Loc-FACMAC: Locality Based Factorized Multi-Agent Actor-Critic Algorithm for Cooperative Tasks

Anonymous authors
Paper under double-blind review

Abstract

In this work, we present a novel cooperative multi-agent reinforcement learning method called **Loc**ality based **Fac**torized **M**ulti-Agent **A**ctor-**C**ritic (Loc-FACMAC). The existing state-of-the-art algorithms, such as FACMAC, rely on the global reward information for critic training. However, in a distributed multi-agent system, the global reward is overgeneralized. The global reward cannot accurately reflect the influence of individual agents' actions, resulting in the mixer's poor performance in assigning credit. We introduce the idea of *locality* into critic learning to connect the strongly related agents into partitions. Agents in the same partition have a more significant impact retained within the partition itself. Thus, agents learning from the local reward can provide a more precise evaluation of the policy. This technique prevents the agent using information from unrelated agents and also helps to deal with the curse of dimensionality due to multiple agents. Loc-FACMAC further improves the efficiency of learning by introducing locality to the actor update as well. We evaluate the performance of Loc-FACMAC on three environments: Multi-cartpole, the StarCraft Multi-Agent Challenge, and Bounded-Cooperative-Navigation. We explore the impact of partition sizes on the performance and compare the result with baseline MARL algorithms such as LOMAQ, FACMAC, and QMIX. The experiments reveal that, if the locality structure is defined properly, Loc-FACMAC outperforms these baseline algorithms up to 45% , indicating that exploiting the locality structure in the actor-critic framework improves the MARL performance.

1 Introduction

Multi-Agent Reinforcement Learning (MARL) is a framework (Foerster et al., 2017; Tan, 1993) that enables a group of agents to learn team behaviors by interacting with an environment. Recently, the impact of MARL has become quite evident in a range of areas (Jiang et al., 2022; Chu et al., 2019; Zhang et al., 2022; He et al., 2016). In the field of reinforcement learning, multi-agent coordination plays a crucial role in various applications such as cooperative searching, human-robot interaction, product delivery, and soccer (Ji et al., 2022; Qie et al., 2019; Vorotnikov et al., 2018; Ota, 2006; Jiménez et al., 2018). In these scenarios, agents often rely on local observations to make decisions that benefit the entire team. In many recent MARL approaches, the access of global rewards is assumed. However, the assumption does not hold as agents often need to learn cooperative behaviors based on local observations and local or group rewards. In this paper, we propose a new MARL technique enabling agents to leverage the locality for multi-agent coordination.

Similar to single-agent RL, most existing MARL frameworks can be classified into two categories: *value-based* (Watkins & Dayan, 1992; Sunehag et al., 2017; Rashid et al., 2018; 2020; Son et al., 2019; Kortvelesy & Prorok, 2022; Xu et al., 2021) approaches and *actor-critic* approaches (Konda & Tsitsiklis, 1999; Peng et al., 2021; Wang et al., 2020). In value-based approaches, agents learn to estimate an action value function by exploring the action space and choosing the action with the maximum action value. value-based approaches are commonly used in MARL, in part, because QMIX (Rashid et al., 2018) has shown the potential of solving complex coordination problems such as the Star-Craft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). The core idea of QMIX is to utilize a monotonic mixer to estimate the joint-action value Q_{tot} from the individual state-action values Q_i. The joint-action value function evaluates the agents' performance and

modifies the policy using back-propagation. The idea of QMIX has been extended in several ways, including WQMIX (Rashid et al., 2020) and Qtran (Son et al., 2019).

Although value-based approaches have shown the potential for solving complicated tasks, the curse of dimensionality prevents the approach from being applied to large-scale tasks. For instance, in QMIX-type approaches, when the number of agents increases, the joint-action space exponentially increases. Meanwhile, value-based approaches must compare all action values, requiring significant search time. Therefore, value-based approaches are not an efficient method for training many agents at once. In contrast, the actor-critic approach directly learns a policy and decides actions from policy rather than maximizing the Q value function. For instance, MADDPG (Lowe et al., 2017) is one of the classical approaches in this category. The actor in MADDPG learns to generate the optimal action and sends the chosen action to a critic for evaluating the value of the chosen action. Then, the actor adjusts the policy according to the score given by the critic. MADDPG can reduce the training time, but agents update their policy via separate policy gradient while assuming that actions of all other agents are fixed. Therefore, the policy commonly falls into the sub-optimal solution. A recently proposed novel algorithm FACMAC (Peng et al., 2021) resolves the limitations of solely value-based and only actor-critic approaches by combining both of them into one algorithm. In FACMAC, QMIX is used for the critic update, and a centralized policy gradient is used for the actors during training. FACMAC is shown to outperform QMIX and MADDPG on different tasks within the SMAC environment in Peng et al. (2021).

Table 1: Comparison of different approaches.

Algorithm	Num. of Mixers	Critic	Actor
QMIX (Rashid et al., 2018)	1	✓	×
LOMAQ (Zohar et al., 2022)	K	✓	×
FACMAC (Peng et al., 2021)	1	✓	✓
Loc-FACMAC (This work)	K	✓	✓

We note that, interestingly, a common feature of most existing MARL approaches is that they aim to maximize a common global reward. However, in various practical applications, it is possible that one agent's actions may not have any effect on another agent. For example, in a task of surveillance by a group of agents (Kolling & Carpin, 2008), if two agents are quite far from each other, it makes sense to assume that their actions will not affect each other. This idea is explored in a recent paper for value function-based MARL approaches, in which the authors proposed the LOMAQ algorithm (Zohar et al., 2022). LOMAQ presents a multi-mixer approach to accelerate the training process by exploiting the locality of the rewards by defining a partition (a subset of agents) across the network of agents. LOMAQ provides theoretical guarantees under fully observable settings that maximizing the global joint-action value is equivalent to maximizing the action value in each partition. The partition's action value reflexes the performance of the partition so each agent can learn a local policy that maximizes the local reward of that partition instead of focusing on maximizing the global reward. The feedback in each partition only updates the most correlated agents' actor-critic networks.

In this work, we extend the benefits of *locality* to actor-critic methods and propose a novel locality-based actor-critic method called **Loc**ality based **Fac**torized **M**ulti-Agent **A**ctor-**C**ritic Algorithm (Loc-FACMAC) for cooperative MARL. The critical features of Loc-FACMAC relative to existing methods are summarized in Table 1. Loc-FACMAC inherits the structure of FACMAC in which each agent has an actor and a critic. Similar to FACMAC, Loc-FACMAC separates the training process of critics and actors, so the critic can precisely evaluate the quality of action value without being influenced by the choice of actions. The actor learns from the high-quality local action value function and quickly converges to an optimal policy. Besides that, the actor in the actor-critic approach outputs an action instead of the action value. It is extremely useful when the action space is large since it can discard the search time of the highest action value from all actions. Moreover, the Loc-FACMAC utilizes multiple mixers to compute the local joint-action value in each partition. Both critic and actor can leverage the information of locality to update the network with an accurate policy gradient value.

Contributions. We summarize our main contributions as follows.

- We propose a novel MARL approach to utilize the inherent locality structure in the multi-agent problem and propose a state-of-art cooperative MARL framework called Loc-FACMAC.
- We test the proposed Loc-FACMAC algorithm on three MARL environments: multi-cartpole, bounded-cooperative-navigation, and SMAC (Samvelyan et al., 2019). Our framework demonstrates

good performance in solving these tasks and is competitive relative to baseline methods. If a proper dependency graph is defined, our framework can achieve the maximum reward with a competitively short amount of training time.

2 Problem Formulation

Many decentralized multi-agent system considers a group of agents works cooperatively to achieve a common goal. We model the problem as a decentralized, partially observable Markov decision process (Dec-POMDP). We define the process as a tuple $\mathcal{M} = (N, \mathcal{S}, \mathcal{A}, P, r, \Omega, O, \gamma, \mathcal{G})$. Here, N denotes a finite set of agents, $s \in \mathcal{S}$ is the true joint state of the environment, and $\mathcal{A}$ is the joint-action space of all agents. At each time step, if the environment is in state $s \in \mathcal{S}$, and each agent i selects an action from a continuous or discrete action space $\mathcal{A}_i$, then the environment would transition to state $s' \sim P(\cdot|s, a)$, where $P(\cdot|s, a)$ represents the transition kernel from state s to s'. We define r as the global reward depending on the global state and the joint action. The discount factor is denoted by γ. We note that Ω is the observation space, which implies that at each time step, agent i can observe partial information o_i sampled from $O_i(s, a) \in \Omega$.

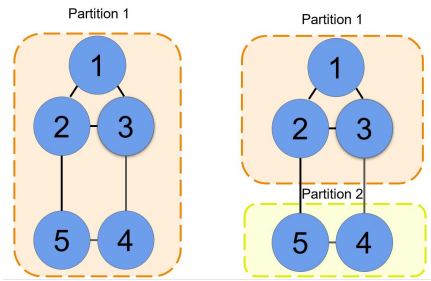


Figure 1: This figure considers a five-agent network and shows how different agents are connected. It also shows two ways of partitioning network to leverage locality in the learning process. Node 4 and Node 5 are away from node 1, node 2, and node 3. Node 4 and node 5 can be grouped separately.

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph of agents, namely a dependency graph, where $\mathcal{V} = \{1, 2, ..., N\}$ denotes the node and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge between nodes. The dependency graph is an additional feature added to the original Dec-POMDP in this work. We assume that the dependency graph can be decomposed into a partition $\mathcal{P} = \{J_k\}_{k=1}^{K}$, such that $J_k \cap J_l = \emptyset, \forall k \neq l$ and $\bigcup_k J_k = \mathcal{G}$. The global reward r is expressed by $\{r_1, r_2, ..., r_K\}$, such that $r = \sum_{k=1}^{K} r_k$. The two agents are correlated if an agent is linked with another agent in the dependency graph, as shown in Fig. 1. Hence, the action of an agent affects its neighbor agent's reward. The goal here is to learn a stochastic policy $\pi_i(a_i|\tau_i)$ or a deterministic policy $\mu_i(\tau_i)$ for each agent i where τ_i is the local action-observation history $\tau_i \in \mathcal{T} = \Omega \times \mathcal{A}$.

We also define the global action value function as

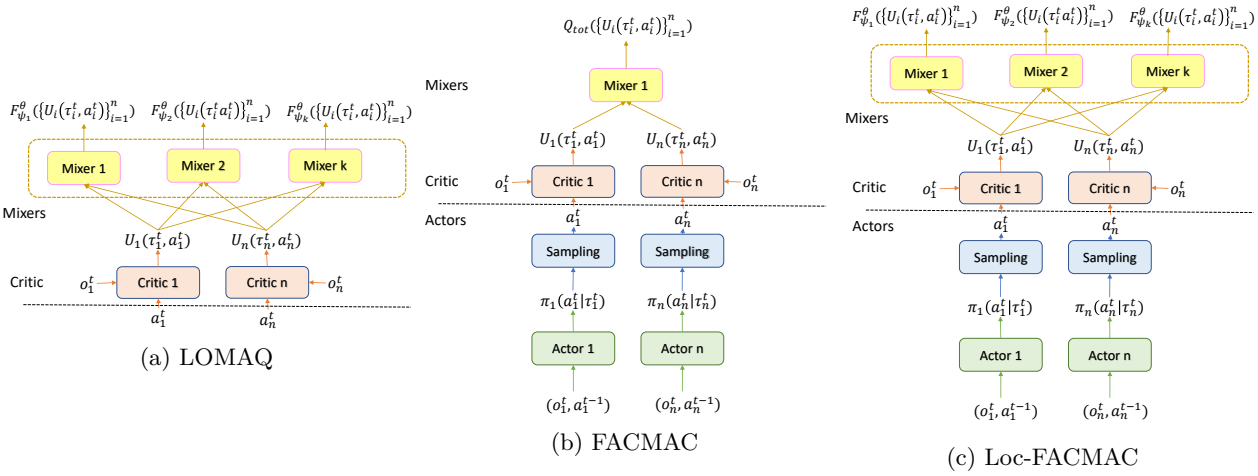
$$Q_{tot}(\boldsymbol{\tau}(t), \boldsymbol{a}(t)) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{\tau}(t), \boldsymbol{a}(t))\right], \tag{1}$$

where we have $\boldsymbol{\tau}(0) = \boldsymbol{\tau}(t), \boldsymbol{a}(0) = \boldsymbol{a}(t)$. Similarly, we define the local Q function $Q_i(\tau_i(t), a_i(t)) = \mathbb{E}[\sum_{l=0}^{\infty} \gamma^t r_i(\tau_i(t), a_i(t))]$, with $\tau_i(0) = \tau_i(t), a_i(0) = a_i(t)$. We assume the condition $Q_{tot}(s(t), a(t)) = \sum_{i=1}^{N} Q_i(s_i(t), a_i(t))$ holds at any time step which is well-defined in value function based approaches (Rashid et al., 2020; Zohar et al., 2022).

3 Proposed Algorithm: Loc-FACMAC

In this section, we present the framework of our new algorithm, Loc-FACMAC, which is a multi-mixer actor-critic method that allows the agents to synchronize the policy update while the locality information of the rewards is retained.

Loc-FACMAC is built upon FACMAC and resolves its limitation of overgeneralized policy gradient. In FACMAC, the agents' network is updated using the policy gradient computed from the global reward. Even though the agent does not contribute to the global reward, it still updates its network using the same policy gradient. Using the distorted policy gradient, agents fail to estimate the action value and fall into a



(a) LOMAQ

(b) FACMAC

(c) Loc-FACMAC

Figure 2: This figure presents the architecture of the proposed LOMAQ, FACMAC, and Loc-FACMAC. Our proposed framework, Loc-FACMAC, consists of Actors, Critics, and Mixers; Actors take local observation and local action history to compute the conditional policy and sample the action from the conditional policy. Critics evaluate the quality of the action. Mixers correlate the local action value and the joint-action value.

Algorithm 1 Proposed Loc-FACMAC Algorithm

Ensure: $i \in N = 1, 2, ..., n$ are the agents
Partition $\mathcal{P}$ of V
Policy $\pi^i_{\theta_i}(\tau_i)$ with parameter θ_i, where $\tau_i \in \Gamma \equiv (\omega \times A)$ is local action-observation history
Local state $s^i_0 \in \Omega$ drawn from $O(s, i)$, where s is true state of environment
Critic Networks $\phi_i \in \mathbb{R}^d$
Mixing networks $\psi_k \in \mathbb{R}$, where $k = 1, 2, .., K$
for iter $t = 1, 2, ..., m$ **do**
  for agent $i = 1, 2, ..., n$ **do**
    Sample action $a_i(t)$ from $\pi^i_{\theta_i}(\tau_i)$ and retrieve next observation $s(t+1)$ and reward $r(t)$
  end for
  We have $Q^\pi_J(\boldsymbol{\tau}^t, \boldsymbol{a}^t, \boldsymbol{s}^t; \boldsymbol{\phi}^t_N, \boldsymbol{\psi}^t_J) = F^t_{\psi_J}(s_t, \{U^{\pi_i}_i(\tau^t_i, a^t_i; \phi^t_i)\}_{i \in \mathcal{N}(\mathcal{J})})$, $\mathcal{N}(\mathcal{J})$ is the κ-hop neighbourhood of J in the dependency graph

  Train mixing and Critic network: update mixing network by minimizing the loss:
  $L(\boldsymbol{\phi}, \boldsymbol{\psi}) = E_D[\sum_{J \in \mathcal{P}}(y^{tot}_J - Q^\pi_J(\boldsymbol{\tau}^t, \boldsymbol{a}^t, \boldsymbol{s}^t; \boldsymbol{\phi}^t_N, \boldsymbol{\psi}^t_J))^2]$
  $\phi_i \leftarrow \phi_i - \alpha \nabla_{\phi_i} L(\boldsymbol{\phi}, \boldsymbol{\psi})$
  $\psi_k \leftarrow \psi_k - \beta \nabla_{\psi_k} L(\boldsymbol{\phi}, \boldsymbol{\psi})$

  Train Actor network: by update
  $\nabla_\theta J(\theta) = E_D[\nabla_\theta \pi \nabla_\pi Q^\pi_{tot}(\tau, \pi_1(\tau_1), ...\pi_n(\tau_n))]$
  Update policy parameter $\theta_k \leftarrow \theta_k - \gamma \nabla_\theta J(\theta)$
end for

sub-optimal policy. Therefore, we adopt the idea of the locality of rewards. The policy gradient is computed from the partitions' reward and used to update the strongly related agents' network. Compared to the policy gradient in FACMAC, the policy gradient computed from partitions' reward is more accurate, so it can give a better update direction to the policy network.

In the Loc-FACMAC, multiple mixers are used to estimate the locality of rewards. Each mixer corresponds to one partition. It takes the local action values and maps them to the partition's joint-action value. By increasing the number of mixers, detail information of the locality of rewards can be retained. A simple

example is guessing the four natural numbers (n1,n2,n3,n4) by giving the sum of the four numbers. If only the total sum is given, the number of possible combinations could be very large. However, if the subset sums are given, i.e. n1+n2 and n3+n4, the number of possible combinations is reduced. Therefore, the extra information of the locality of rewards can aid agents to find the global optimal policy. Then, the loss can be calculated by summing the error in each partition using,

$$L(\boldsymbol{\phi}, \boldsymbol{\psi}) = E_D[\sum_{J \in \mathcal{P}} (y_J^{tot} - Q_J^{\pi}(\boldsymbol{\tau}^t, \boldsymbol{a}^t, \boldsymbol{s}^t; \boldsymbol{\phi}_N^t, \boldsymbol{\psi}_J^t))^2] \tag{2}$$

where $y_J^{tot} = \sum_{j \in J} r_j + \gamma \max_{a^{t+1}} F_{\psi_J}^{t+1}(s_{t+1}, \{U_i(\tau_i^{t+1}, a_i^{t+1})\}_{i \in \mathcal{N}(\mathcal{J})})$ is the target reward of partition J. $\mathcal{N}(\mathcal{J})$ is the κ-hop neighbourhood of J in the dependency graph.

Loc-FACMAC also inherits the property of FACMAC that it separates the learning process of actor and critic. The framework of Loc-FACMAC consists of two parts: Actor and Mixer. The actor generates an optimal action based on the local observation and historical actions and the mixer criticizes the performance of agents' action and computes the loss of the network. The advantage of separating the learning process is to prevent the actor learning from the overestimated/underestimated temporal difference (TD) error. This is critical in MARL because an agent should learn optimal policy by considering the other agents' chosen action during updating the policy network. However, policy gradient updates the agent's policy network by assuming the other agents' policy fixed. The assumption fails to maintain. Especially, when the number of agents increases, the dynamic of agents' behavior becomes hard to predict. Hence, agents frequently get trapped in a sub-optimal policy using MADDPG or QMIX, since no individual agent can modify its optimal action conditioning on fixing all other agents' action under the sub-optimal policy. This problem is resolved by actor-critic methods in that the mixer first measures the true error of the estimated reward. Then, the actor can be updated by computing the policy gradient from the global joint-action. Since mixer and actors are updated in two independent steps, agents can synchronize the actor update by considering other agents' action. The policy gradient is

$$\nabla_\theta = E_D[\nabla_\theta \pi Q_{tot}^{\pi}(\tau, \pi_1(\tau_1, \theta_1), \pi_2(\tau_2, \theta_2), ..., \pi_n(\tau_n, \theta_n)))] \tag{3}$$

where $\pi = \pi_1(\tau_1, \theta_1), \pi_2(\tau_2, \theta_2), ..., \pi_n(\tau_n, \theta_n))$ is the collection of all agents' current policy. Using (3), the policy update of an agent does not only rely on its local observation and local action. Instead, the policy gradient is computed using the lasted estimated error from the mixer and the global sampled action μ.

The details of the Loc-FACMAC framework are shown in Algorithm 1. Each agent has its individual actor and critic. The actor takes the inputs of current observation o_i^t and previous action μ_i^{t-1} to compute conditional policy π_i. Then, the current action μ_i^t can be sampled from the policy π_i and feeds into the critic. The critic produces the utility function U_i which evaluates actions taken by the actor. In the last stage, there are K mixers. Each mixer takes the outputs of the critic and the global state to find a monotonic function F estimating the total rewards of a partition i. The number of mixers is equal to the number of partitions $\mathcal{P}$. Each partition contains a subgroup of agents and each agent has to belong to one partition, such that $J_i \cap J_j = \emptyset, \forall i, j \in 1, ..., k$.

4 Experiments

In this section, we demonstrate the advantages of our proposed algorithm by comparing it with the other state-of-art MARL algorithms, QMIX (Rashid et al., 2018), FACMAC (Peng et al., 2021), and LOMAQ (Zohar et al., 2022) on three discrete cooperative multi-agent tasks and Coupled-Multi-Cart-Pole, StarCraft II, and Bounded-Cooperative Navigation. In Coupled-Multi-Cart-Pole and Bounded-Cooperative Navigation, our algorithm outperforms the other tested algorithms. The result also reveals that utilizing multiple mixers can enhance the maximum reward while the actor-critic controls the speed of convergence. Besides that, we study the effect of the dependency graph of agents and the cluster of rewards on the agents' overall performance. The following results will show that by delicately clustering the rewards of strongly related agents, the agents effectively learn a better policy.

To construct the dependency graph, we operate under the assumption that agents share a strong relationship when their geometric distance is relatively small. If a strong relationship exists, it follows that a corresponding edge connecting the agents must be present in the dependency graph.

4.1 Coupled-Multi-Cart-Pole

The Coupled-Multi-Cart-Pole problem described in Fig. 3 is developed by Zohar et al. (Zohar et al., 2022) based on the standard Cart-Pole problem. The objective of the Coupled-Multi-Cart-Pole problem is to hold the pole in an upright direction to receive positive rewards. The Coupled-Multi-Cart-Pole problem also restricts the cart movement by linking the cart to its neighboring carts with a spring. Hence, when a cart moves in one direction, its force also impacts the motion of its neighboring carts along the connected spring. Since the cart only impacts its neighboring carts, the relationship can be simply modeled as a linear graph. In the following, we show the simulation result of n=6 carts cooperatively holding the pole for 300 steps, so the ideal maximum reward is 1800. We apply our approach Loc-FACMAC to compete with QMIX, FACMAC, and LOMAQ. For Loc-FACMAC and LOMAQ, we partition the rewards into six, $P_i = i, i = 1, 2, ..., 6$ which is the maximum partition.

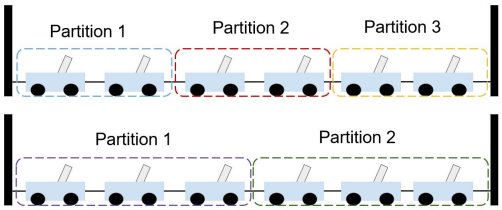
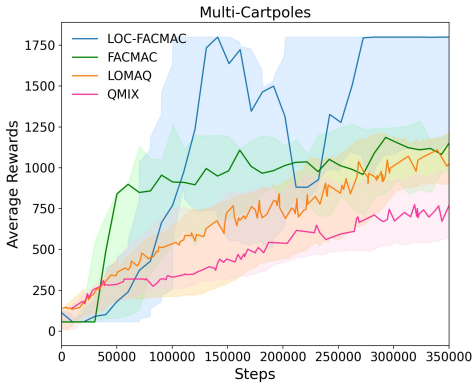


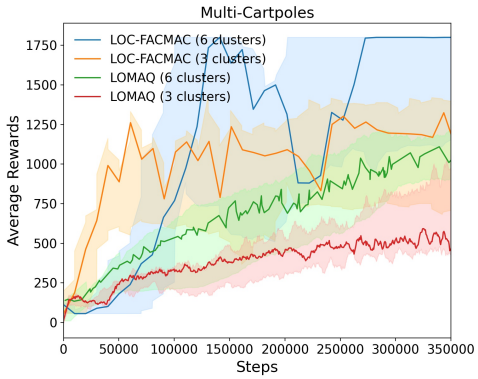
Figure 3: This figure describes the partition for one specific environment of multi-cartpole. We can divide six carts into two possible partitions (2-2-2 and 3-3). There exist other possible partitions, such as 1-2-3.

In this experiment, Loc-FACMAC leverages the locality information and performs the best among all tested frameworks up to 45%. In Fig. 4a, Loc-FACMAC quickly reaches the maximum possible reward at around 140,000 steps. It takes another 150,000 steps to stabilize the rewards. Loc-FACMAC's outstanding performance is due to the integration of the merits of actor-critic and the multiple mixers. The actor-critic method can synchronize the update of agents' policy which guarantees the consistency of agents' gradient and convergence speed. Adopting this approach, FACMAC converges (at around 250,000) much fast than LOMAQ and Qmix. Loc-FACMAC and FACMAC have a very similar reward pattern in that they both have a high initial momentum boosting the speed of convergence of the two approaches. Meanwhile, the weakness of FACMAC is obvious that FACMAC using a single mixer to coordinate agents overgeneralizes loss between the target global reward and the estimated global reward. The limitation is reflected in the maximum reward of FACMAC which is bounded above by 1250. On the other side, LOMAQ can sustainably improve the policy because of using multiple mixers to capture the detailed locality information. Using multiple mixers can better evaluate the performance of each cluster and hence, a higher reward can be reached. The performance of LOMAQ matches the expectation. LOMAQ has a relatively slower initial value compared to FACMAC and Loc-FACMAC, but it improves the reward sustainably over time and reaches approximately the same reward as FACMAC at 350,000. Loc-FACMAC combines the actor-critic method and multiple mixers so it is the fastest method achieving the highest possible reward among all tested methods and its reward is almost 3 times the fundamental method Qmix at 350,000 steps.

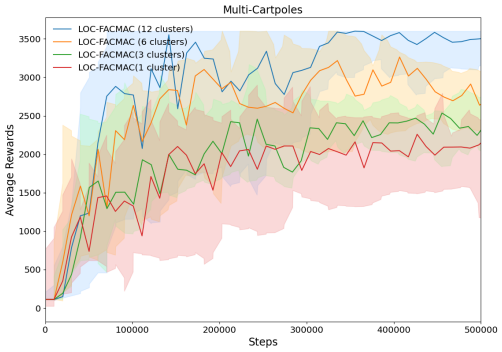
In the aspect of performance, maximizing the number of partitions and the number of mixer inputs can reach the highest performance. However, the trade-off is increasing the computational time. In some machines with limited computational resources, the highest performance may not be their first concern. Therefore, we are also interested in understanding how the size of the partition affects the performance of reinforcement learning agents. In Fig. 4b, we tested LOMAQ and Loc-FACMAC with two different partitions: (1) 6 partitions, {{1},{2},...,{6}} and (2) 3 partitions, {{1,2},{3,4},{5,6}}. For the consistency of the experiment, the inputs of the mixers are always the same in which all agents are taken as the input. The result matches our expectation that if the number of partitions increases, the higher performance can get. The result of Loc-FACMAC using 6 partitions is overwhelming which reaches 1800 at around 140,000. Meanwhile, Loc-FACMAC using 3 partitions performs relatively well compared to LOMAQ using 6 portions and 3 partitions. It obtains roughly 1200 rewards at 350,000 which is slightly better than the number of rewards (1000) of LOMAQ using 6 partitions. Furthermore, in our experimentation, we conducted tests using the LOC-FACMAC algorithm



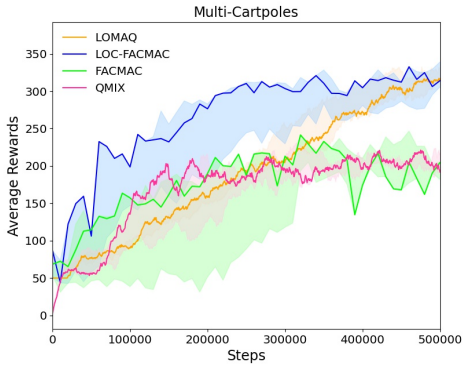
(a) The simulation in Multi-Cartpole environment using the different frameworks.



(b) The simulation in Multi-Cartpole environment comparing the Loc-FACMAC and LOMAQ with different partition size.



(c) he simulation compares different partitions for one specific environment of multi-cart pole using Loc-FACMAC.



(d) The simulation in Bounded-Cooperative-Navigation environment using the different frameworks.

Figure 4: This figure compares the performance of proposed loc-FACMAC algorithm against the baseline algorithms QMIX, LOMAQ, and FACMAC.

in the environment with 12 agents. The results indicated that increasing the number of partitions in the environment had a positive impact on learning speed.

4.2 Bounded-Cooperative-Navigation Environment

The Bounded-Cooperative Navigation task restricts the location of n agents in n bounded regions. The agents can move freely inside their pre-defined region. The bounded regions may have overlapped areas that multiple agents can access the area. The goal of the agents is to cooperatively cover all n landmarks. In this task, the agent's behavior will impact the agents sharing the bounded region. Therefore, if any two agents have a common bounded area, they are connected in the dependency graph. We applied Loc-FACMAC to nine agents Bounded-Cooperative Navigation task. The number of partitions is 9, $P_i = i, i = 1, 2, ..., 9$.

The result in Fig. 4d shows that Loc-FACMAC significantly improves the speed of convergence in the Bounded-Cooperative-Navigation problem. In this problem, the structure of locality dominates the quality of agents' action. An agent's local reward highly depends on whether its' neighboring agent has already covered a landmark in the common area. Agents can only receive local rewards if its covered landmark is available. Therefore, using only one mixer, the agent is nearly impossible to comprehend its contribution to the local reward because one mixer is not capable to estimate an accurate local reward from computing

the global reward. Both Qmix and FACMAC use one mixer so their performance is no different in this task. They both reach approximately 200 rewards at 500,000 steps. In contrast, LOMAQ is able to reach a higher global reward which obtains 316 rewards at 500,000 steps. It uses multiple mixers to capture the detailed information of the locality so each agent estimates its action value from the perspective of local agents and global coordination. Similarly, Loc-FACMAC can achieve 300 rewards with a higher speed of convergence. It is due to Loc-FACMAC also utilizes the actor-critic to coordinate the policy update. Hence, it takes much less time to explore the optimal policy for this task.

4.3 Experiment in Locality Structure

In this section, we want to justify the importance of obtaining a proper locality structure for using Loc-FACMAC. We experiment with Loc-FACMAC on a super hard map MMM2. MMM2 consists of 10 controllable allies and 12 enemies. The allies can observe the state of agents within their visible range and decide their next action based on the learned policy. The goal of the task is to defeat all the enemies in a given time. The dependency graph is determined by the position. Each ally is considered as a node and it is connected with the four closest allies. The number of partitions is 10, $P_i = i, i = 1, 2, ..., 10$.

In Fig. 5, the experiment result does not show a significant improvement using Loc-FACMAC. Both Loc-FACMAC, FACMAC, and LOMAQ have nearly perfect win rates. We claim that Loc-FACMAC cannot improve the performance because the dynamic of SMAC is more complicated and we lack an understanding of the relationship between the agents.

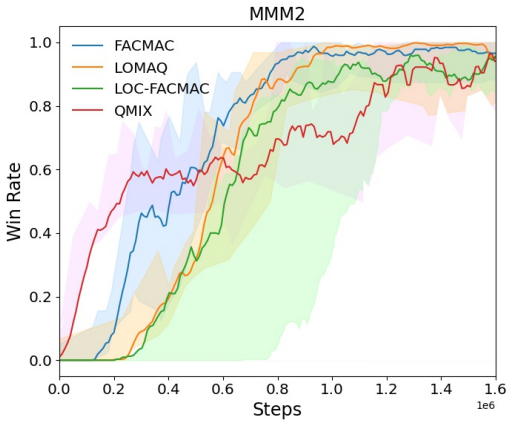


Figure 5: The simulation in SMAC MMM2 environment using the different frameworks.

5 Limitation

This paper is based on two important assumptions: Individual-Global-Max (IGM) and agents' dependency. The first assumption assumes that each agent maximizing its Q value reward always benefits the team as a whole. However, this assumption may not always hold. In scenarios where resources are limited, agents may exhibit competitive behavior, where their local objectives are against the team's objectives. This becomes a challenge for the mixer to find an appropriate monotonic function for the problem. Therefore, it would be interesting to explore the problem with constraints and examine how it affects the MARL approach.

The second assumption is that the dependency among agents is known and fixed. However, in dynamic environments, the dependency between agents can change over time, making it difficult to accurately model the dependency graph. Failure to properly model the dependency graph can lead to degraded performance of locality-based MARL approaches like SMAC. As a result, there is a growing interest in developing methods to model dependency graphs without prior knowledge or assumptions.

6 Conclusions

In this paper, we propose a state-of-art MARL method, Loc-FACMAC. This method combines the idea of the locality of rewards and the actor-critic method to improve agents' performance and learning efficiency. We also verify the advantage of using Loc-FACMAC in three different tasks. In the Coupled-Multi-Cart-Pole and Bounded-Cooperative Navigation, Loc-FACMAC outperforms all other existing methods. The result also reveals the correlation between framework structure, the maximum reward, and the speed of convergence. In

future works, we will address the issue that the construction of a dependency graph requires prior knowledge and the scenario of IGM does not hold.

References

Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients, 2017.

Xiaofan He, Huaiyu Dai, and Peng Ning. Faster learning and adaptation in security games by exploiting information asymmetry. *IEEE Transactions on Signal Processing*, 64(13):3429–3443, 2016.

Tianchen Ji, Roy Dong, and Katherine Driggs-Campbell. Traversing supervisor problem: An approximately optimal approach to multi-robot assistance. *arXiv preprint arXiv:2205.01768*, 2022.

Qize Jiang, Minhao Qin, Shengmin Shi, Weiwei Sun, and Baihua Zheng. Multi-agent reinforcement learning for traffic signal control through universal communication method. *arXiv preprint arXiv:2204.12190*, 2022.

Andrés C Jiménez, Vicente García-Díaz, and Sandro Bolaños. A decentralized framework for multi-agent robotic systems. *Sensors*, 18(2):417, 2018.

Andreas Kolling and Stefano Carpin. Multi-robot surveillance: an improved algorithm for the graph-clear problem. In *2008 IEEE International Conference on Robotics and Automation*, pp. 2360–2365. IEEE, 2008.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

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

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

Jun Ota. Multi-agent robot systems as distributed autonomous systems. *Advanced engineering informatics*, 20(1):59–70, 2006.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12208–12221. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/65b9eea6e1cc6bb9f0cd2a47751a186f-Paper.pdf.

Han Qie, Dianxi Shi, Tianlong Shen, Xinhai Xu, Yuan Li, and Liujing Wang. Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE access*, 7: 146264–146272, 2019.

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 Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4295–4304. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/rashid18a.html.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Sergey Vorotnikov, Konstantin Ermishin, Anaid Nazarova, and Arkady Yuschenko. Multi-agent robotic systems in collaborative robotics. In *Interactive Collaborative Robotics: Third International Conference, ICR 2018, Leipzig, Germany, September 18–22, 2018, Proceedings 3*, pp. 270–279. Springer, 2018.

Rose E Wang, Michael Everett, and Jonathan P How. R-maddpg for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Zhiwei Xu, Dapeng Li, Yunpeng Bai, and Guoliang Fan. Mmd-mix: Value function factorisation with maximum mean discrepancy for cooperative multi-agent reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2021. doi: 10.1109/IJCNN52387.2021.9533636.

Tianyu Zhang, Andrew Williams, Soham Phade, Sunil Srinivasa, Yang Zhang, Prateek Gupta, Yoshua Bengio, and Stephan Zheng. Ai for global climate cooperation: Modeling global climate negotiations, agreements, and long-term cooperation in rice-n. *arXiv preprint arXiv:2208.07004*, 2022.

Roy Zohar, Shie Mannor, Guy Tennenholtz, , and and. Locality matters: A scalable value decomposition approach for cooperative multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9278–9285, 2022.

Appendix

A Background

QMIX (Rashid et al., 2018) is a recent framework adopting Centralised Training with Decentralised Execution (CTDE) to solve cooperative MARL tasks. The framework consists of individual agent networks representing the action value functions U_i and a mixing network that combines the action values of all agents to obtain the global joint-action value Q_{tot}. Note that each U_i is technically a utility function as they do not directly estimate the discounted expected return. In the training stage, the mixing network can access global state information and the global joint-action value. The purpose of the mixing network is to eliminate the difference between Q_{tot} and the function of the action value set $(U_1, U_2, ..., U_i)$, such that

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{a}) = f_\psi(\boldsymbol{\tau}, U_1(\tau_1, a_1), U_2(\tau_2, a_2), U_N(\tau_N, a_N)),$$

where τ_i is the local action-observation history of agent i and a_i is the local action specific to agent i. Here, f_ψ is required to be a continuous monotonic function to ensure the consistency between the global optimal action and the agents' optimal action. The loss function of QMIX is given by

$$L(\theta, \psi) = \sum_{i=1}^{b} (y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \mathbf{a}; \theta, \psi))^2, \tag{4}$$

where b is the batch size of transitions sampled from the replay buffer and $y_i^{tot} = r + \gamma \max_{\boldsymbol{a}'} Q_{tot}(\boldsymbol{\tau}', \mathbf{a}'; \theta^-, \psi^-))$ is the target value of Q_{tot}. Here, θ and ψ are the parameters of the agent's network and the mixing network, respectively. Through back-propagation, θ and ψ are adjusted to minimize the loss. Accordingly, the learned action value function U_i can properly evaluate the quality of the action and it only depends on the local observation and the parameter θ_i. In the execution stage, each agent only maximizes its local action value using the local observation.

LOMAQ (Zohar et al., 2022) extends the idea of QMIX and improves the algorithm efficiency using the locality of rewards. The core idea is that the agents can be partitioned into different K clusters $\{J_k\}_{k=1}^K$ which defines a partition $\mathcal{P}$, such that $J_k \cap J_l = \emptyset, \forall k \neq l$ and $\bigcup_k J_k = \mathcal{G}$. LOMAQ proved that under the Q-Summation Maximisation (QSM) Condition given by:

$$\max_{\mathbf{a}} \sum_{i=1}^{n} Q_i^\pi(\boldsymbol{\tau}, \mathbf{a}) = \sum_{J \in \mathcal{P}} \max_{\mathbf{a}} \sum_{i \in J}^{n} Q_i^\pi(\boldsymbol{\tau}, \mathbf{a}), \tag{5}$$

we can exploit the locality of rewards to reduce the regret of decisions. LOMAQ utilizes K mixers where the value of K can be defined manually to leverage the property of reward locality. Each mixer corresponds to the joint-action value function of one partition J so that the mixer can map the local action values to the joint-action value of the partitions. The loss function of LOMAQ is

$$L_F(\theta, \psi) = \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{s}'} \left[\sum_{J \in \mathcal{P}} \left(y_J - F_J^\psi(U_{i \in J}^\theta(\tau_i, a_i)) \right)^2 \right], \tag{6}$$

where $y_J = \sum_{j \in J} r_j + \gamma \max_{a'} F_J^\psi(U_{i \in J}^\theta(\tau_i', a_i'))$ is the target reward of partition J, and F_J is the joint-action value function of J.

FACMAC (Peng et al., 2021) is a recent state-of-art framework proposed by Peng et al. (2021). This approach resolves the limitation of biased updates in QMIX by utilizing a factorized critic in an actor-critic framework. Each agent has a critic and an actor. The actor takes the local observation $o_i(t)$ and the past action $a_i(t-1)$ as the input to compute the local policy π_i. Then, the action $a_i(t)$ is sampled from π_i. The function of the critic is to evaluate the local action value U_i from $a_i(t)$ and $\tau_i(t)$. Similar to the structure of QMIX, FACMAC contains a mixer responding to coordinate all agents' action by mapping U_i to the joint-action value Q_{tot}. Hence, the loss function of the joint-action value function is the same as the loss function in QMIX. Agents' action value functions are updated using the policy gradient given by:

$$\nabla_\theta = \mathbb{E}_{\mathcal{D}}[\nabla_\theta \boldsymbol{\mu} Q_{tot}^\mu(\boldsymbol{\tau}, \mu_1(\tau_1, a_1), \mu_2(\tau_2, a_2), ..., \mu_n(\tau_n, a_n))], \tag{7}$$

where $\mathcal{D}$ is the data buffer and θ is the parameter of actor network.

The back-propagation of FACMAC relies on a single joint-action value Q_{tot}. This means that even if an agent does not contribute to the reward in a particular step, the policy gradient ∇_θ still updates its parameters. On the other hand, LOMAQ does not take into account the impact of changing other agents' parameters when updating its parameters, resulting in a slower convergence time. To address these limitations, we propose a novel approach called LOC-FACMAC, which combines the strengths of both methods.

B Details of the Environments

Multi-Cartpole environment is a variant of Single Cartpole environment. Each agent has to uphold its pole to obtain the reward. Each agent can obtain a +1 reward when its pole does not fall. Multi-Cartople adds a restriction on the Single cart pole that each agent is connected to its neighbor by a spring. Each agent has two inputs that can control its agent to move toward left or right. Its motion affects its neighbor by applying force to the spring. The state of an agent is a four dimensions vector that includes the position, velocity, pole angle, and pole angular velocity. An agent can observe two neighborhood agents' states. The global state is a collection of the state of all agents.

Bounded-Cooperative-Navigation is a task in which each agent has to cover a landmark and avoid collision with their agent in its own common area. Each agent is bounded in a circle region and has some areas overlapping with near agents. The reward of an agent is computed by three factors: occupant reward, bonus reward, and collision reward. The occupant reward is triggered when an agent covers the landmark and another agent does not cover the landmark. The value of the bonus reward is based on the distance between the agent and the landmark to enough agent to move toward the landmark. The collision reward punishes the agent when it collides with another agent. The agent can observe the nearby agents' relative position, the landmark's relative position, and the number of occupant landmarks. The state is the collection of agents' velocity, agents' position, and landmarks' position.

StarCraft Multi-agent Environment. In the Starcraft environment, we run the map MMM2, which is heterogeneous & asymmetric type of mission. It consists of 10 allies which include 1 Medivac, 2 Marauders & 7 Marines. The mission of the 10 allies is to eliminate all 12 enemies (1 Medivac, 3 Marauders & 8 Marines). The observation of an agent has four sections which are agent movement features, enemy features, ally features, and agent unit features. The movement features are the position of it moving to, height information, and pathing grid. The enemy features have information about the enemy in sight range available to attack, the enemy's health, the enemy's relative position, the enemy's shield, and the enemy's unit type. The ally features include visible allies' id, visible allies' relative position, visible allies' shield, and visible allies' unit type. The agent unit features are the health of the agent, shield, and unit type. The state is the collection of ally agents' attributes, enemies' attributes, the last actions of all agents, and time steps. The reward is calculated by the accumulative hit and shield point damage dealt to the enemy and rewards death value per enemy unit killed. The possible actions of the agent are to stop, move in four directions, and attack/heal agents in range.

C Dependency graph

The construction of a dependency graph is important in Loc-FACMAC. If the dependency graph can accurately reflect the relationship of all agents, the network update can target on the most correlated agents and improve the efficiency of the training. The construction of dependency graph requires the prior knowledge of the problem. In the graph, each agent is considered as a node. If the agent i is affected by the agent j, the node i and the node j should be connected.

The partition in each mixer can be determined manually based on the need of the task and the amount of resources. The smaller size of partition can provide more locality information and better performance but it also increases the computational time. Also, according to the dependency graph, the strongly related agents should be grouped into one partition. By grouping strong relationship agents into the same partition,

the locality of the rewards becomes meaningful because it can truly reflect the performance of the related cooperative agents.

D Hyper-parameters

Table 2: Experiment 1:Multi-Cartpole

Hyperparameter	**Values**
ϵ_{start}	1.0
ϵ_{end}	0.05
ϵ anneal time	50000
batch size	128
γ	0.99
grad norm clip	20
actor learning rate	0.0025
mixer and critic learning rate	0.008
td lambda	0.2
target update mode	soft
target update rate	0.5
target update interval	50
Mixer	Linear, Relu, Linear, ReLu, Linear (mixing embed dim 32)
Parameter Sharing in Mixer	No
Critic	Linear, Linear, Linear (embed dim 64)
Actor	Linear, Relu, GRU, Linear (hidden dim 64)

Table 3: Experiment 2: Bounded-Cooperative-Navigation

Hyperparameter	**Values**
ϵ_{start}	1.0
ϵ_{end}	0.05
ϵ anneal time	100000
batch size	128
γ	0.99
grad norm clip	20
actor learning rate	0.0025
mixer and critic learning rate	0.0005
td lambda	0.2
target update mode	soft
target update rate	0.5
target update interval	50
Mixer	Linear, Relu, Linear, ReLu, Linear (mixing embed dim 32)
Parameter Sharing in Mixer	No
Critic	Linear, Linear, Linear (embed dim 64)
Actor	Linear, Relu, GRU, Linear (hidden dim 64)

Table 4: Experiment 3: SMAC

Hyperparameter	**Values**
ϵ_{start}	0.8
ϵ_{end}	0.05
ϵ anneal time	50000
batch size	32
γ	0.99
grad norm clip	20
actor learning rate	0.0025
mixer and critic learning rate	0.0005
td lambda	0.2
target update mode	soft
target update rate	0.5
target update interval	50
Mixer	Linear, Relu, Linear, ReLu, Linear (mixing embed dim 32)
Parameter Sharing in Mixer	No
Critic	Linear, Linear, Linear (embed dim 64)
Actor	Linear, Relu, GRU, Linear (hidden dim 64)