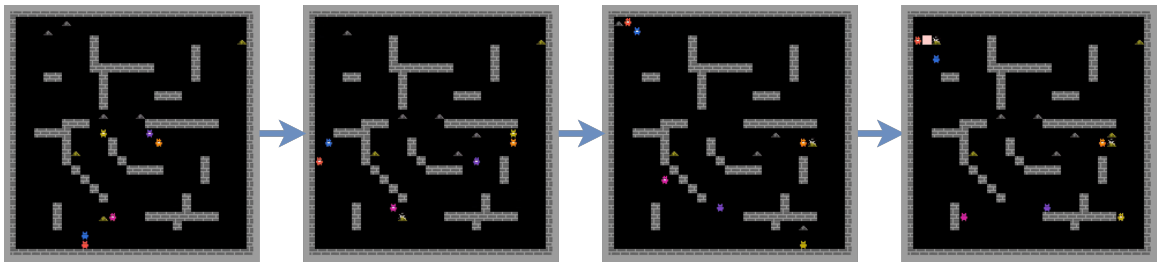

Figure 27: **Melting Pot: Cooperative Mining** environment. Example illustration showcasing four agents working together to mine Iron and Gold ores.

Cooperative Mining is originally a six-player game designed to assess the effectiveness of cooperation in multi-agent systems. In this game, two types of ore, Iron and Gold, spawn randomly in empty spaces throughout the map. When an agent mines Iron ore, they receive a reward of 1, and this extraction process requires no collaboration. In contrast, Gold ore demands the coordinated efforts of two agents for successful mining, granting a reward of 8 to each participant. When an agent initiates mining gold, the ore flashes, signalling other agents to mine within a 3-timestep window. If no other agent attempts mining or too many agents engage, the gold ore reverts to its regular state and no rewards are given.

By encouraging agents to maintain close proximity and collaborate in mining gold, higher rewards can be achieved compared to mining iron individually. Additionally, through the communication of ore locations and mining intentions, the global reward of the system can be further increased. Thus, the Cooperative Mining game serves as a valuable benchmark for evaluating cooperation and coordination in multi-agent learning algorithms. In this work, we utilise only four agents instead of six.

D.7.6 Collaborative Cooking

Based on the game *Overcooked* (Games, 2016), Melting Pot's Collaborative Cooking is a unique environment where multiple players work together to follow recipes and serve food to customers. This task involves agents creating soup by cooking three tomatoes in a pot for 20 timesteps, followed by plating and serving to customers. Depending on the map configuration and the number of players, these scenarios can demand a high degree of collaboration and cooperation to achieve sufficient performance. In this work, we focus on two specific scenarios: **Asymmetric** and **Forced**.

D.7.7 Asymmetric

In the Asymmetric scenario, two players are separately placed in non-connected parts of the map with differing proximity to plates, pots, and tomatoes. For one player, the goal delivery spot is

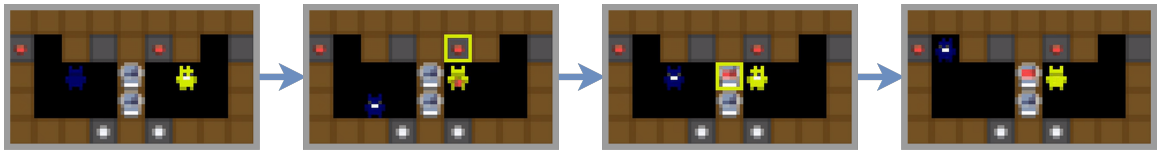

Figure 28: **Melting Pot: Collaborative Cooking - Asymmetric** environment. Example illustration of the asymmetric scenario's map layout.

located close to the cooking pots, but far from the tomato dispenser. Conversely, the second player has the goal delivery spot far from their pots, but close to the tomato dispenser. This setup allows both players to function independently to some extent, but they can achieve a significantly greater reward if they learn to collaborate and leverage their individual advantages by specialising to their side of the map.

D.7.8 Forced

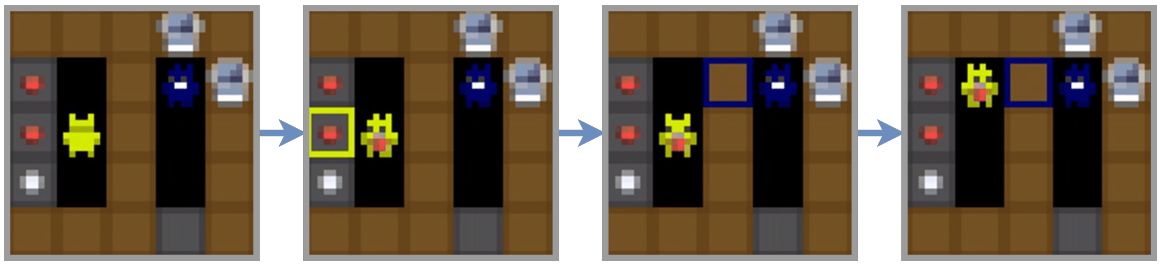

Figure 29: **Melting Pot: Collaborative Cooking - Forced** environment. Example illustration of the Forced scenario's map layout.

The Forced scenario, like the Asymmetric one, features two players on separate and non-connected parts of the map. However, differing from Asymmetric, each player has exclusive access to certain resources. One player has access to the tomato and plate dispensers, while the other has access to the cooking pots and the drop-off point. Thus, to earn any reward in this setup, the players must work together and cooperate effectively.

D.8 Environmental Impact

In the final evaluation of this work, we ran 5 different algorithms: IPPO, IDreamer, CoDreamer, AC Comm, and WM Comm. Each of these algorithms was evaluated for 4 runs on 7 tasks each. This equates to $5 \times 4 \times 7 = 140$ experimental training runs. Each training run lasted for approximately 6 to 12 hours. Taking the average experiment time of 9 hours, we utilised hardware accelerators for a total of $140 \times 9 = 1260$ hours.

All experiments were conducted using TPU v2 and v3 chips in the Europe-west4 and us-central1 regions on the Google Cloud Platform, which has an approximate carbon efficiency of 0.57 $kgCO_2eq/kWh$. The TPUv2 and v3 chips have an approximate power draw of 221W and 283W respectively. Using the average between these, we set the power draw of the hardware used to 252W.

Given these values, total carbon emissions are estimated to be:

$$252\text{W} \times 1260\text{h} = 317.52\text{kWh} \times 0.57\ \text{kgCO}_2\text{eq/kWh} = 180.99\ \text{kgCO}_2\text{eq}$$

These estimates are exclusively calculated for the final evaluation and not for any experiments run during the development of the proposed methods.

These estimations were in part conducted using the MachineLearning Impact calculator as presented in Lacoste et al. (2019).

D.9 Supplementary Evaluation Information

	IDREAMER	IPPO	CODREAMER
Median	0.89 [0.85, 0.91]	0.80 [0.78, 0.82]	0.97 [0.96, 0.97]
IQM	0.90 [0.84, 0.91]	0.79 [0.78, 0.83]	0.97 [0.96, 0.97]
Mean	0.89 [0.85, 0.91]	0.80 [0.78, 0.82]	0.97 [0.96, 0.97]
Optimality Gap	0.11 [0.09, 0.15]	0.20 [0.18, 0.22]	0.03 [0.03, 0.04]

Table 11: Aggregated Results for **Sequential Estimate Game**

	IDREAMER	IPPO	CODREAMER
Median	0.91 [0.90, 0.92]	0.92 [0.92, 0.93]	0.93 [0.93, 0.94]
IQM	0.81 [0.79, 0.82]	0.79 [0.78, 0.79]	0.84 [0.82, 0.84]
Mean	0.68 [0.66, 0.71]	0.65 [0.65, 0.66]	0.72 [0.70, 0.74]
Optimality Gap	0.32 [0.29, 0.34]	0.35 [0.34, 0.35]	0.28 [0.26, 0.30]

Table 12: Aggregated Results for **VMAS**

	IDREAMER	IPPO	CODREAMER
Median	0.30 [0.17, 0.44]	0.05 [0.05, 0.06]	0.48 [0.45, 0.50]
IQM	0.30 [0.17, 0.44]	0.05 [0.05, 0.06]	0.48 [0.45, 0.50]
Mean	0.34 [0.25, 0.44]	0.05 [0.04, 0.05]	0.51 [0.50, 0.53]
Optimality Gap	0.66 [0.56, 0.75]	0.95 [0.95, 0.96]	0.49 [0.47, 0.50]

Table 13: Aggregated Results for **Melting Pot**

We list the results obtained in Table 11, 12, and 13. Additionally, we present the plots for each task in Figures 30 and 31.

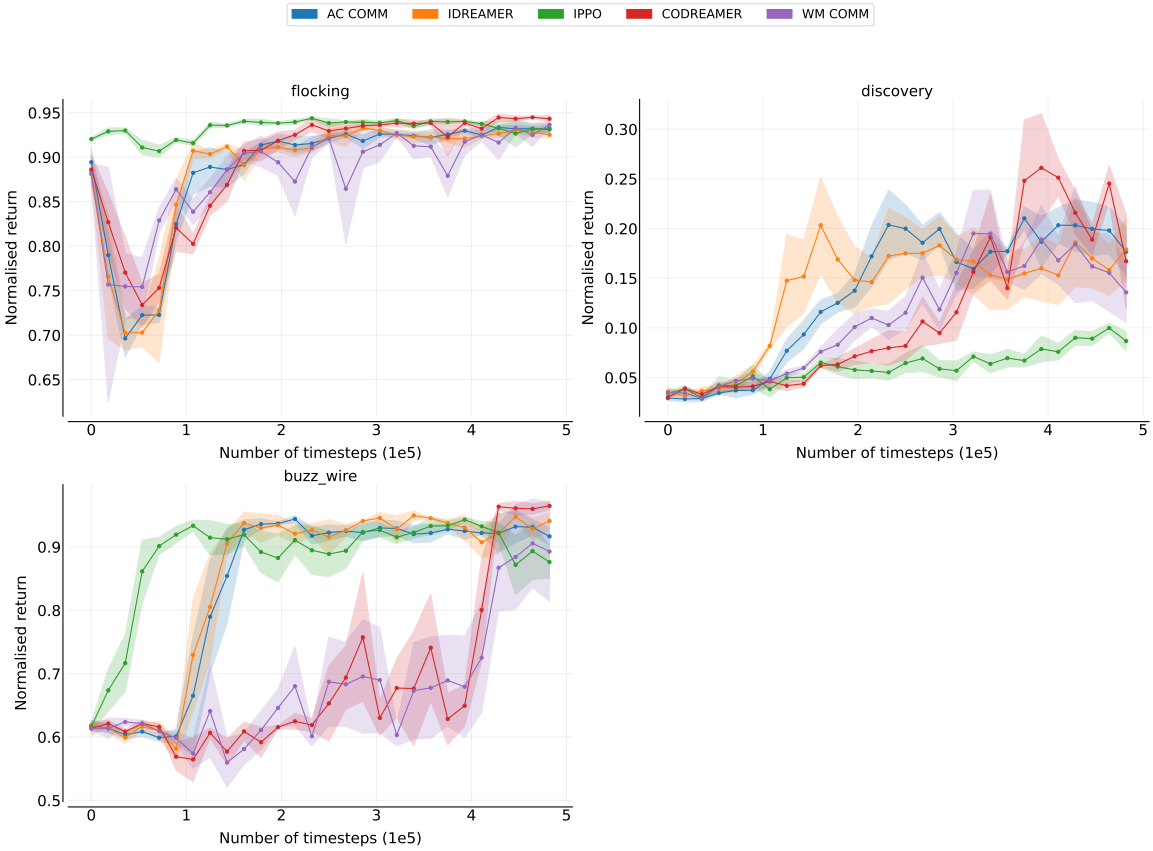

Figure 30: Individual IQM plots for each task in **VMAS**

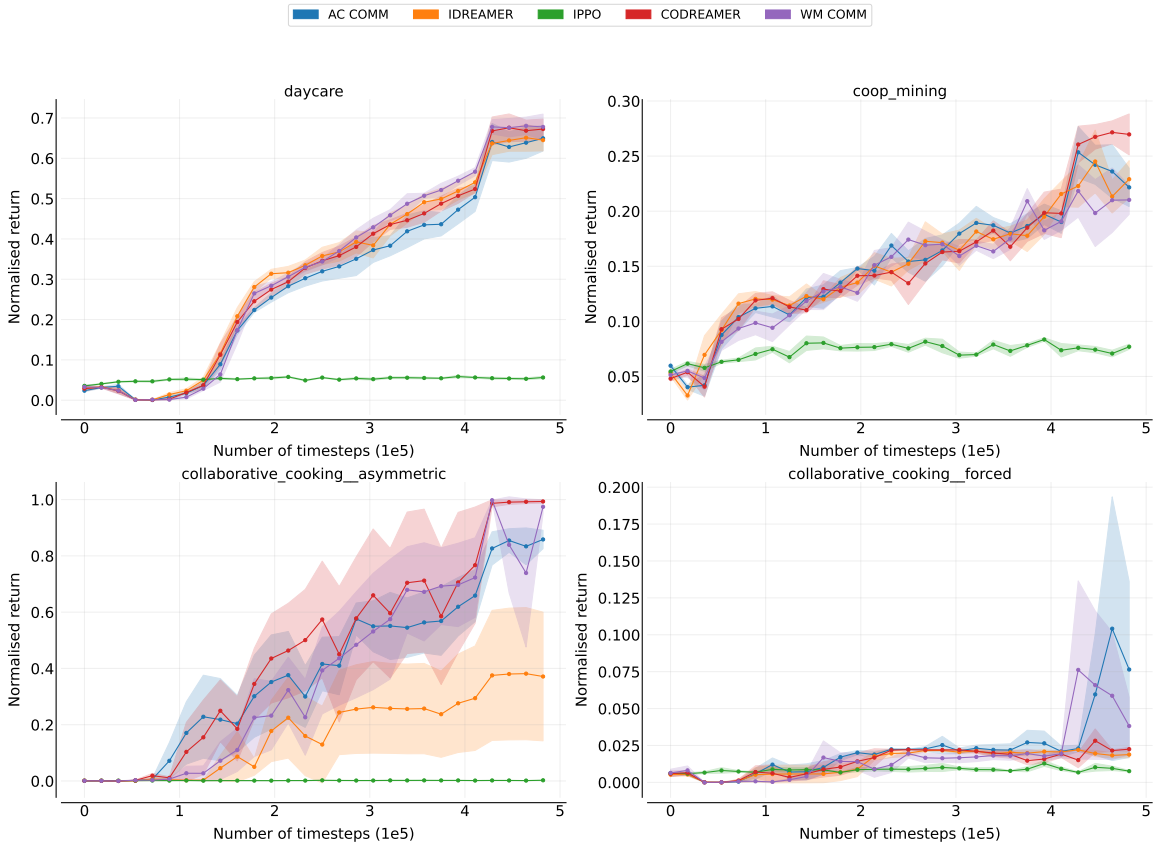

Figure 31: Individual IQM plots for each task in **Melting Pot**