Reproducibility Study of "Efficient Episodic Memory Utilization of Cooperative Multi-Agent Reinforcement Learning"

Anonymous authors
Paper under double-blind review

Abstract

This paper reports on the reproducibility study on the paper "Efficient episodic memory utilization of cooperative multi-agent reinforcement learning" by Na et al. (2024). The original study proposed a method to enhance MARL performance by leveraging episodic memory to accelerate learning and prevent local optima convergence. EMU introduced a trainable encoder/decoder structure for memory retrieval and an episodic incentive reward mechanism to promote desirable transitions. The original work evaluated the method in StarCraft II and Google Research Football, demonstrating improvements over state-of-the-art approaches. This study further examines the effectiveness of EMU by assessing its reported performance improvements, the impact of its state embedding approach on exploration efficiency, and the robustness of its incentive mechanism in preventing suboptimal convergence. The analysis focuses on the SMAC benchmark, particularly in complex scenarios where EMU showed the most promise, while also exploring its scalability in high-performance computing environments to determine its computational feasibility. Our findings confirm the advantages of EMU but underscore the sensitivity of its performance to embedding quality and hyperparameter selection. Our extended implementation and results are available on Github.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) has become increasingly vital in real-world applications, from traffic management Wiering et al. (2000) to manufacturing systems Dittrich & Fohlmeister (2020), resource allocation Dandanov et al. (2017). Despite notable successes, MARL faces persistent challenges in achieving effective agent coordination, primarily due to partial observability and complex inter-agent interactions during training. The framework of centralized training and decentralized execution (CTDE) Oliehoek et al. (2008) emerged as a promising solution, enabling agents to leverage global information during training while maintaining decentralized execution capabilities. Value factorization approaches within CTDE have achieved state-of-the-art performance on challenging tasks like the StarCraft II Multi-agent Challenge (SMAC) Samvelyan et al. (2019).

However, MARL systems still struggle with prolonged convergence times and susceptibility to local optima, particularly in complex task environments. While recent work has explored committed exploration mechanisms to escape these local optima, a fundamental tension exists between exploration and exploitation. The introduction of Efficient Episodic Memory Utilization (EMU) Na et al. (2024) addresses this challenge by refining how agents leverage past experiences. Traditional episodic memory approaches rely on random projections for state embeddings, which often fail to capture semantic similarities between states. EMU selectively encourages desirable transitions with semantic memory embeddings that identify and prioritize high-value transitions based on their contextual relevance to goal achievement, enabling more efficient exploration while avoiding the pitfalls of repeated local optima. Consequently, given the enhanced semantic state representations and efficiency gains achieved by EMU, we deem it both pertinent and valuable to replicate these findings to further substantiate its potential in mitigating exploration–exploitation trade-offs within complex MARL environments.

Our work makes the following contributions:

- A systematic replication of the EMU framework Na et al. (2024), focusing on reproducible experiments while documenting implementation challenges and resource requirements. This validation effort strengthens the theoretical foundations of episodic control in MARL.
- A rigorous parametric analysis of the state embedding threshold (δ) unveils its pivotal role in dictating convergence and stability across diverse memory embedding techniques (Random Projection, EmbNet, and dCAE).
- Adaptation of the EMU framework for distributed computing environments, enabling scalable training across high-performance computing clusters while maintaining coordination efficiency. This extension demonstrates EMU's potential for large-scale, computation-intensive MARL applications.

2 Scope of Reproducibility

EMU was introduced to address the challenges of efficient exploration and exploitation in cooperative multi-agent reinforcement learning (MARL) through improved episodic memory utilization. The framework builds upon episodic control techniques, enhancing them with semantic embeddings and targeted incentive mechanisms. Na et al. (2024) presented three main claims in their paper, which we aim to investigate:

1. **Claim 1: EMU achieves superior performance compared to state-of-the-art MARL frameworks.**
   The authors demonstrate that EMU, when integrated with existing frameworks like QPLEX and CDS Chenghao et al. (2021), significantly improves performance and accelerates convergence to optimal policies, particularly in challenging scenarios like super hard SMAC maps. Our reproduction focuses on validating these improvements and understanding the computational requirements needed to achieve them.
2. **Claim 2: The proposed state embedding approach enhances exploration efficiency.**
   The authors claim that their dCAE (deep Convolutional AutoEncoder) structure effectively handles a wider range of similarity thresholds (δ) compared to traditional embedding networks. They demonstrate that selecting appropriate δ values significantly impacts learning performance in complex MARL tasks. Our work examines the robustness of these improvements across different threshold values and validates the reported optimal ranges.
3. **Claim 3: The episodic incentive mechanism prevents convergence to local optima.**
   The original paper shows that their episodic incentive mechanism reduces performance variation across different random seeds, particularly when compared to conventional episodic control implementations. They argue this demonstrates the mechanism's ability to prevent detrimental local convergence. Our reproduction investigates this stability claim and examines the mechanism's effectiveness across various environmental conditions.

Our reproduction efforts focus particularly on the SMAC benchmark environment, as it represents the primary testbed used in the original paper and provides a standardized way to evaluate MARL performance. We pay special attention to the super hard scenarios (e.g., 6h_vs_8z) where the benefits of EMU were most pronounced. Additionally, we examine the scalability of these results in high-performance computing environments, as this aspect was not extensively covered in the original work.

3 Methodology

In this section, we detail our approach for reproducing and analyzing the Efficient Episodic Memory Utilization (EMU) framework for cooperative multi-agent reinforcement learning (MARL) using the publicly available code from the repository of EMU.

3.1 EMU

Efficient Episodic Memory Utilization (EMU) enhances cooperative multi-agent reinforcement learning (MARL) by integrating structured memory representations with an adaptive incentive mechanism. Standard

MARL approaches often struggle with inefficient exploration and suboptimal convergence due to complex agent interactions. EMU addresses these limitations by employing a trainable memory embedding that encodes and organizes past experiences, allowing agents to retrieve relevant information and refine their decision-making. Simultaneously, it introduces an episodic incentive mechanism that selectively amplifies the influence of beneficial transitions, ensuring a balance between exploitation and exploration.

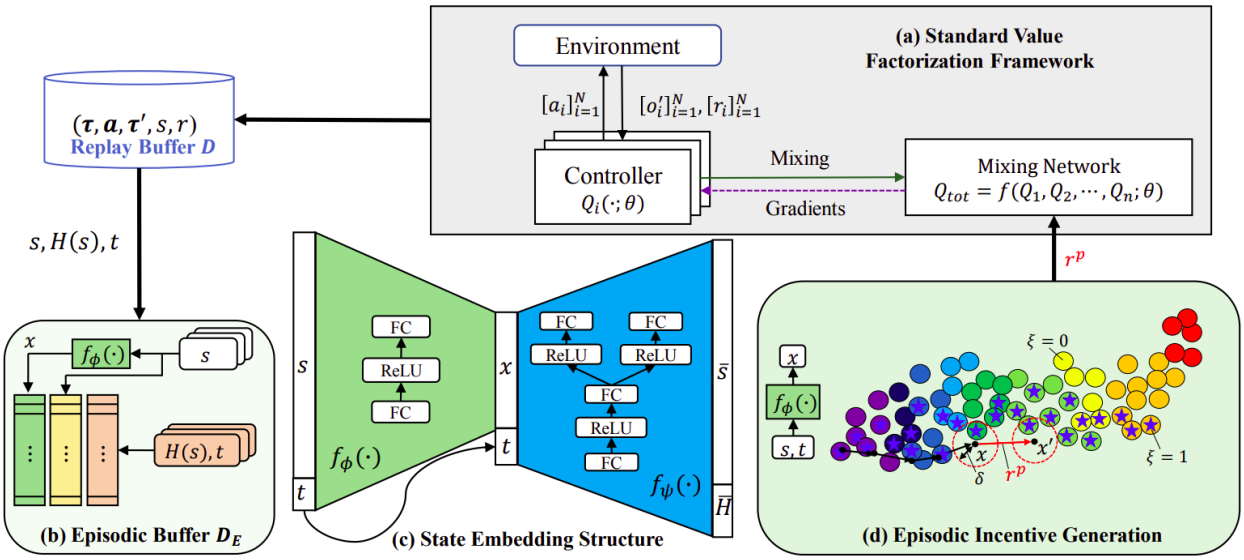


Figure 1: Overview of EMU framework. *Note*: Image from Na et al. (2024)

As illustrated in Figure 1, EMU refines episodic memory utilization in MARL by structuring the way agents store, retrieve, and leverage past experiences. Instead of relying on random projections, EMU employs a deep Convolutional Autoencoder (dCAE) to encode state representations, ensuring that semantically similar states are clustered in the latent space. This structured memory representation enables agents to retrieve meaningful past experiences, improving policy learning and decision-making efficiency Na et al. (2024). Unlike conventional approaches that apply fixed-distance thresholds for memory recall, EMU dynamically adapts its retrieval process, reducing errors in state generalization and allowing more effective long-term credit assignment.

Beyond memory encoding, EMU introduces an episodic incentive mechanism that promotes efficient exploration by modifying the learning process. Rather than relying solely on environmental rewards, EMU computes additional incentives based on the desirability of visited states. A state's desirability is determined by its historical contribution to successful trajectories, with high-value transitions receiving greater reinforcement. Formally, the episodic incentive r_p is defined as:

$$r_p = \gamma \hat{\eta}(s') \tag{1}$$

where $\hat{\eta}(s')$ quantifies the discrepancy between the actual state value $V^*(s')$ and the predicted value of the function Q. In practice, this term is computed as follows:

$$\hat{\eta}(s') = \frac{N_\xi(s')}{N_{\text{call}}(s')} \cdot \Big(H(f_\phi(s')) - \max_{a'} Q_{\theta^-}(s', a') \Big) \tag{2}$$

Here, $N_\xi(s')$ and $N_{\text{call}}(s')$ represent the state's visitation count and memory retrieval frequency, respectively, while $H(f_\phi(s'))$ denotes the entropy of the state's latent representation. This mechanism prevents premature convergence to suboptimal policies by encouraging agents to revisit promising but underexplored states. By integrating structured memory representations with adaptive incentives, EMU significantly enhances MARL performance, improving convergence speed, exploration efficiency, and overall agent coordination.

3.2 Environments

Feature	StarCraft Multi-Agent Challenge	Google Research Football
Description	Cooperative environment where agents control units in StarCraft II to defeat enemy units	Multi-agent soccer simulation where agents control players to score goals
State space	Coordinates of all agents and comprehensive features of both allied and enemy units	Player coordinates, ball possession information, and player directional data
Action space	Movement and attack actions; action space increases with the number of enemy units	Fixed set of actions including movement, ball control (kick, intercept, dribble), and running
Episode duration	Variable duration depending on map configuration (ranges from 70 to 400 time steps)	Consistent duration of 150 time steps across all tasks
Reward structure	Agents receive rewards for damaging enemies and bonuses for winning. The maximum cumulative reward per episode is scaled to 20	Sparse reward system: +1 for successful goal, -1 for conceded goal

Table 1: Comparison of Multi-Agent Reinforcement Learning Environments

3.3 RL Algorithms

3.3.1 CDS

Coordinate Descent Scheduling (CDS) is an optimization algorithm that iteratively updates individual coordinates of the decision variable while keeping others fixed, progressively improving the objective function. It is particularly useful in high-dimensional problems due to its computational efficiency and ability to exploit sparsity. According to Wright (2015), the CDS algorithm follows a structured update process where, at each iteration, a single coordinate x_i of the decision variable $\mathbf{x} \in \mathbb{R}^n$ is selected and updated based on the partial derivative of the objective function $f(\mathbf{x})$. The update rule can be expressed as:

$$x_i^{(k+1)} = x_i^{(k)} - \alpha \frac{\partial f(\mathbf{x})}{\partial x_i} \tag{3}$$

where α is the step size. The choice of coordinate selection can follow different strategies, such as cyclic selection, random selection, or adaptive methods that prioritize coordinates with the highest gradient magnitudes. By updating one coordinate at a time, CDS reduces the computational burden compared to full-gradient methods, making it well-suited for large-scale optimization problems. Its convergence depends on the objective function and selection rule, with strong theoretical guarantees in convex settings.

3.3.2 EMC

EMC (Episodic Multi-agent Reinforcement Learning with Curiosity-driven Exploration) is a reinforcement learning framework designed to enhance exploration in multi-agent systems by introducing curiosity-driven mechanisms. The algorithm focuses on promoting exploration by using intrinsic rewards based on the agents' curiosity about the environment, which encourages them to explore unknown or less visited states.

According to Zheng et al. (2021), EMC incorporates episodic memory to retain past experiences and uses curiosity-driven exploration to address the exploration-exploitation trade-off in multi-agent settings. The episodic memory enables agents to revisit useful past experiences, improving both sample efficiency and the overall learning process. The curiosity mechanism is implemented by an intrinsic reward function that depends on the agents' uncertainty about their environment, formalized as:

$$r_{\text{curiosity}}(s, a) = \alpha \cdot \log\left(\frac{1}{P(s, a)}\right), \tag{4}$$

where $P(s, a)$ represents the predicted probability of an agent's state-action pair, and α is a scaling factor. This intrinsic reward encourages the agent to explore actions that lead to states with high uncertainty, complementing the extrinsic rewards from the environment. By integrating episodic memory with curiosity-driven exploration, EMC improves exploration efficiency and accelerates the convergence of policies in cooperative multi-agent environments. Its approach is particularly beneficial in tasks where agents need to balance exploration and exploitation over extended episodes.

3.4 Experimental setup

This section describes the experimental setup used to evaluate the performance of EMU. The experiments were conducted in two challenging environments: StarCraft Multi-Agent Challenge (SMAC) and Google Research Football (GRF) Kurach et al. (2020), which we will briefly detail. The setup includes information about the hyperparameters, evaluation metrics, and infrastructure.

3.4.1 Hyperparameters

Parameter	Specification
δ (Delta, controls the influence of episodic memory)	• Easy/Hard SMAC Maps: 1.3×10^{-5} • Super Hard SMAC Maps: 1.3×10^{-3} • GRF Tasks: 1.3×10^{-3}
f_ϕ (Embedding Function)	• Random Projection • EmbNet (neural network-based) • dCAE (deep convolutional autoencoder-based)
Training Configuration	• ncircle: 1 (32 episodes/batch) • Batch Size: 32 episodes • Random Seeds: 5 (32 episodes/seed)

Table 2: Hyperparameters and Training Configuration Details

3.4.2 Evaluation Metrics

- **Overall Win-Rate ($\bar{\mu}_w$)**: A new performance index that combines training efficiency (speed) and quality (win-rate) across different random seeds.
- **Learning Curves**: Win-rate plot over training time to visualize model's convergence and stability.
- **Convergence time**: Number of episodes required to reach a stable win-rate.

3.4.3 Infrastructure

- Experiments were conducted on NVIDIA GeForce RTX 3090 GPUs.
- The most time-consuming training experiment was completed in under 18 hours using EMU (CDS).
- When training with ncircle = 2, the time increased by more than 1.5 times additionally.
- Training of the encoder / decoder structure and update of D_E (Dynamic Embedding) with f_ϕ (embedding function) took less than 2 seconds in the corridor task.
- The embedding network (f_ϕ) and D_E are updated periodically with t_{emb} (time interval for updating). While this adds some time to the training process, that additional time is negligible compared to the total MARL training time.

3.5 Additional Experiments

3.5.1 Parametric analysis of the state embedding threshold (δ)

In this experiment, we aim to perform a comprehensive parametric analysis to assess how the state embedding threshold (δ) influences the performance of episodic memory representations in cooperative MARL. By systematically comparing three memory embedding strategies (Random Projection, EmbNet, and dCAE) over a range of δ values, we aim to elucidate their impact on learning dynamics, particularly in terms of convergence behavior and the trade-off between exploration and exploitation. The insights derived from this analysis are critical to refining the theoretical foundations of episodic memory utilization and to guiding the design of more robust and efficient multi-agent systems capable of addressing complex learning challenges.

3.5.2 Overcoming Local Optima Through Exploration Incentives

EMU's semantic memory learning reinforces high-value transitions using a loss function that aligns predicted and actual state values. However, this can lead to premature convergence, as frequently retrieved states dominate exploration, trapping agents in suboptimal policies. To counter this, we introduce an exploration incentive that encourages broader exploration before agents rely on memory-driven decision-making, preventing overcommitment to familiar states and mitigating early convergence to local optima.

Mathematically, the challenge is expressed as:

$$E\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid \pi_{\text{local}}\right] < E\left[\sum_{t=0}^{\infty}\gamma^t r_t \mid \pi^*\right], \tag{5}$$

where π_{local} is a suboptimal policy, π^* is the optimal policy, γ is the discount factor, and r_t is the reward at time t. To regulate exploration, we introduce:

$$I(s') = \begin{cases} 0, & \text{if } N_{\text{call}} \leq N_{\text{threshold}}, \\ 1 - e^{-\beta(N_{\text{call}} - N_{\text{threshold}})}, & \text{otherwise.} \end{cases} \tag{6}$$

Here, $I(s')$ penalizes oversivisited states, with N_{call} tracking visits, $N_{\text{threshold}}$ defining when penalties start, and β controlling penalty growth. The modified total reward is:

$$r_{\text{total}} = r - \lambda I(s'), \tag{7}$$

where λ scales the penalty. This ensures agents fully explore the environment before relying on memory-driven learning, leading to more robust policies while preserving EMU's strengths.

3.5.3 Extending EMU with Deep Q-Networks (DQN) for complex discrete scenarios in iTHOR

The EMU framework enhances multi-agent reinforcement learning (MARL) by leveraging semantic episodic memory embeddings and an episodic incentive mechanism to improve sample efficiency and prevent convergence to suboptimal policies. However, existing evaluations have focused primarily on value factorization methods such as QMIX Rashid et al. (2018), QPLEX, and CDS, as well as episodic control techniques like EMC Wang et al. (2020).

To broaden the scope of the study and assess the impact of episodic memory representation on different RL paradigms, we propose integrating Deep Q-Networks (DQN) into EMU. This will provide a comparative analysis between single-agent reinforcement learning (DQN) and multi-agent approaches, particularly in complex discrete decision-making scenarios within the iTHOR simulation environment. Given that episodic memory mechanisms can be incorporated into both single-agent and multi-agent reinforcement learning frameworks, this study aims to empirically evaluate the feasibility and effectiveness of such an integration. Specifically, we seek to test the applicability of episodic memory strategies originally designed for multi-agent settings in single-agent learning, thereby examining their influence on sample efficiency, policy optimization, and overall performance. The integration of Deep Q-Networks (DQN) allows us to address more complex discrete scenarios. Using the iTHOR library, we will evaluate the performance of DQN in these scenarios by comparing the effectiveness of different episodic memory representation techniques: Autoencoder, Random Projection, and Embeddings. This comparison will highlight how memory representation impacts learning efficiency, policy performance, and overall task success.

3.6 Computational Requirements

The reproduction experiments were conducted on a high performance computing (HPC) cluster. Specifically, we utilized both CPU and GPU processing nodes with the following configurations: (1) **CPU Node:** Intel® Xeon® Gold 6130 processor with 157 GB of DRAM. (2) **GPU Node:** AMD EPYC 64-Core processor with 1 TB of DRAM and 2 NVIDIA A100 GPUs (40 GB).

Table 3 presents the average training times across all maps. The experiments performed on the CPU node required between 3 and 6 GB of RAM, whereas those on the GPU node utilized 3 to 5 GB of RAM and 4 to 7 GB of GPU memory.

Map	Mean Time (hours)	Standard Deviation (hours)	Hardware
1c3s5z	32	6	CPU
3s_vs_5z	15	4	CPU
5m_vs_6m	19	4	CPU
6h_vs_8z	95	28	CPU
3s5z_vs_3s6z	79	27	CPU
MMM2	84	13	CPU
3_vs_1WK	65	9	GPU
CA_easy	42	8	GPU
CA_hard	96	5	GPU

Table 3: Training times for different maps

4 Results

4.1 Results reproducing original paper

4.1.1 Q1. Comparative evaluation on Starcraft II (SMAC)

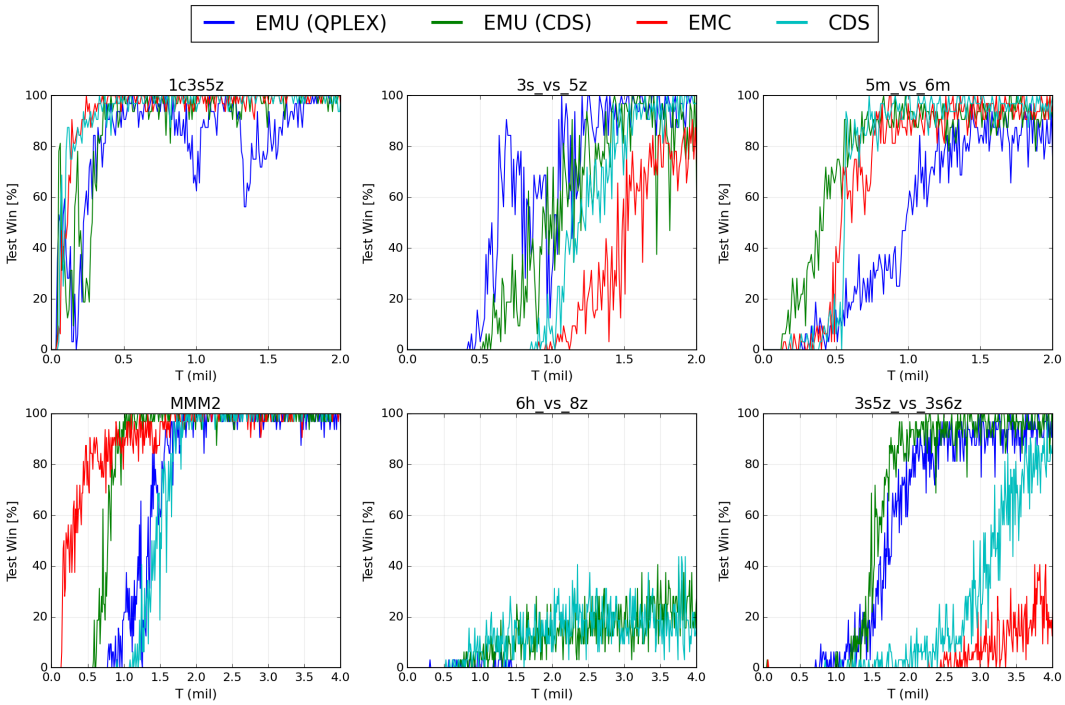


Figure 2: Reproduction of comparative evaluation on Starcraft II (SMAC)

One of the primary claims of EMU is its superior performance compared to state-of-the-art MARL frameworks. Our reproduced results, presented in Figure 2, largely support this assertion. In most scenarios tested, EMU (CDS) achieves either comparable or superior performance relative to EMC and CDS, consistently reaching a 100% test win rate while stabilizing in an equal or fewer number of steps, thus validating Claim 1.

Furthermore, EMU (QPLEX) exhibits performance on par with EMU (CDS) in the 1c3s5z and 3s_vs_5z maps. However, its effectiveness diminishes in more challenging environments, such as 5m_vs_6m, and particularly in super-hard maps like 6h_vs_8z and 3s5z_vs_3s6z, where it underperforms in comparison to both EMC and CDS. This discrepancy suggests that the stochastic nature of multi-agent reinforcement learning (MARL) tasks significantly influences policy convergence, leading to performance variations between EMU (CDS) and EMU (QPLEX).

The results indicate that the episodic incentive mechanism in EMU effectively mitigates the impact of stochasticity, enabling faster convergence, especially in EMU (CDS), fulfilling Claim 3. However, in super-hard scenarios, the increased randomness amplifies estimation errors in the Q-value function, ultimately degrading EMU (QPLEX)'s performance. These findings highlight a trade-off between algorithmic complexity and resilience to environmental uncertainty, suggesting that while EMU enhances learning efficiency, its robustness varies depending on the chosen architecture and the stochastic properties of the task.

4.1.2 Q1. Comparative evaluation on Google Research Football (GRF)

As illustrated in Figure 3, this study presents a performance comparison of EMU (QPLEX), EMU (CDS), and EMC across three different maps within the Google Research Football (GRF) environment. While the original evaluation was conducted over a range of 3 to 8 million steps, this reproduction is limited to the first 2 million steps. Nevertheless, despite this restriction, the results obtained remain largely consistent with the original findings. Specifically, EMU (CDS) consistently outperforms EMU (QPLEX) across all tested maps.

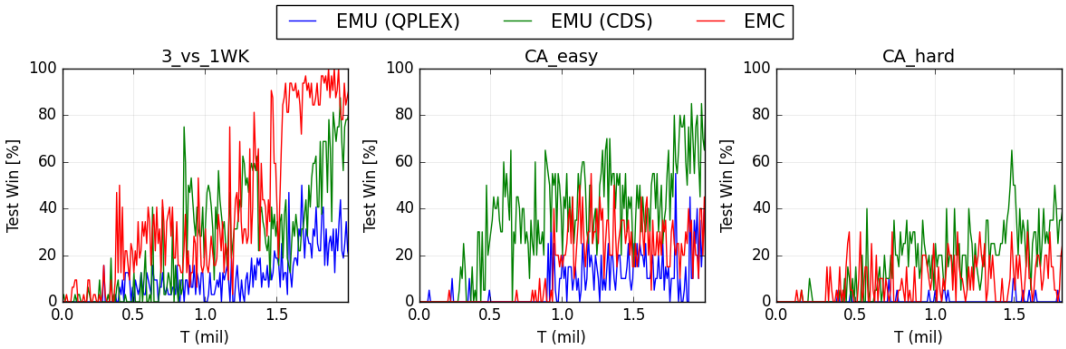


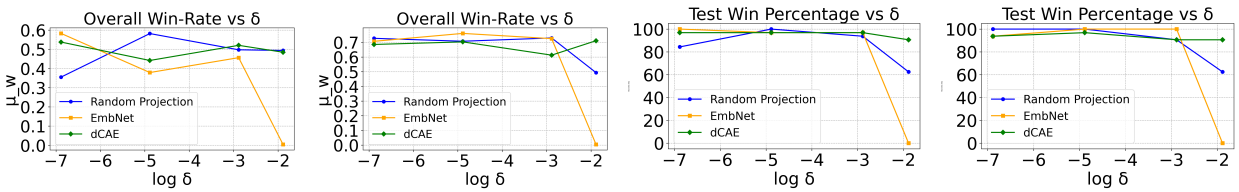
Figure 3: Reproduction of performance comparison of EMU against baseline algorithms on Google Research Football

Furthermore, EMC exhibits an inconsistent performance across different scenarios. In less complex maps, such as 3_vs_1WK, EMC demonstrates competitive results, occasionally surpassing EMU (QPLEX). However, in more challenging environments, such as CA_hard, EMC struggles to achieve stable performance, likely due to its limited adaptability in highly dynamic multi-agent interactions. In contrast, EMU (CDS) maintains a more robust learning trajectory, reinforcing the effectiveness of its episodic memory mechanism in mitigating policy instability.

These findings underscore the advantages of EMU's memory-driven approach, particularly in scenarios where traditional methods like EMC exhibit performance degradation due to the increased complexity of agent coordination, validating Claim 1.

4.1.3 Q2. Parametric and ablation study

Here, our objective is to recreate the original experiment to compare the hyperparameters and evaluate the reproducibility of the work. We carefully followed the experimental setup and configuration as recommended by the author, ensuring that all parameters and settings matched those described in the original work. To achieve this, we replicated the experiment on selected SMAC maps to measure $\bar{\mu}_w$ as a function of δ and the choice of design for f_ϕ, considering the following configurations:



(a) 3s_vs_5z (b) 5m_vs_6m (c) 3s_vs_5z (Test) (d) 5m_vs_6m (Test)

Figure 4: Comparative evaluation of experiments on SMAC maps.

Random Projection and **EmbNet**, with $\delta_1 = 1.3 \times 10^{-7}, \delta_2 = 1.3 \times 10^{-5}, \delta_3 = 1.3 \times 10^{-3}, \delta_4 = 1.3 \times 10^{-2}$.

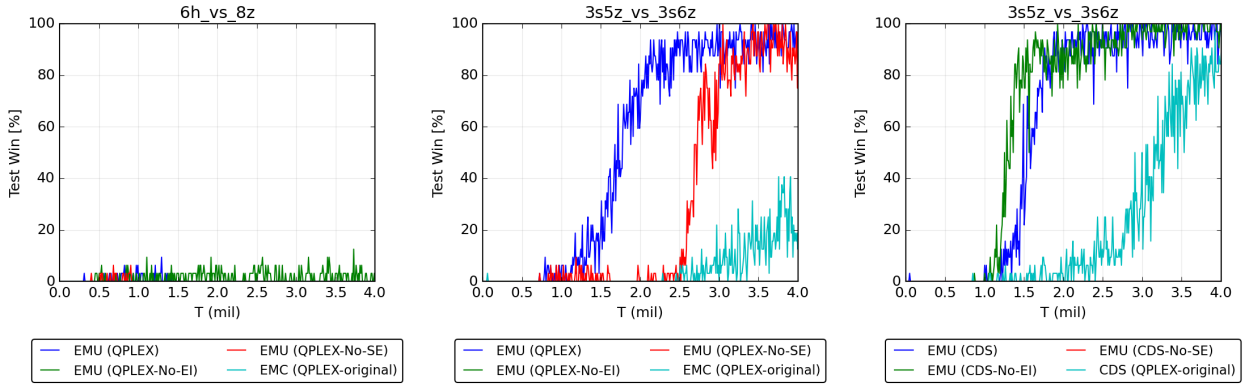
Variability Across Runs: When running the experiment multiple times, we observed that the results of *Test Win [%]* and the *overall win-rate* $\bar{\mu}_w$ exhibit significant variations. This variability may be due to several factors, including differences in the initialization of random seeds and sensitivity of the model to the values of δ and the architecture of f_ϕ. Therefore, we perform further experiments to explore Claim 2.

Impact on the Interpretation of Results: Since the values obtained in each run may differ, this suggests that the model's performance is not entirely deterministic and may be influenced by the inherent randomness of training. Our results suggest that the choice of the hyperparameter δ and the architecture of f_ϕ affects training stability. The variability across runs indicates that convergence is not entirely deterministic, highlighting the need for stabilization strategies in future research.

4.1.4 Q3. Further ablation study

Figure 5 presents the ablation study results, revealing discrepancies from the original findings. In case (a), test win rates remained below 10%, despite running all experiments twice with the provided configuration files. In case (b), EMU (QPLEX) outperformed its variations, while removing the episodic incentive (QPLEX-No-EI) resulted in zero test win rates, even after two independent runs. Similarly, in case (c), EMU (CDS) and its variation without episodic incentive (CDS-No-EI) showed comparable results, but removing state embedding led to zero test win rates, even after two executions.

Claim 3 states that the episodic incentive mechanism mitigates detrimental local convergence by reducing variation across different seeds. Our ablation study partially supports this claim: removing the episodic incentive significantly degrades EMU (QPLEX) performance, whereas its effect on EMU (CDS) is less pronounced. This difference may arise from factors such as the stochasticity of SMAC environments, especially in super-hard maps. Given MARL methods' sensitivity to random initialization and environmental variations, our findings underscore the role of episodic incentives in stabilizing performance across different runs.



(a) 6h_vs_8z SMAC (b) 3s5z_vs_3s6z SMAC (c) 3s5z_vs_3s6z SMAC

Figure 5: Reproduction of ablation studies on episodic incentive via complex MARL tasks.

4.2 Results additional experiments

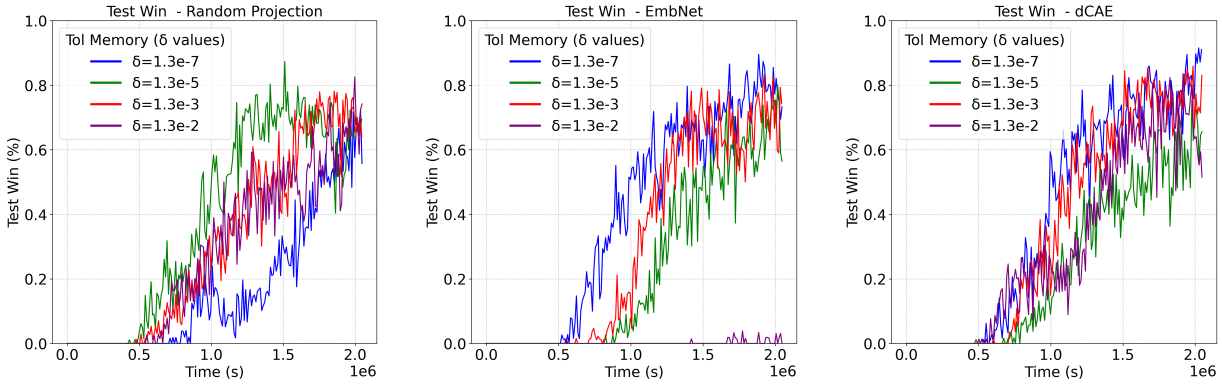
4.2.1 Parametric analysis

As described above, we conducted a comprehensive evaluation to determine how the state embedding threshold (δ) influences the performance of different memory embedding strategies in cooperative MARL. In this experiment, we compared three memory types (Random Projection, EmbNet, and dCAE) by analyzing the evolution of the Test Win over time under the algorithm's original configuration (**Figure** 7).

In the case of Random Projection, we observed that at $\delta = 1.3e-5$, the model achieves competitive performance, even surpassing the other memory types at certain points during training. However, by the end of the process, its performance declines relative to the others, suggesting its inability to consolidate semantic memories over time, which limits its long-term effectiveness.

On the other hand,in EmbNet, a clear relationship is observed between δ values and the model's performance. As δ decreases, performance improves, indicating that this parameter has a significant impact on the quality of learning. This result suggests that, in this type of memory, δ is a key factor in the model's evolution and its ability to learn optimal strategies.

However, in dCAE, no clear correlation between δ and the model's final performance is observed. Despite the fact that $\delta = 1.3e-7$ (red) and $\delta = 1.3e-2$ (blue) have considerably different values, the obtained performance is very similar. Moreover, the intermediate value $\delta = 1.3e-5$ does not lie between them in the graph, suggesting that the impact of δ in this case is much lower. This indicates that, in the context of the game, the model is not strongly affected by δ when using dCAE, and it is possible that another factor is playing a more significant role in shaping the learning dynamics. Thus, these findings validate Claim 2, demonstrating that the state embedding approach effectively enhances exploration efficiency through the proper tuning of δ, particularly evident in the EmbNet configuration.



(a) Random Projection (b) EmbNet (c) dCAE

Figure 6: Comparison of BattleWon Mean for Memory type

4.2.2 Overcoming Local Optima Through Exploration Incentives

In Figure 7, we observe that the proposed penalty function enhances early-stage performance, particularly in the Random Projection method for the $3s\_vs\_5z$ environment. The effect is most noticeable during the first 0.75 million steps, where the penalty function facilitates a faster increase in test win percentage compared to the original function. However, after 1.5 million steps, the difference diminishes, suggesting that while the penalty function aids in early exploration, long-term policy refinement remains unaffected.

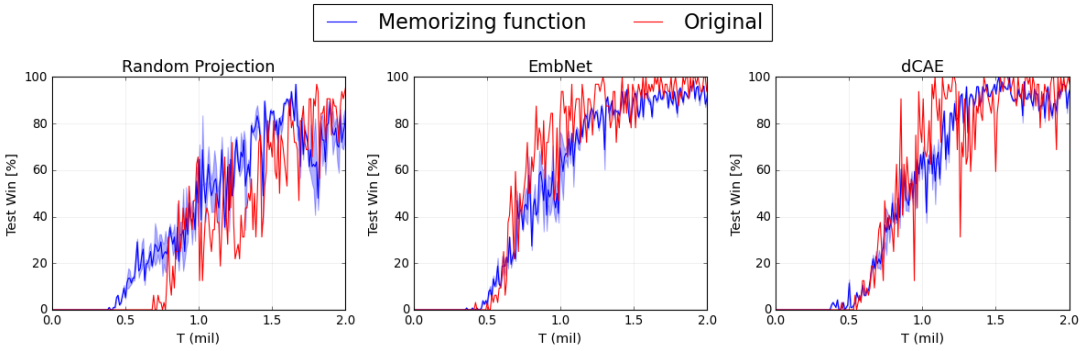


Figure 7: Comparison of our penalty function vs the original proposal in $3s\_vs\_5z$

A similar trend is seen in dCAE, though with greater fluctuations, indicating that the penalty function might introduce momentary instability in certain representations. In contrast, EmbNet shows minimal impact from the penalty function, suggesting that it may already incorporate effective generalization mechanisms. This highlights that the effectiveness of the penalty function varies across different models, benefiting high-variance methods more significantly.

Overall, these results suggest that the penalty function primarily accelerates early learning rather than improving final performance. This insight opens the possibility of adaptive penalty scaling, where the penalty gradually reduces over time to balance exploration and exploitation. A deeper variance analysis could help clarify whether the observed fluctuations in dCAE and Random Projection are due to inherent stochasticity or systematic effects of the penalty. Further discussion on local optima and additional results are provided in Appendix A.1.

4.2.3 Experimental Challenges in Integrating DQN within EMU for iTHOR

Although integrating Deep Q-Networks (DQN) into the EMU framework was initially proposed to extend its applicability to discrete environments such as iTHOR, this integration ultimately proved unfeasible. The project's design is closely aligned with SMAC and Google Research Football, which provided the foundation for EMU's episodic memory and incentive configurations. In contrast, DQN inherently relies on a replay buffer to store and sample past experiences, a requirement that conflicts with the episodic memory structure employed by EMU. This necessitated a decoupling of memory configurations, ultimately rendering the experimental setup impractical. Moreover, the existing codebase lacked sufficient robustness to accommodate significant structural modifications, and the system configuration complicated the integration with iTHOR. Another critical factor was the discrepancy in training paradigms: while DQN trains agents individually, the EMU framework is designed to accommodate both individual and cooperative training approaches, thereby requiring distinct architectural considerations. This divergence in training strategies further impeded the successful incorporation of DQN. Additionally, the comparison algorithms are distributed across multiple repositories, making standardization and reproducibility more challenging. Establishing a unified library for episodic memory-based reinforcement learning methods would facilitate benchmarking, streamline integration, and enhance research consistency.

As a result, despite initial efforts, the experimental evaluation of DQN within the EMU framework on iTHOR could not be realized, highlighting the challenges of adapting memory-centric reinforcement learning methods to heterogeneous environments. These findings underscore the importance of aligning reinforcement learning architectures with the specific demands of their target environments. The incompatibilities observed between memory management strategies and training paradigms offer valuable insights into the inherent trade-offs of adapting diverse reinforcement learning methodologies. While the integration of DQN into EMU was ultimately unsuccessful, the challenges encountered provide a promising foundation for future research aimed at developing hybrid architectures or novel approaches that reconcile these differences.

5 Discussion

As we can appreciate from results, EMU (CDS) consistently demonstrated superior stability and efficiency across various test environments. Its episodic incentive mechanism played a crucial role in mitigating stochasticity, thereby facilitating faster and more reliable convergence, validating Claim 3. Conversely, EMU (QPLEX) exhibited more variability, performing competitively in certain scenarios but displaying greater fluctuations in performance. This suggests that while EMU improves exploration efficiency, its effectiveness is contingent on the specific reinforcement learning architecture employed.

A key insight from this replication study is the sensitivity of EMU's performance to hyperparameter selection and computational architecture. Variability in results across different runs suggests that minor adjustments to parameters such as the state embedding threshold (δ) can lead to significant differences in training stability. This highlights the need for further research on robust tuning strategies to enhance model generalizability and reproducibility across different computational setups.

Despite these promising findings, the study also revealed notable challenges in reproducibility. Difficulties encountered during experimental replication-ranging from software compatibility issues to structural limitations in the provided code-emphasize the importance of improving documentation, dependency management, and code modularity. The integration of Deep Q-Networks (DQN) within the EMU framework proved infeasible, underscoring the need for greater architectural flexibility when adapting episodic memory mechanisms to diverse reinforcement learning paradigms.

5.1 Reproducibility

When recreating the graphs proposed by the authors for the third time, we found some discrepancies in the results obtained with the presented models (see Appendix A.2, Figure 10).. This is due to the inherent variability in state exploration during the training of each game. Although we followed the steps described in original paper, particularly for configuring the experiment with the 5m_vs_6m game, we observed that the algorithm is still not as efficient as it should be. Its performance is significantly affected by the randomness of the environment, suggesting that its stability and generalization could be improved.

For example, while Random Projection maintains a relatively stable win rate across different values of $\log \delta$, dCAE exhibits a sharp drop at $\log \delta = -5$, reaching almost 0.0, before recovering. Similarly, EmbNet remains stable until $\log \delta = -3$, where it drastically decreases. These fluctuations indicate that these memory models are more sensitive to specific environmental variations than others.

5.2 What was easy

Certain aspects of the implementation process were relatively straightforward. Once the appropriate environment was set up on a compatible system, the execution of the provided scripts proceeded without major complications. Additionally, the presence of predefined configuration files minimized the need for extensive manual adjustments, enabling a smoother initialization of experiments.

5.3 What was difficult

Throughout this study, several challenges hindered the reproduction and extension of the original paper. Despite strictly following the original setup and repeating the experiment several times, the results obtained did not exactly match those reported in the original study. Consequently, we contacted the authors of the original paper to adjust the setup (see Appendix A.4), but this still resulted in varying results due to the inherent variability in performance due to the stochastic nature of the problem. Additionally, the reproducibility of the model is influenced by the computational architecture, as running the same experiment on different machines can lead to variations in the results. Notably, the use of a GPU versus a CPU affects processing times and overall performance, highlighting that system configuration is a critical factor beyond just the experiment parameters.

The integration of different reinforcement learning approaches exposed methodological incompatibilities, making standardization challenging. Similarly, certain parameters impacted the model's performance inconsistently, complicating its analysis. Lastly, limitations in configuration flexibility prevented some experiments from being conducted as planned, affecting comparisons with previous studies. Further explanation are provided in Appendix A.3

6 Conclusions

This study aimed to systematically reproduce and extend the findings of the original work on Efficient Episodic Memory Utilization (EMU) for cooperative multi-agent reinforcement learning (MARL). The results confirm that EMU provides significant performance enhancements in MARL tasks by leveraging structured episodic memory representations and an adaptive incentive mechanism. These advantages were particularly evident in environments where traditional exploration strategies struggle with inefficient learning and local optima convergence.

In conclusion, while EMU effectively enhances MARL performance, further refinements in its implementation, stability, and scalability are necessary. Future work should focus on addressing the challenges identified in this study, particularly in improving cross-platform compatibility, refining memory management techniques, and optimizing hyperparameter selection for more robust performance. A standardized library for episodic memory-based MARL approaches could facilitate benchmarking and streamline future research efforts.

References

Li Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

Nikolay Dandanov, Hussein Al-Shatri, Anja Klein, and Vladimir Poulkov. Dynamic self-optimization of the antenna tilt for best trade-off between coverage and capacity in mobile networks. *Wireless Personal Communications*, 92(1):251–278, 2017.

Martin-André Dittrich and Silas Fohlmeister. Cooperative multi-agent system for production control using reinforcement learning. *CIRP Annals*, 69(1):389–392, 2020.

Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajkc, 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.

Hyungho Na, Yunkyeong Seo, and Il-chul Moon. Efficient episodic memory utilization of cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2403.01112*, 2024. URL `https://doi.org/10.48550/arXiv.2403.01112`.

Frans A. Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

Timur Samvelyan, Anuj Mahajan, Tabish Rashid, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jian Wang, Ziyu Ren, Tong Liu, Yang Yu, and Cheng Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020. URL `https://arxiv.org/abs/2008.01062`.

Marco A. Wiering et al. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'2000)*, pp. 1151–1158, 2000.

Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015. URL `https://arxiv.org/abs/1502.04759`.

Lianmin Zheng, Jianye Chen, Jiawei Wang, Jinchi He, Yujing Hu, Yuxiao Chen, Cheng Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *arXiv preprint arXiv:2111.11032*, 2021. URL `https://arxiv.org/abs/2111.11032`.

A Appendix

A.1 Further Results Of Local Optima Through Exploration Incentives

Our goal is to demonstrate that the **MARL algorithm** can overcome local optima by prioritizing states with higher past rewards. However, this approach may limit exploration of better alternatives. To address this, we introduce an additional incentive to encourage new trajectory exploration and mitigate decision-making minimization.

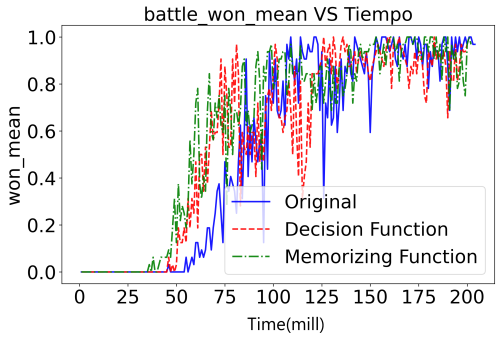
For this experiment, we set the penalty function parameters as follows:

$$N_{\text{threshold}} = 10, \quad P_{\max} = 0.5, \quad \beta = 0.0009, \quad \delta_1 = 1.3 \times 10^{-7}, \quad \text{Memory Type: } \textbf{DCAE}.$$

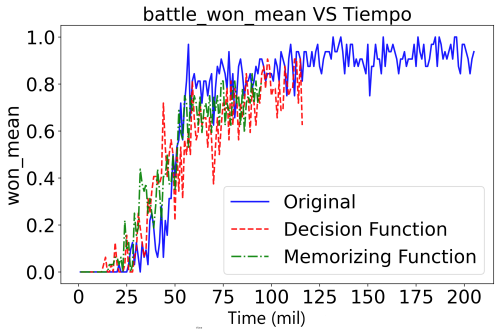
The penalty function is defined as:

$$I(s') = \begin{cases} 0, & \text{if } N_{\text{call}} \leq N_{\text{threshold}} \\ 1 - e^{-\beta(N_{\text{call}} - N_{\text{threshold}})}, & \text{if } N_{\text{call}} > N_{\text{threshold}} \end{cases}$$

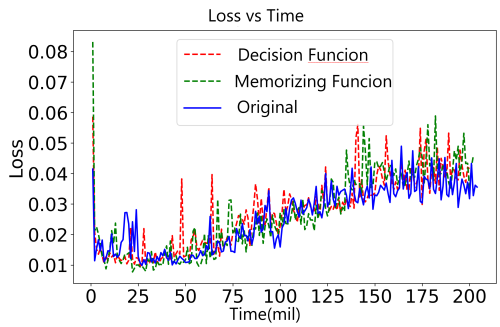
Figure 9 presents performance comparisons. The red function, used as a **Decision Function**, applies penalties to modify rewards while storing unpenalized values to compare strategies and prevent over-penalization. Two cases were considered: storing versus not storing penalized values. Our results indicate that when penalties are not stored, decisions improve in some cases, but feedback absence causes **overfitting** between steps **100 and 125**. Storing penalties led to better performance in **3s_vs_5z** and **5m_vs_6m** scenarios. The **loss graph** highlights that the **Memorizing Function** (green) applies a more balanced penalty than the Decision Function.



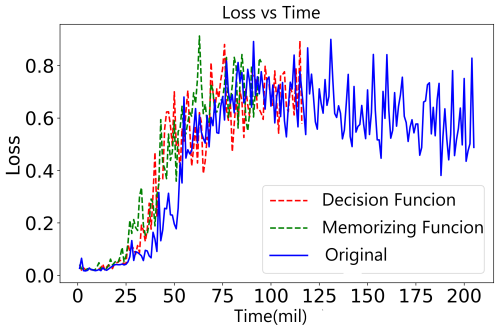
(a) 3s_vs_5z



(b) 5m_vs_6m



(c) 3s_vs_5z

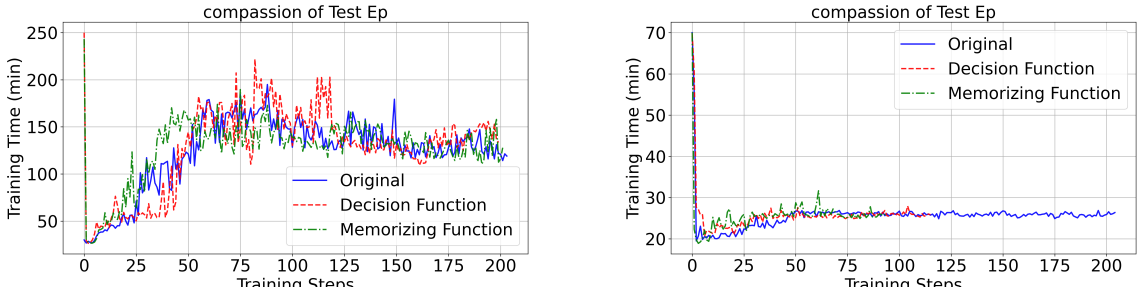


(d) 5m_vs_6m

Figure 8: Penalty loss corporation.

During the first **75k steps**, the penalty function enhances performance, but afterward, it converges to the original function. This aligns with expectations, as early training favors exploration. Over time, the original algorithm reaches optimal learning, confirming our hypothesis in the initial stages while progressively favoring high-reward states.

To validate our approach, we conducted three runs per experiment with varying configurations to compute variance. Figure 9 illustrates variance results, providing deeper insights. As step count increases, the original algorithm efficiently differentiates between moves. However, despite its **O(1) complexity**, our modification introduces overhead, with execution time increasing significantly beyond **80 steps**.



(a) 3s_vs_5z (b) 5m_vs_6m

Figure 9: execution time.

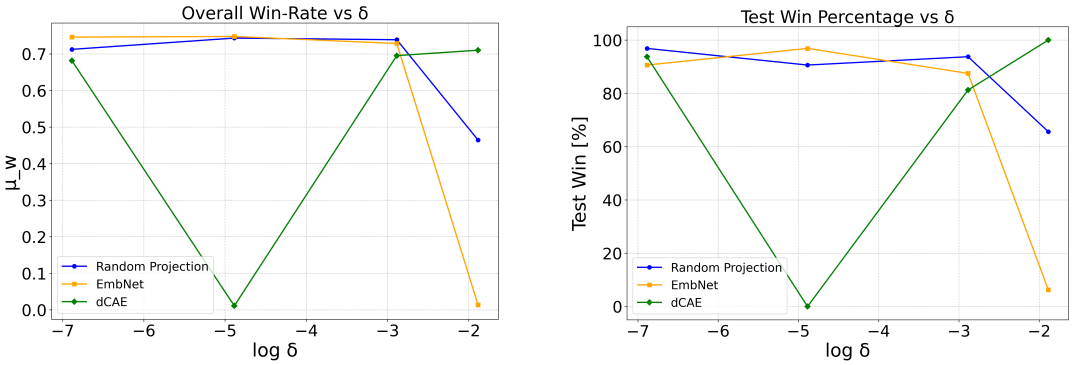
Efforts to optimize the code revealed that **lack of standardization and modularity** hampers readability and modifications. A structured architecture would facilitate analysis and experimentation.

Both **semantic and episodic memory** contribute to AI training stability in later stages, with or without the penalty function. A more effective strategy would emphasize early-stage exploration to accelerate learning and then seamlessly integrate the original implementation beyond **80 steps**. Furthermore, optimizing **modularity, standardization, and architecture** is critical for improved performance and research progress.

A.2 Additional Parametric Results

When recreating the parametric and ablation study proposed by the authors for the third time, we observed several discrepancies between our results and those reported in the original paper. This is due to the inherent variability in state exploration during the training of each game. Although we used the same yaml files as in the original paper, particularly for configuring the experiment with the **5m_vs_6m** game, we observed that the algorithm is still not as efficient as it should be. Its performance is significantly affected by the randomness of the environment, suggesting that its stability and generalization could be improved.

For example, while the Random Projection model maintains a relatively stable win rate across different values of $\log \delta$, the dCAE model exhibits a sharp drop at $\log \delta = -5$, reaching almost 0.0, before recovering. Similarly, the EmbNet model remains stable until $\log \delta = -3$, where it drastically decreases. These fluctuations indicate that these memory models are more sensitive to specific environmental variations than others.



(a) Overall win-rate according to δ (b) Final win-rate according to δ

Figure 10: Performance comparison on SMAC maps.

A.3 Specific Limitations

A.3.1 Challenges regarding the code structure and the instructions provided in the GitHub repository

The installation process was hindered by unclear instruction and missing dependency specifications in the Github repository. As a result, we encountered multiple issues with package compatibility and dependency resolution. To address these problems, we had to rely on the source repositories of SMAC and Google Research Football to manually resolve dependencies and ensure proper installation. Additionally, the required version of *starcraft.py* differed from the one provided in the SMAC repository, yet no guidance was available on how to install the correct version. This lack of comprehensive documentation significantly increased the time required to set up the environment and replicate the experiments. This lack of comprehensive documentation significantly increased the time required to set up the environment and reproduce the experiments.

A.3.2 Challenges encountered when utilizing non-Linux-based operating systems

Our initial installation attemps were conducted on a Windows laptop equipped with an NVIDIA GeForce RTX 4050 GPU. However, the estimated runtime for a single experiment on an easy SMAC map ranged from 4 to 8 days, making this approach impractical. As a result, we transitioned to an HPC cluster to leverage greater computational resources and enable parallel execution of multiple experiments.

A.3.3 Challenges related to the handle of Google Research Football

While we resolved the package related installation issues on Windows, transitioning to the HPC cluster introduced new challenges due to its Linux-based environment. The installation process was further complicated by the cluster's restrictions, which prevented us from installing system level packages required for Google Research Football due to the lack of root access. To overcome this limitation, we initially disabled all GFootball-related code and focused solely on SMAC experiments. Weeks later, Apptainer was installed on the cluster allowing us to containerize the environment. We built a Docker image with all necessary dependencies and executed the experiments within an Apptainer container. Although this solution enabled us to run the experiments, it added complexity to the setup and maintenance of the experimental workflow.

A.4 Contact with the authors

We contacted the authors to clarify specific hyperparameter configurations and training protocols, particularly for the superhard SMAC maps. They emphasized the necessity of extending training to at least five million steps (t_max = 5M) instead of the two million steps (t_max = 2M) initially used. According to their feedback, prolonged training is essential for ensuring convergence in complex tasks, as shorter runs may not fully capture the performance potential of the models.

Additionally, they highlighted the importance of maintaining consistency in baseline configurations by strictly following the hyperparameter settings outlined in their official repository or paper. To ensure fair comparisons, they recommended enabling the "emu_circle=2" setting in default.yaml. In light of this guidance, several models were retrained to align with these revised specifications, thereby ensuring methodological fidelity to the original study.

A.5 YAML Configuration Changes

In our efforts to replicate the ablation study, we modified only two parameters within the YAML configuration file: `memory_emb_type` and `optimality_incentive`. Since EMU was implemented on top of both the original QPLEX and CDS frameworks, separate YAML configurations were provided for each variant to reproduce Figure 5. Consequently, we adjusted these parameters in both configuration files. The modifications are summarized in the following tables.

Table 4: Modified YAML Parameters Across Experiments in EMU_sc2

Parameter	**EMU (QPLEX)**	**EMU (QPLEX-No-EI)**	**EMU (QPLEX-No-SE)**	**EMC (QPLEX-original)**
`memory_emb_type`	3	3	1	1
`optimality_incentive`	True	False	True	False

Table 5: Modified YAML Parameters Across Experiments in EMU_sc2_cds

Parameter	**EMU (CDS)**	**EMU (CDS-No-EI)**	**EMU (CDS-No-SE)**	**CDS (QPLEX-original)**
`memory_emb_type`	3	3	1	1
`optimality_incentive`	True	False	True	False