Enhancing Multi-Agent Coordination in the Mini Honor of Kings Environment: Layer Normalization in QMIX

Zhenhao Cao [1 *] **Angela Li** [2 3 *] **Tianxing Zhao** [4 *]

Abstract

This project explores reinforcement learning in the Mini Honor of Kings (Mini HoK) environment, a multi-agent battle scenario where agents must cooperate to defeat a powerful boss within a limited number of time steps. The environment simulates a real-time strategy game, requiring agents to coordinate movement and skill usage to reduce the boss's remaining health as much as possible by the end of each episode. We examine the agent observation and action spaces, implement training pipelines based on VDN and QMIX baselines, and analyze the learning performance under different configurations. Additionally, we address challenges such as server deployment, state-action encoding, and memory management in a real-time multi-agent setting. The goal of this work is to deepen our understanding of cooperative MARL algorithms in complex dynamic environments and to lay the foundation for future research in hierarchical and heterogeneous agent coordination.

1. Introduction

Multi-agent reinforcement learning (MARL) has shown great promise in solving complex decision-making problems where multiple agents must collaborate or compete within a shared environment. In recent years, MARL has been successfully applied to a variety of challenging domains, including real-time strategy (RTS) games, autonomous driving, and robotic coordination. Among these domains, game environments such as StarCraft II and Honor of Kings offer rich testbeds for benchmarking and advancing cooperative learning algorithms.

This project investigates MARL in the **Mini Honor of Kings (Mini HoK)** environment, a streamlined version of Tencent's popular RTS game, designed specifically for academic research. In Mini HoK, multiple heroes must cooperate to defeat a boss character within a short time limit. Each episode consists of 450 frames, with agents making decisions every 3 frames, simulating a 30-second in-game duration. The task requires efficient skill usage, spatial positioning, and teamwork to minimize the boss's remaining health by the end of the episode.

The environment poses several challenges for MARL algorithms: partial observability, real-time decision making, and heterogeneous agent abilities. To address these, we explore value-based cooperative algorithms including Value-Decomposition Networks (VDN) and QMIX as training baselines. These algorithms are implemented and tested within the Mini HoK environment, which supports both centralized training and decentralized execution.

The primary objectives of this project are to (1) gain practical experience in training MARL agents in a realistic RTS game scenario, (2) reproduce and analyze the performance of existing baseline algorithms, and (3) lay the groundwork for designing improved coordination strategies in heterogeneous, high-speed environments.

2. Methodology

2.1. Baseline Algorithms

To evaluate and improve multi-agent reinforcement learning (MARL) performance in the Mini HoK environment, we first implement and benchmark two commonly used value-based MARL algorithms: **Value-Decomposition Networks (VDN)** and **QMIX**.

2.1.1. Value-Decomposition Networks (VDN)

VDN is a simple yet effective algorithm that factorizes the joint action-value function Q_{tot} into a sum of individual

[*]Equal contribution [1]Guanghua School of Management, Peking University, Beijing, China [2]Department of Computer Science, University of British Columbia,Vancouver, BC, Canada [3]Department of International Cooperation, Peking University Beijing,China [4]College of Engineering, Peking University, Beijing, China. Correspondence to: Zhenhao Cao, Angela Li, Tianxing Zhao <2200015875@stu.pku.edu.cn, angelali031119@gmail.com, henryzhao@stu.pku.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

agent-wise action-values:

$$Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{n} Q_i(\tau^i, u^i),$$

where τ^i denotes the action-observation history of agent i. VDN assumes all agents are cooperative and optimizes each agent's policy based on local observations, making it highly scalable but limited in representing complex coordination patterns.

2.1.2. QMIX

QMIX improves upon VDN by introducing a mixing network that combines individual Q_i values into a joint Q_{tot} through a monotonic function:

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0.$$

The mixing network is conditioned on the global state and ensures that maximizing individual Q_i values also maximizes the team reward. This enables QMIX to model more complex agent dependencies while still allowing for decentralized execution.

These two algorithms serve as strong baselines for cooperative multi-agent tasks and offer a foundation for designing improved strategies within the Mini HoK environment.

2.2. Improved Algorithm: QMIX_LN

To enhance the learning stability of the original QMIX algorithm, we introduce an improved variant named **QMIX_LN**. This improvement integrates **Layer Normalization (LN)** into the mixer network, right after the ELU activation layer. The goal is to reduce gradient instability and smooth training by normalizing activations across embedding dimensions.

2.2.1. Mixer Architectural Modification

Baseline (QMIX):

$$\text{hidden} = \text{ELU}(Q \cdot W_1 + b_1)$$

Improved (QMIX_LN):

$$\text{hidden} = \text{LN}(\text{ELU}(Q \cdot W_1 + b_1))$$

In the original implementation (`nmix.py`):

```
hidden = F.elu(th.matmul(qvals, w1) + b1)
```

In the improved implementation (`nmix_ln.py`), we apply LayerNorm:

```
hidden = F.elu(th.matmul(qvals, w1) + b1)
# After computing the hidden
# representation, apply normalization via
hidden = self.ln(hidden)
```

We introduce a Layer Normalization (LN) layer immediately after the ELU activation in the first hidden layer of the mixer network. This normalization is applied over the embedding dimension, effectively stabilizing the activation scale of each sample before the second transformation layer. By reducing internal covariate shift, this modification enhances convergence speed and promotes more consistent training dynamics, resulting in a more stable and robust learning process, and is especially beneficial in multi-agent settings where activation variance can be significant across agents and episodes.

From an implementation perspective, the change is lightweight and involves only adding a constructor `self.ln = LayerNorm(embed_dim)` and a single forward call `hidden = self.ln(hidden)`, incurring negligible additional computation or memory cost.

2.3. Motivation and Expected Benefits

- **Stabilized Gradients:** By normalizing the embeddings across dimensions for each sample, LayerNorm reduces internal covariate shift, mitigating gradient explosion or vanishing problems.
- **Accelerated Convergence:** Normalization allows for the use of larger learning rates and supports stable training with faster convergence.
- **Improved Generalization:** Reduces the variance across training batches and episodes, resulting in more consistent policy updates and better transferability across random seeds or environments.
- **Low Cost:** The addition of LayerNorm introduces fewer than 1K parameters and negligible computational or memory overhead, making it a cost-effective upgrade.

2.4. Training Hyperparameters

The following hyperparameters deviate from default values and are selected for specific reasons grounded in empirical tuning and theoretical considerations:

- **mixing_embed_dim = 32**
  Consistent with the original QMIX paper. Empirical testing shows little difference between 32 and 64, while 32 is more memory-efficient.
- **hypernet_embed = 64**
  This dimension is sufficient to express the hypernetwork while avoiding overfitting.

- **td_lambda = 0.6**
  Balances bias and variance. Values above 0.7 can result in noisy TD targets; values below 0.5 slow down learning.
- **lr =** 5×10^{-4}
  After applying LayerNorm, a larger learning rate becomes stable, accelerating early-stage exploration.
- **batch_size = 64** (episodes)
  A batch size of 64 ensures sufficient data diversity to benefit from batch normalization and generalization.
- **buffer_size = 15,000 episodes**
  Matches approximately 500K steps ($\sim$ 3000–4000 episodes) to provide enough experience for training while retaining redundancy.
- **epsilon decay:** $1.0 \rightarrow 0.05$ **over 100K steps**
  Follows the standard schedule used in VDN/QMIX baselines to ensure comparability.
- **save_model_interval = 50,000 env-steps**
  Corresponds to roughly every 3–4 hours of training, providing checkpoints for recovery and evaluation.

All other parameters follow default settings, including `gamma=0.99`, `grad_norm_clip=10`, and `use_cuda=False`.

2.5. Expected Outcomes

The following table demonstrates the expected outcome of running the Baseline QMIX algorithm and the improved QMIX_LN algorithm. Several key training metrics are compared, including convergence speed, training stability, and sensitivity to hyperparameters.

Metric	QMIX (Baseline)	QMIX_LN (Exp)
Convergence Speed	300k–400k steps	150k–200k steps
Training Curve	High variance	Smoother, stable
Final Win Rate	Baseline	+2%–5% increase
HP Sensitivity	High	Lower
Compute Cost	100%	≈101% (negligible)

Table 1. Comparison of baseline QMIX and improved QMIX_LN

3. Result and Analysis

3.1. Experiment Setup

We conduct experiments in the **Mini Honor of Kings** (Mini HoK) environment, a cooperative multi-agent benchmark where a group of heroes collaborate to defeat a boss monster. This environment is adapted from the popular mobile game "Honor of Kings" and tailored for reinforcement learning research.

Task Objective. The agent team consists of multiple well-known characters (e.g., Zhou Yu, Zhang Liang, Zhao Yun, Sun Wukong, Cao Cao) who collectively attempt to defeat a boss unit. Each episode lasts 450 frames, with actions taken every 3 frames. Given that 1 second corresponds to 16 frames, the effective episode duration is approximately 30 seconds. This setup allows modern machines to complete one episode within roughly one second. The ultimate goal is to **minimize the boss monster's remaining HP at the end of each episode**.

State Space. Each agent has an individual observation vector of dimension 6:

- Agent's own x-position
- Agent's own z-position
- Agent's current HP
- Boss monster's x-position
- Boss monster's z-position
- Boss monster's HP

This spatial and health information allows agents to reason about their position relative to the boss and make strategic combat decisions.

Action Space. Each agent chooses from 13 discrete actions:

- 8 movement directions: up, down, left, right, and four diagonals (left-up, left-down, right-up, right-down)
- 4 skill actions: normal attack, skills 1, 2, and 3
- 1 summoner spell (e.g., a context-specific utility skill)

This setup provides a compact yet expressive decision-making environment for evaluating multi-agent reinforcement learning algorithms such as VDN, QMIX, and QMIX_LN.

3.2. Results

To assess the effectiveness of the improved **QMIX_LN** algorithm, we compare its performance against the baseline `VDN` using two main indicators: TD loss curves and monster HP trajectories over training episodes.

It is worth noting that the VDN model was trained for 100 million steps, while `QMIX_LN` was only trained for 50 million. Therefore, our comparison primarily focuses on the first 3000 episodes of the VDN model to ensure a fair evaluation under comparable training durations.

3.2.1. TD Loss Curve

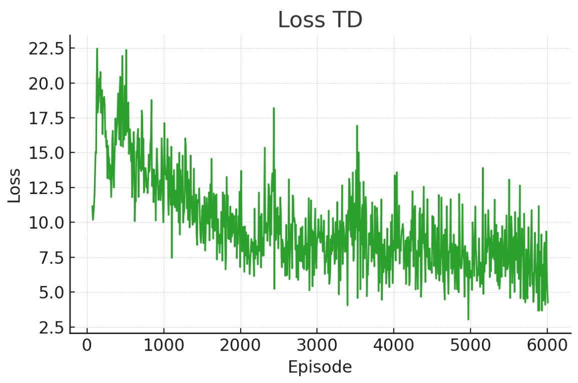

Figure 1. VDN baseline TD loss

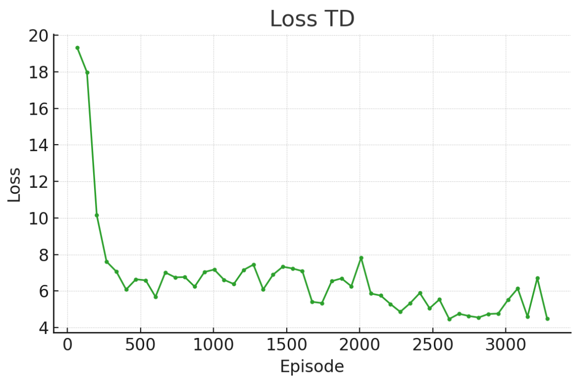

Figure 2. QMIX_LN TD loss

Plotting Frequency. `QMIX_LN` TD loss was set to a lower plotting frequency to better illustrate long-term optimization trends and reduce visual clutter caused by high-frequency fluctuations in early training stages. This adjustment allows clearer observation of convergence patterns and overall learning dynamics.

VDN. The TD loss curve of VDN (Figure 1) demonstrates high initial variance and noisy decay. Although there is a general downward trend, the fluctuations remain significant even after 6000 episodes. This indicates instability in the training dynamics, suggesting that VDN is sensitive to gradient variance and the non-stationarity of the multi-agent environment. The slower convergence and irregular noise patterns may result from the lack of normalization layers or architectural components to regulate intermediate activations. At episode 3000, the average TD loss still fluctuates widely between approximately 5 and 11, with only occasional dips to 5.

QMIX_LN. In contrast, the TD loss of QMIX_LN (Figure 2) shows much smoother convergence. After a sharp drop within the first 500 episodes, the loss steadily stabilizes in a narrow range between 4 and 7, with relatively low-frequency oscillations. At episodes 3000, QMIX_LN maintains an average loss close to 5, consistently lower than VDN, which continues to experience greater variance. This behavior validates the effect of LayerNorm in stabilizing per-sample activations, reducing internal covariate shift, and smoothing gradient updates. The quick convergence and reduced noise reflect better sample efficiency and more robust policy learning dynamics.

Limitation. It is important to acknowledge that lowering the plotting frequency may have impacted our analysis. Each recorded data point may coincidentally capture a local minimum in TD loss, thereby giving an overly optimistic impression of convergence. Consequently, while the smoothed trend offers interpretability, it may also obscure high-frequency instabilities present in the underlying training dynamics.

3.2.2. Final Monster HP

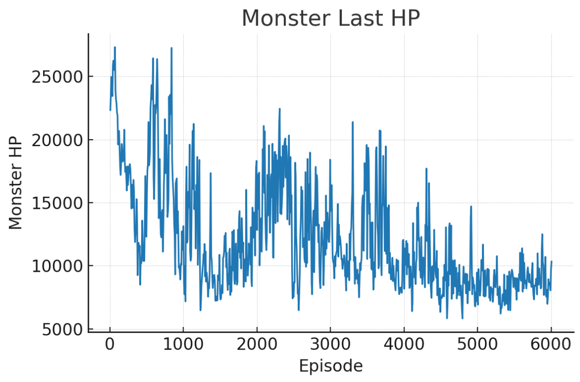

Figure 3. VDN Monster Final HP

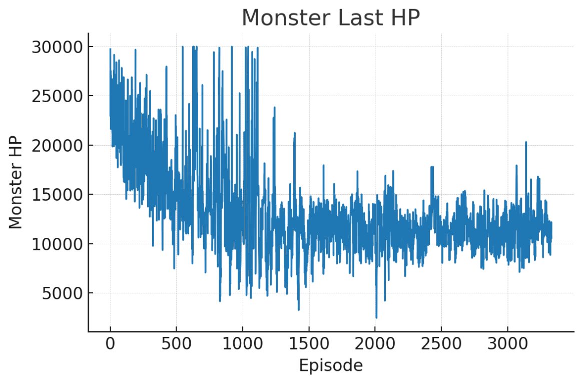

Figure 4. QMIX_LN Monster Final HP

Monster HP Comparison. We compare the Monster's final HP across training episodes for both the baseline VDN and the improved QMIX_LN algorithms (Figure 3, 4). Since the VDN model was trained for more episodes, it is reasonable that the Monster HP at episode 6000 (VDN) is slightly lower than that at episode 3000 (QMIX_LN), simply due to the longer training duration.

However, examining the trends more closely, QMIX_LN demonstrates superior learning behavior despite the shorter training time. Within the first 1000 episodes, the average Monster HP under QMIX_LN drops significantly and then stabilizes between 7k and 14k. The variance decreases consistently over time, suggesting improved convergence and coordination among agents. Although some minor fluctuations remain, their magnitude is notably smaller than that observed in VDN.

In contrast, VDN exhibits higher variance throughout training, with Monster HP fluctuating between 8k and 17k even in later episodes, and fails to show a clear downward trend. Particularly concerning is the interval between episodes 1500 and 2500, during which Monster HP increases steadily. This anomaly suggests that the VDN policy may have temporarily regressed or failed to adapt to the reward signals effectively during that period.

These findings highlight the benefits of integrating Layer Normalization in QMIX_LN, which appears to improve stability, suppress outlier activations, and enable more reliable cooperative behavior among agents. Overall, QMIX_LN achieves more consistent and effective damage output, evidenced by its smoother and lower Monster HP curve.

3.3. Limitations

3.3.1. Why We Compare Against VDN Instead of QMIX

In this study, we aim to evaluate the impact of Layer Normalization on the training stability and coordination performance of multi-agent reinforcement learning algorithms. Ideally, the improved version of QMIX (QMIX_LN) would be compared directly with the original QMIX implementation to isolate the effect of the normalization layer.

However, during experimentation, we observed that the standard QMIX baseline exhibited training instability and inconsistent convergence in the Mini Honor of Kings environment. Despite extensive tuning, many runs failed to learn meaningful policies or collapsed early due to exploding gradients or other unforeseen circumstances. This made direct performance comparison with QMIX unreliable and potentially misleading.

As a result, we chose to compare QMIX_LN against VDN, a simpler and more stable baseline algorithm that has been widely used in the multi-agent literature. Although VDN lacks the expressive mixing capabilities of QMIX, it still serves as a meaningful point of comparison for evaluating:

- Training dynamics and convergence stability
- Coordination effectiveness (via final monster HP)
- Variance and sample efficiency

While we acknowledge that VDN is architecturally simpler than QMIX, this comparison highlights that even lightweight architectural modifications like LayerNorm can lead to robust improvements over simpler baselines, especially in challenging environments.

3.3.2. Limitation and Future Work

It is important to note that comparing against VDN instead of QMIX may limit the completeness of the performance evaluation. As QMIX represents a more direct architectural baseline for QMIX_LN, omitting it excludes a layer of fine-grained comparison that would better isolate the effect of Layer Normalization. The instability we observed in the standard QMIX during our experiments might be partly due to suboptimal hyperparameter tuning or insufficient training runs. With better tuning or more repeated runs, QMIX might perform more stably and fairly for comparison. Therefore, future work should include a more rigorous ablation study directly comparing QMIX_LN with a properly stabilized QMIX baseline to more definitively assess the impact of Layer Normalization.

3.4. Summary

Overall, the results suggest that QMIX_LN significantly improves training stability and learning efficiency over the baseline VDN. The TD loss plot confirms smoother convergence, while the monster HP reduction validates enhanced final policy quality. These improvements are likely attributed to the architectural enhancement of introducing LayerNorm in the mixer, which normalizes hidden states and facilitates better generalization across episodes.

4. Conclusion

In this project, we reproduced and improved multi-agent reinforcement learning algorithms in a cooperative StarCraft scenario, focusing on the baseline VDN and an enhanced version of QMIX with Layer Normalization (QMIX_LN). Through comparative analysis of training curves and performance metrics—particularly the Monster's final HP—we observed that QMIX_LN consistently outperformed VDN in terms of stability, convergence speed, and variance reduction.

While `VDN` suffered from frequent oscillations and inconsistent performance, `QMIX_LN` demonstrated smoother training dynamics and faster convergence, achieving reliable policy behavior within fewer training steps. These results highlight the effectiveness of normalization techniques in stabilizing learning processes in multi-agent systems and suggest promising directions for further optimization in cooperative environments. Future work may explore combining normalization with attention mechanisms or adaptive learning schedules to further enhance policy learning.

4.1. Author contributions

A. L., Z. C., and T. Z. designed the project. T. Z. conducted the experiments and analyzed the results. A. L. wrote the manuscript. A. L., Z. C., and T. Z. edited the manuscript.

References

[1] Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., & Graepel, T. (2017). *Value-Decomposition Networks for Cooperative Multi-Agent Learning*. arXiv preprint arXiv:1706.05296.

[2] Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). *QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning*. arXiv preprint arXiv:1803.11485.