Extended Abstract

Motivation Multi-agent reinforcement learning (MARL) systems face a fundamental challenge in competitive environments: how to enable teams of autonomous agents to develop specialized roles and coordinate adaptively against other teams. This challenge was highlighted by the 2024 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) Maritime Capture the Flag competition, where deep MARL approaches were outperformed by heuristic and stochastic search methods. We aim to improve team coordination and adaptation as measured by performance against heuristic opponents and sample efficiency in on-policy learning.

Method We introduce continuous latent role embeddings to MAPPO that enable agents to dynamically specialize their behavior based on local observation while maintaining shared policy parameters. The role encoder maps each agent's local observations to a Gaussian latent distribution, with sample role vectors concatenated to observations and fed to a conditional policy actor network. This allows the same parameter set to specialize into distinct behavioral roles on-the-fly. Roles are updated for agents every 48 time steps based on the most recent observation to enable more stable learning. We incorporate KL divergence regularization to prevent role distribution drift and implement role diversity loss to encourage behavioral differentiation between agents.

Implementation We establish a curriculum of increasingly difficult heuristic opponents. Agents advance to the next stage of the curriculum upon reaching specified success thresholds within a moving window of 50 episodes. We compare our role-based MAPPO to baseline MAPPO on this curriculum to obtain a metric of on-policy sample efficiency. We then evaluate the trained models against heuristic policies of varying difficulty levels, measuring win rates.

Results Role-based MAPPO achieved improved sample efficiency on the curriculum relative to baseline MAPPO, advancing through difficult heuristic stages faster than baseline MAPPO. Role-based MAPPO also advanced further through the curriculum than baseline MAPPO, successfully progressing to competition-level opponents that baseline MAPPO did not reach. Against heuristic opponents, role-based MAPPO also demonstrated improved win rates. Although final trained role-based MAPPO models devolved to a more defensive policy, they were able to ensure only 2-3$\%$ loss rate, resulting in draws the vast majority of the time. Despite the improvement, both deep MARL approaches were ultimately outperformed by the most advanced heuristic models.

Discussion Role-based MAPPO models represented an improvement over baseline MAPPO, learning coordinated attacks, counter-attacking, and opportunistic positioning. However, there is still room for improvement. The collapse in the KL divergence of the latent role distribution at the end of training indicates that more work needs to be done to regulate the information flowing through the role encoder. Exploration behaviors need to be incentivized in order to find more diverse and traditional roles. Hand-crafted reward functions and reward shaping function are ultimately limiting on the exploration possibilities in this problem space. Additional room for growth exists in capturing higher fidelity snapshots of the time interval since last role update to be passed to the role encoder. This may result in improved latent role priors.

Conclusion In this report, we establish a baseline heuristic curriculum for on-policy training for MCTF. We establish a baseline MAPPO for the MCTF environment, and propose a latent variable role embedding MAPPO. While MAPPO with role conditioning was not able to defeat the most advanced heuristic models, Role-MAPPO was able to improve upon Baseline MAPPO by improving sample efficiency and achieving a higher win rate against heuristic models. We believe that further improvements could lead to significantly better win rates against heuristic models, enhanced sample efficiency, and on-the-fly adaptation to opponents and environmental shift. Future work could focus on ensuring diversity in the role conditioning distribution, improved automated reward handling, improved exploration behavior, and more sophisticated informational bottleneck handling.

Divide, Embed, Conquer: Role-conditioned MAPPO with Continuous Latent Embeddings

Avinash Singh Rajput
Department of Computer Science
Stanford University
rajputav@stanford.edu

Abstract

Teams of autonomous agents face a fundamental challenge in competitive environments: developing specialized roles and the ability to coordinate adaptively, as highlighted by the 2024 AAMAS Maritime Capture the Flag competitions, where deep MARL approaches were outperformed by heuristic policies. We address this by extending Multi-Agent Proximal Policy Optimization (MAPPO) with continuous latent role embeddings that enable agents to dynamically specialize their behavior based on local observations while maintaining shared policy parameters. We measure success by sample efficiency on a standardized curriculum of increasingly difficult heuristic opponents and win rates against heuristic policies. Role MAPPO demonstrated superior sample efficiency, advancing further through the curriculum than baseline MAPPO. Against heuristic opponents, role MAPPO achieved improved win rates, although extended training runs resulted in policies that collapsed to defensive strategies. This research contributes to improved dynamic coordination across teams of autonomous agents, which is essential for important real-world challenges like search and rescue missions and environmental monitoring.

1 Introduction

Multi-agent reinforcement learning (MARL) systems face a fundamental challenge in competitive environments: how to enable teams of autonomous agents to develop specialized roles and coordinate adaptively against other teams. This challenge was highlighted by the 2024 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) Maritime Capture the Flag competition, where deep MARL approaches were outperformed by heuristic and stochastic search methods.

This project uses the Pyquaticus environment and addresses the Maritime Capture the Flag (MCTF) competition by AAMAS.

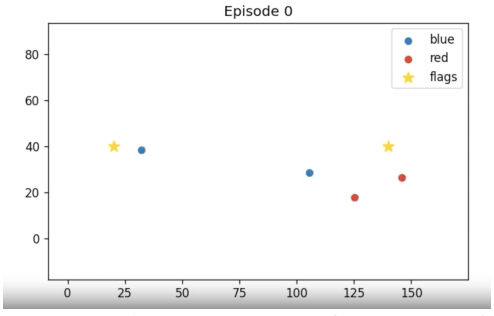

Figure 1: Sample screen capture of an MCTF episode

These coordination challenges reflect real-world problems in autonomous maritime operations, where unmanned vessels need to collaborate for search and rescue missions, environmental monitoring, and security tasks while adapting to dynamic conditions and potential dangers.

In this project, we implement a baseline Multi-Agent Proximal Policy Optimization (MAPPO) for the MCTF environment, experiment with self-play, and establish a baseline MCTF curriculum of heuristic policies of increasing difficulty. We then address the limitations

described above by extending MAPPO with a novel role encoding mechanism. Our approach learns continuous latent role embeddings that let agents dynamically specialize their behavior based on local observations. This results in greater on-policy sample efficiency and improved win rate against heuristic opponents, relative to baseline MAPPO.

2 Related Work

Capture the flag is played between two teams in which each team has a flag in its own half of the field. The objective is to grab the opponent's flag and return it while avoiding being tagged. If a team member is in enemy territory, they can be tagged even without carrying the flag. This game functions as a test-bed for cooperative multi-agent deployments in a competitive zero-sum setting.

One of the most significant recent works in the Capture the Flag game environment and Deep Multi-Agent Reinforcement Learning space is the (Jaderberg et al., 2019) paper on deep multi-agent reinforcement learning results via self-play. This paper utilized self-play with self evolving hyperparameters and internal reward shaping (among other things) to achieve 99th percentile ELO relative to human performance. Unfortunately, the compute budget of this project was much too large to replicate in an academic project, even six years later. Additionally, the learned policies for the environment are task specific. Transferability to new maps or dynamics changes appears to be limited.

In the most recent, AAMAS 2024 Marine Capture the Flag competition (Dasgupta et al., 2024), it was shown that models struggled to adapt in real-time to changing environmental conditions and opponent strategies not encountered in training. In the 2024 competition, while deep MARL techniques were used, stochastic search and naive heuristics outperformed the MARL techniques.

There was a model in the 2024 competition that utilized QMIX (Rashid et al., 2020). QMIX learns per-agent action-value functions and combines them through a mixing network with non-negative forced weights that sees the global state. QMIX then combines these results into a joint Q function. However, in terms of capture performance in the 2024 competition, methods that divided the game into behavioral primitives were the most successful at completing captures. Examples of behavioral primitives include attacking, defending, avoiding, guarding, etc.

This suggests that a model such as ROMA (Wang et al., 2020), could be more successful than QMIX. ROMA adds a role layer on top of the Q value training. It classifies agents based on their local observations into roles in a categorical embedding space and hyper generates agent specific Q weights from the sampled role code, which allows for automated learning of behavior primitives that optimize group success. ROMA uses discrete role embeddings through a Gumbel-softmax sampler.

While value based approaches seem promising, (Yu et al., 2022) show that multi-agent PPO style training outperforms or matches state-of-the-art Q-learning methods, in part because the clipped surrogate objective and on-policy updates mitigate value over-estimation and training divergence that is often seen in off-policy Q learning in sparse reward environments.

One potential weakness of Multi-Agent PPO is that it uses the same policy network for all agents. While weight sharing improves sample efficiency and stability, it may restrict the expressiveness of the policy and coordination space. Moreover, MAPPO's per-agent PPO clip only provides an approximation for a monotonic improvement guarantee because all agents update simultaneously, the joint policy can step outside the trust region, compounding the stationarity problem in MARL. To address the monotonic improvement guarantee, (Kuba et al., 2021) released Heterogeneous Agent PPO (HAPPO), which sequentially updates agents, ensuring monotonic joint-policy improvement without assuming parameter sharing. However, this method is less sample efficient and has scalability issues because the policy updates have to be implemented in sequence instead of in parallel.

MAPPO (and other deep MARL generally) also adapts slowly to out of distribution situations. The Probabilistic Embeddings for Actor-critic Reinforcement Learning (PEARL) paper (Rakelly et al., 2019) proposes a continuous latent task embedding inferred by a probabilistic encoder that conditions the learned policy. This enables a partitioning of the policy space into tasks without requiring additional learning at test time. The continuous conditioning gives PEARL strong expressive power for handling previously unseen tasks.

This study aims to improve upon the state of the art multi-agent policies by introducing a latent role embedding that partitions agent behavior in the multi-agent space rather than task behavior in the single agent space, by conditioning on local observations. We aim to implement this in an on-policy setting, rather than the semi off-policy setting of PEARL and QMIX. In contrast to ROMA's categorical embedding, this study proposes a continuous latent role representation.

In addition to a baseline MAPPO optimized for the MCTF environment, we implement a role encoder that maps each agent's local observation to a Gaussian latent, and a conditional actor that takes the concatenated observation and role vector as input, allowing the same weight set to specialize into distinct roles on the fly. Theoretically, this should increase sample efficiency in the learning process by enabling faster adaptation to harder tasks. This should also lead to improved win rate performance against heuristic models that have previously beaten deep MARL methods in the MCTF space.

3 Methods

3.1 Overview

Although deep MARL entrants in the 2024 MCTF competition relied on value-based approaches such as QMIX, we base our study on Multi-Agent PPO. As mentioned earlier, we chose PPO style training because the clipped surrogate objective and on-policy updates mitigate value over-estimation and training divergence that is often seen in off-policy Q learning in sparse environments (Yu et al., 2022). In order to evaluate these models, we also create a benchmark curriculum of increasingly difficult opponents for the models to train against. The details of this curriculum can be found in the Experimental Setup section.

3.2 Multi-Agent Proximal Policy Optimization

Multi-Agent Proximal Policy Optimization, or MAPPO, implements the centralized training with decentralized execution paradigm to solve MARL. It consists of decentralized actor networks that process local observations during execution, and a centralized critic that processes global state information during training. Although the actor networks at execution time individually process their own local observations, they share the same set of parameters for training, and thus have the same policy space. The centralized critic is able to provide value estimations that capture the coordination and cooperation aspects of the multi-agent problem that would be missing from solely local observations.

The actor network uses a fully connected three-layer architecture with ReLU activation functions with an output layer that soft-maxes logits to action probability distributions. The actor network takes individual agent observations as input. There are 17 discrete action IDs to choose from: 16 correspond to full and half speed vectors, for all $\frac{1}{4}\pi$ radian headings, covering the full circle. The 17th is a no-op, representing zero speed and heading. The actor network outputs logits that are converted to action probabilities for the aforementioned actions using softmax normalization.

We implement the Centralized Critic using PopArt normalization (Hessel et al., 2019). Similar to the actor network, the critic network has 3 fully connected hidden layers and an output layer. PopArt maintains a running statistic of returns and automatically rescales the value function output. This lets the value function handle non-stationary reward distributions, which we experience when entering changing training stages. Further discussion on training stages can be found in Experimental Setup.

The core MAPPO training loop implements Generalized Advantage Estimation (Schulman et al. (2015)) after collecting rollouts and before applying PPO updates. GAE reduces variance in policy gradient estimates by computing a bias-variance tradeoff between bootstrapped and Monte Carlo returns. Our MAPPO loop computes the advantages separately for each agent's trajectory ensuring that the advantage estimator aligns with each agent's respective reward stream and transition sequence. The GAE advantage is given by:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l} \tag{1}$$

where the temporal difference residual is:

$$\delta_t = r_t + \gamma V(s_{t+1})(1 - d_t) - V(s_t) \tag{2}$$

$d_t \in \{0, 1\}$ is the termination flag and $V(s_t)$ is the current value estimate. The factor $(1 - d_t)$ ensures that bootstrapping ceases at episode boundaries. The smoothing parameter $\lambda \in [0, 1]$ controls the bias–variance tradeoff: $\lambda = 0$ yields low-variance but biased estimates (akin to bootstrap learning), while $\lambda = 1$ recovers high-variance, unbiased Monte Carlo returns. In our setup, we used $\lambda = 0.95$

The computed advantage $\hat{A}_t$ is used in the actor loss within the standard PPO clipped surrogate objective:

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \mathrm{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\big)\Big], \tag{3}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ is the policy probability ratio and ϵ is the clip threshold. In our experiments, ϵ was set in the range (0.175, 0.2)

The value loss, or critic loss is given by the mean squared error between the critic output, normalized by PopArt (v_{norm}) and normalized return $R_t^{\mathrm{norm}} = \frac{R_t - \mu}{\sigma}$, where μ and σ are the mean and standard deviation of the returns, respectively. This gives us

$$\mathcal{L}_{\mathrm{critic}} = \|v_{\mathrm{norm}} - R_t^{\mathrm{norm}}\|^2 \tag{4}$$

We also implement an entropy loss term in the loss function with the measured entropy of the policy multiplied by the entropy coefficient (0.01), to encourage exploration.

The total loss function for our baseline MAPPO setup is given by:

$$\textbf{Total loss} = \text{Actor loss} + \text{Value loss} - \text{Entropy loss} \tag{5}$$

3.3 MAPPO with Role Encoder

MAPPO with Role Encoder extends MAPPO with a role encoder that maps each agent's local observation o_t to a latent Gaussian $\mathcal{N}(\mu(o_t), \sigma^2(o_t))$. A role sample $r_t = \mu + \exp(\log \sigma) \odot \epsilon,\ \epsilon \sim \mathcal{N}(0, I)$ is concatenated to the local observations, o_t, and fed to the shared actor, allowing the shared actor policy to specialize over disjoint behavioral sub-spaces. The Role Encoder network has a fully connected two-layer network with ReLU activation functions that maps agents' local observations to an abstract intermediary role space that is the size of the hidden layer dimensions (128). The encoder has two output layers, mu and logstd, which correspond to the average and log standard deviation of the Gaussian distribution that describes the latent role space. These layers are the size of role dimension, which we set to be 16. These output layers map the observation mapping network to the latent role distribution. Following the example of PEARL, we use log standard deviation because it ensures positivity and has well behaved gradients to enable learning to flow through the network parameters.

One important technique we use from PEARL's example is the reparameterization trick, which makes sampling from the distribution (which we do when selecting a role for the agent), differentiable. A random function is not differentiable, so rather than creating a normal distribution with the learned standard deviation and mean of the role space and sampling from there, we generate standard normal, multiply the sample by the learned standard deviation, and add the learned mean. By parameterizing the mean and variance we keep gradients flowing and allow gradient-based learning of the appropriate distribution. In essence, rather than sampling $r \sim \mathcal{N}(\mu, \sigma^2)$, which is stochastic, we sample $\epsilon \sim \mathcal{N}(0, I)$ and compute $r = \mu + \sigma \odot \epsilon$.

The sampled role vector r is concatenated with the agent's observation and fed to the policy network, which theoretically enables the same shared agent network parameters to have different specialized behaviors based on the role assignment embodied by the role vector.

To complete the informational flow through to the end of the system, the learned μ and σ (via log standard deviation) are incorporated in the loss function with a KL divergence penalty as follows:

```
def kl_gaussian(mu, log_std):
    return 0.5 * torch.sum(mu.pow(2) + torch.exp(log_std * 2) - 2 * log_std - 1, dim=-1)
```

This method is the analytical form to

$$\mathrm{KL}\left(\mathcal{N}(\mu, \sigma^2) \,\|\, \mathcal{N}(0, I)\right) \tag{6}$$

where $\mathcal{N}(\mu, \sigma^2)$ is the learned role distribution. Implementing this loss ensures that our learned role distribution does not drift too far from the standard normal prior with mean 0 and variance 1. This is important because we model our role distribution as a normal distribution - if we drift too far from this family of distributions, the model would no longer remain properly constrained within the parameterized family. This KL penalty also forces both agents to use the same embedding space by trying to align to the same normal prior. Also, as discussed in class, the KL divergence penalty also functions as an informational bottleneck, forcing the role encoder to compress local observations into only the most strategic relevant patterns for role and coordination, instead of using it to memorize situations and responses.

We also implement a beta schedule to gradually ramp up the KL divergence penalty for roles. At the beginning of the curriculum, the intention is to teach the agents base behaviors like tagging and grabbing the flag. Role specialization should only occur after these base skills have been acquired.

In order to prevent the computed agent roles from collapsing into the same policy, we also implement a role diversity loss, inspired by ROMA Wang et al. (2020). The role diversity loss calculates the normalized cosine similarity between agent roles and penalizes accordingly. Cosine similarity increases when vectors are pointing in more similar directions.

```
def role_diversity_loss(self, r_new, freeze_roles):
        r_unit = F.normalize(r_new, p=2, dim=-1)
        sim = r_unit @ r_unit.T
        div_loss = sim.sum() - sim.diag().sum()
        div_loss = (DIVERSITY_LOSS_WEIGHT / (r_new.size(0) ** 2 - r_new.size(0))) * div_loss
        return div_loss
```

This yields the total loss function as

$$\textbf{Total loss} = \text{PPO Loss} + \text{KL Divergence Loss} + \text{Role diversity Loss} \tag{7}$$

Finally, instead of recalculating the role embeddings at every time step, we lock each agent's roles for 48 time steps, equivalent to roughly 4.8 seconds for our environment settings ($\tau = 0.1$). This stabilizes learning and reflects the human intuition that roles persist over time. We scale the encoder learning rate by 0.25 the policy learning rate to ensure more stable learning. When calculating the role embeddings we only pass the agent's most recent local observation, enabling the agent to receive a role based on its current situation. Although some experiments omitted the role vector from the critic input, effective learning required concatenating the role vector with the global observations (as in baseline MAPPO) and passing this to the critic network.

4 Experimental Setup

The experimental setup is built on the Pyquaticus environment. Pyquaticus is a PettingZoo environment for maritime capture the flag with uncrewed surface vehicles. It was built by researchers at Massachusetts Institute of Technology with support of the United States Under Secretary of Defense. While the AAMAS competition is a 3v3 competition, to ease computational demands, this study trained agents in a 2v2 configuration.

Capture the flag is played between two teams in which each team has a flag in its own half of the field. The objective is to grab the opponent's flag and return it while avoiding being tagged. If a team member is in enemy territory, they can be tagged even without carrying the flag. Details on reward parameters from game state used for the reward function can be found in Section B.1.

The initial experimental setup was to have all agents play against each other, inspired by Jaderberg et al. (2019).

The second, and primary, experimental setup, used a staged curriculum approach to training. There were eight stages to this curriculum. Each stage had its own set of heuristic strategies defined. The earliest stages had idle or random opponents. As stages progressed, they had agents configured with increasingly difficult heuristic agents, as released in the Pyquaticus environment. In order for training to progress to the next curriculum stage, the team had to reach a predefined reward threshold, n times in the previous 50 episodes. The earliest stages were only intended to teach basic skills against idle and random opponents, and thus had lower reward thresholds and shorter window requirements. The later stages, against opponents using more intelligent heuristic strategies required the trained agents to at least achieve a flag capture and a draw for 60% of a running window of 50 episodes. This curriculum was standardized across experiments, in order to ensure that training comparison across models could be standardized. Curriculum details can be found in Section B.2.

The trained models were then tested for 100 full length matches against the various heuristic policies and evaluated for win rate, draw rate, and qualitative performance.

5 Results and Discussion

The first experiment, conducted with self-play, led to disappointing results. While DeepMind's approach achieved success with self-play techniques, we did not find it was effective for our application, in part because exploration of the action space was limited by the algorithm's search methods. Additionally, Jaderberg et al. (2019) used an internal reward generation system to map from sparse win-loss rewards. Our handcrafted, artisanal reward functions were not able to escape the Nash equilibrium of mutual deterrence. Furthermore, doing back-of-the envelope calculations, we did not believe we had the computational capacity to match the self-play techniques in the referenced paper. As you can see in the image below, both teams of agents converged to a static equilibrium, refusing to explore the attack space required to achieve wins.

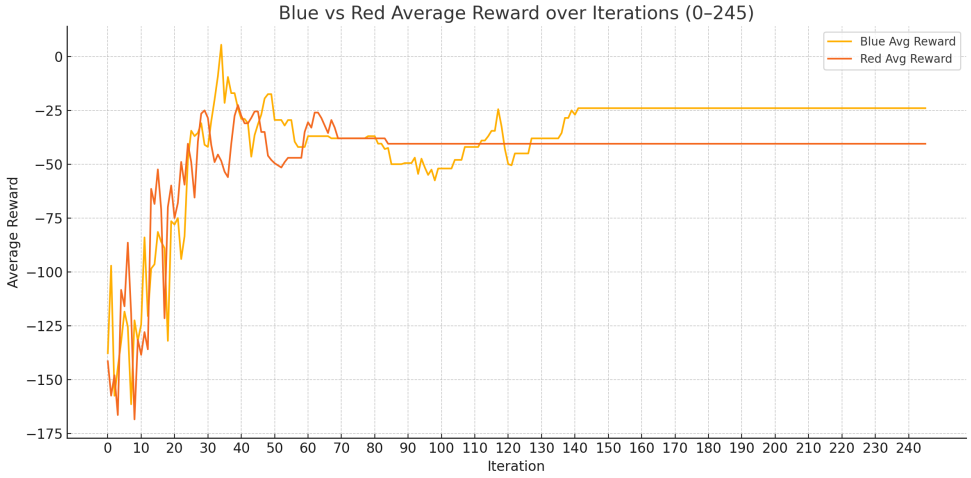

Figure 2: Rewards vs iterations for teams of agents with self-play

When self-play did not yield promising results, curriculum stage learning was implemented. It should be noted that the reward function was changed after self-play. Details about reward parameters can be found in Section B. Baseline MAPPO was able to successfully clear many of the curriculum stages with appropriately tuned hyperparameters. As can be seen in Figure 3, Baseline MAPPO managed to clear early stages relatively quickly, reaching "base-easy" in less than 100 iterations.

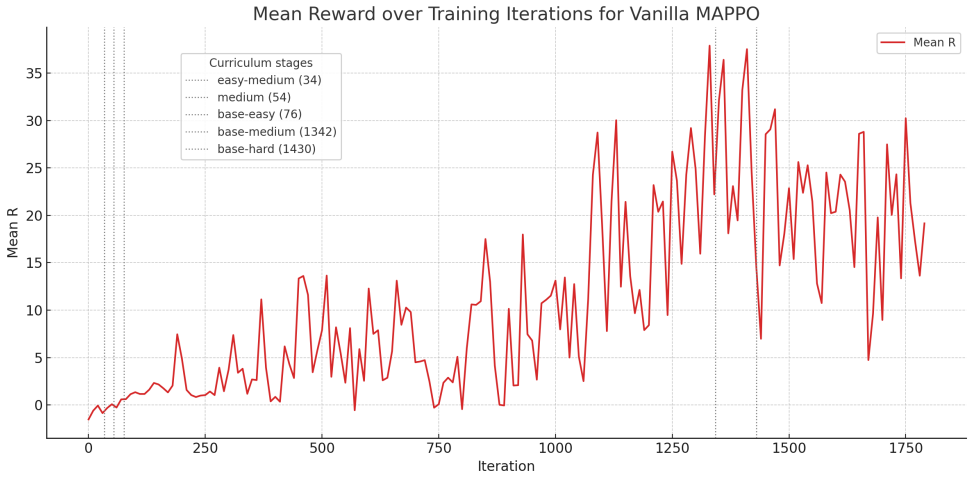

Figure 3: Rewards vs iterations for Baseline MAPPO agent training

It did however struggle to learn effective strategies against base-easy heuristics, only doing so after more than 1200 iterations. However, once it found winning strategies against base-easy, it managed to utilize this learned lesson on base-medium, quickly progressing to base-hard in less than 100 iterations. Baseline MAPPO remained in the base-hard task through the end of training (1800 iterations). There was significant volatility in the rewards per iteration in the learning task. The mean reward of around thirty at the time of base-easy graduation indicates that the agents were winning or drawing their matches.

While Baseline MAPPO managed to clear earlier curriculum stages slightly more quickly compared to Role MAPPO, Role MAPPO managed to advance to the more difficult curriculum stages within iteration limits - something that Baseline MAPPO was not able to do, as can be seen in Table 1.

Curriculum Stage	**Baseline MAPPO**	**Role MAPPO**
easy–medium	34	50
medium	54	70
base–easy	76	110
base–medium	1342	1270
base–hard	1430	1293
competition–easy	–	1314

Table 1: Iterations at which each curriculum stage begins for the two training setups.

This suggests that Role MAPPO was more expressive and had greater learning capacity compared to Baseline MAPPO, and was better able to generalize learned skills and roles from earlier opponents to newer ones. Looking more closely at the data from the training run for Role MAPPO in Figure 3, we can see that the mean reward between 20 and 30 around base-hard stage graduation indicates a majority of matches are being won or drawn for the trained team.

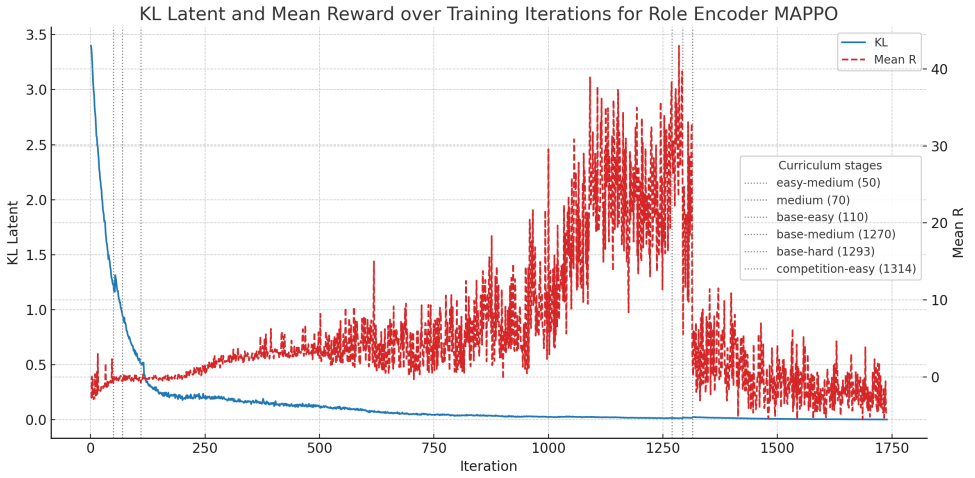

Figure 4: Rewards vs iterations for agents with Role MAPPO

However, Role MAPPO collapsed when facing the competition-easy heuristic bots from iteration 1300 to iteration 1800 (training end). Role MAPPO appears to have turtled into an overly conservative defensive strategy, consistently securing draws in lieu of wins.

The KL divergence of the latent role distribution curve in the figure represents the average KL divergence between the learned role encoding distribution and the standard normal prior ($\mathcal{N}(0, I)$). This KL divergence, also used in the loss term, serves as a proxy for how expressive the learned role space is - larger values indicate greater deviation from the prior, and more informative, differentiated role encodings.

As shown, KL divergence of the latent role distribution falls to nearly zero by the end of training run indicating that the learned role distribution has nearly collapsed to the prior. We interpret this as evidence that the role space has lost most of its expressive power, which likely contributed to the agent's collapsed policy by training end.

The ultimate test of these models is their performance in matches. Because of the observation that Role MAPPO performance significantly degraded from exposure to competition-easy models, we also evaluated a saved Role MAPPO model at iteration 1400, 86 iterations after initial exposure to competition-easy. The results are shown in Figure 5

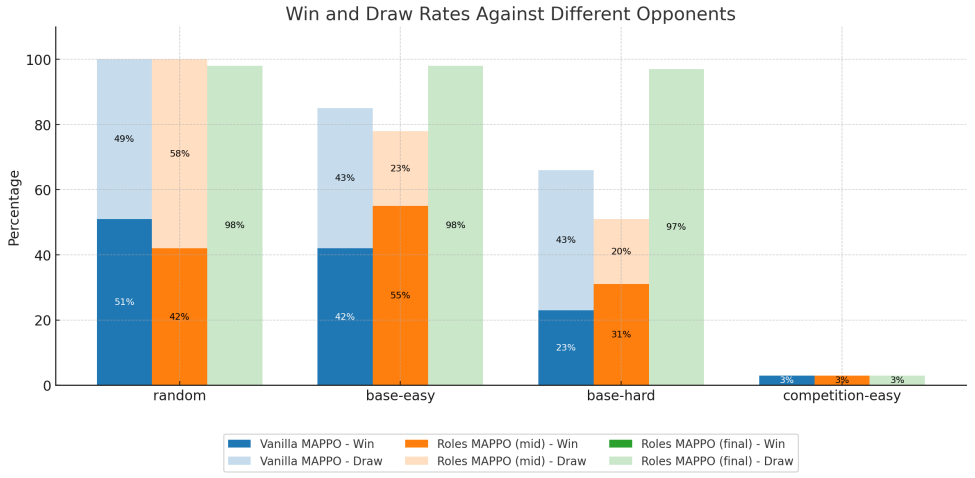

Figure 5: Win and draw rates for various models against heuristic opponents of varying difficulties

The Final Role MAPPO model learned to aggressively secure draws, achieving 97% and 98% draw rates for random, base-easy and base-hard opponents. Baseline MAPPO had slightly lower loss rates for base-easy and base-hard opponents, compared to Mid Role MAPPO, but also had lower win rates compared to Mid Role MAPPO. All of the learned models were completely outclassed by the competition easy model, falling to a 97% loss rate. The gap between the learned models and the heuristic models can be further illuminated by the knowledge that there exists an even more advanced competition-medium heuristic model that these Deep MARL models never faced.

The learned strategies of the models were very interesting to observe. The Baseline MAPPO agents tended to maintain more distance from each other. When attacking, the agents would maintain a perimeter around the opponent flag. When an agent saw an opening it would seek to grab the flag. Sometimes, an agent would post just on the opponent side of the border, leveraging the exploration bonus for entering opponent territory, which was added to encourage flag-capture attempts. The agents also seemed to prefer trying to capture the enemy flag together.

The Final Role MAPPO policy tended to employ a counter-attacking strategy. It tended to stay close to its partner, which was a behavior carried over from mid training observations. They learned to stay close to their partner to observe a tag, and then if they were closer to the flag than their partner after a capture, would head to enemy territory to utilize the temporary numeric advantage (tagged agents are inactive until they return to base).

The Mid Role MAPPO policy was very opportunistic, moving slowly and broadly out of its own zone, watching for opponent agents entering its zone. It tended to stay close to its partner when advancing in opponent territory, scouting slowly, sometimes slowly maintaining distance from defenders and then accelerating at max speed to the flag when it saw an opportunity. Usually when attacking together, only one agent would lead the way to the flag and the other agent would trail closely behind, taking an alternate path to the flag once the lead agent accelerated on a path.

We find it interesting that all of the trained agents tended to attempt to capture the flag synchronously. We suspect this may have had something to do with captures against random opponents requiring tandem attacks because single agent attacks were liable to get tagged by unpredictable movement. When agents attacked together against a random opponent, if one agent was tagged, the other might have some free space to go after the flag.

In general, upon curriculum shift, if a policy distribution was consistently unsuccessful, the agents would shift to a more defensive policy, and seek a more offensive policy only after securing consistent draws. This is perhaps what a human would do.

The most significant challenge with Role MAPPO is management of the information contained in the role embedding space. We had a diversity collapse in role MAPPO as embodied by near 0 KL divergence of the latent role distribution. In part, this is because exploration in the policy space remained challenging, even with reward shaping. While a small diversity loss was implemented, with tuning, perhaps a better diversity loss hyperparameter could be found. Perhaps a contrastive loss could be implemented to address this issue. It is also possible that the diversity loss was interfering with the KL loss term. The informational bottleneck must also be carefully managed at the beginning of training. Too tight an informational bottleneck can restrict learning of baseline skills. We also believe that experimentation against fixed roles or discrete roles would be a worthwhile comparison.

The specific time step for role freeze was chosen due to human intuition about the problem - with more time and computing resources this would likely be a tuned hyperparameter. Additionally, role prediction used only an agent's most recent observation. Using PEARL's product of gaussians trick, we could have expanded to pass more than the agent's most recent transition when calculating an agent's role embedding.

Artisanal reward functions are a labor of love, but they are indeed a labor. Evolved reward functions or simply relying on the sparse rewards of the environment and implementing algorithmic improvements would be greatly preferred. Additionally, we believe the learned dynamics of Role MAPPO in particular was a product of the reward function pushing the agents towards defensive/draw seeking behavior in dangerous situations. With regards to the current setup, in order to encourage more win seeking behavior, the curriculum reward thresholds should likely be increased.

Finally, it is possible that models such as RNNs, LSTMs, or Mamba models, could hold sequence information that could be useful for the role encoder.

6 Conclusion

In this report, we establish both a baseline MAPPO and a baseline heuristic curriculum for the MCTF environment and also propose a latent variable role embedding MAPPO. While MAPPO with role conditioning was not able to defeat the most advanced heuristic models, Role-MAPPO was able to improve upon Baseline MAPPO by improving sample efficiency and achieving a higher win rate against heuristic models. We believe that further improvements could lead to greatly improved win rates against heuristic models, improved sample efficiency, and on-the-fly adaptation to opponents and environmental shifts. Future work could focus on ensuring diversity in the role conditioning distribution, improved automated reward handling, improved exploration behavior, and more sophisticated informational bottleneck handling.

7 Team Contributions

- **Avinash Rajput:** I completed all portions of this project. I collaborated with ChatGPT and Claude to generate boilerplate code, to set up data loading tools, to parallelize workflows to support multiple workers, and to help set up test harnesses and episode rendering videos.

Changes from Proposal In the proposal, we wanted to focus on handling environmental shifts at test time that were out of distribution from the training set. While we were able to create a test environment that ran matches in different dynamics, it was verified that achieving a baseline of good performance against simple heuristics was actually quite difficult, as evidenced by results of state-of-the-art work at AAMAS 2024.

References

Prithviraj Dasgupta, John Kliem, Zachary Serlin, Michael Novitzky, Jordan Beason, Tyler Errico, Michael Benjamin, Tyler Paine, Peter Crowley, Simon Lucas, James Chao, Wiktor Piotrowski, Gautham Dixit, Matthew Leprell, Jeffery Richley, Christopher Tucker, Abdullah Alamar, Katsuki Ohto, and John Meo. 2024. The First International Maritime Capture the Flag Competition: Lessons Learned and Future Directions. In *Proceedings of the 2024 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. `https://openreview.net/pdf?id=4xlDyyY19L`

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado Van Hasselt. 2019. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3796–3803.

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2021. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251* (2021).

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. 2019. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*. PMLR, 5331–5340.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039* (2020).

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems* 35 (2022), 24611–24624.

A Additional Experiments

We conducted additional experiments with capacity-based informational bottlenecks, sampling additional transitions for latent role sampling, encoder freezing, role holds, and encoder learning rate scaling but there wasn't enough time to refine results from these experiments. We also experimented with RNN-based critics, but these did not yield successful results, likely due to implementation errors.

B Implementation Details

B.1 Reward Settings

B.2 Curriculum settings

Reward Component	Value	Comment
FLAG_CAPTURE	+10.00	
FLAG_PICKUP	+0.38	
SUCCESSFUL_TAG	+0.08	
GOT_TAGGED_PEN	-0.05	
OOB_PENALTY	-0.05	out of bounds penalty
COLLISION_PENALTY	-0.05	
OPP_FLAG_CAPTURE_PENALTY	-0.45	penalty for opponent captures
OPP_FLAG_PICKUP_PENALTY	-0.02	penalty for opponent flag pickup
OPP_SIDE_BONUS	+0.01	bonus for being opponent territory
NEAR_FLAG_BONUS	+0.01	
CARRY_HOME_BONUS	+0.05	bonus for every step the flag is carried home
REWARD_CLIP	60.00	win game plus two captures
WIN_GAME_REWARD	40.00	blue caps $>$ red
LOSE_GAME_PENALTY	-5.00	red caps $>$ blue
DRAW_GAME_PENALTY	-0.50	blue == red

Table 2: Reward and Penalty Parameters for MAPPO Training

Table 3: Curriculum stages used during MAPPO training.

Stage	Threshold	Active Red	Reward Mult.	Min Return	Policy Mode
Easy	10	0	1	1.0	`2 inactive`
Easy-Medium	10	1	1	1.1	`1 inactive, 1 random`
Medium	10	2	1	1.1	`2 random`
Base-Easy	30	2	1	5.0	`heuristic easy`
Base-Medium	30	2	1	5.0	`heuristic medium`
Base-Hard	30	2	1	5.0	`heuristic hard`
Competition-Easy	30	2	1	5.0	`heuristic comp_easy`
Competition-Medium	40	2	1	–	`heuristic comp_med`