mathematics

MDPI

Article

Noise-Regularized Advantage Value for Multi-Agent Reinforcement Learning

Siying Wang [1], Wenyu Chen [1], Jian Hu [2,*], Siyue Hu [3] and Liwei Huang [1,4]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2] Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei 106, Taiwan
[3] Department of Computer Science & Information Engineering, National Taiwan University, Taipei 106, Taiwan
[4] The State Key Laboratory of IoTSC, University of Macau, Taipa, Macau 999078, China
* Correspondence: r08944053@ntu.edu.tw

Abstract: Leveraging global state information to enhance policy optimization is a common approach in multi-agent reinforcement learning (MARL). Even with the supplement of state information, the agents still suffer from insufficient exploration in the training stage. Moreover, training with batch-sampled examples from the replay buffer will induce the policy overfitting problem, i.e., multi-agent proximal policy optimization (MAPPO) may not perform as good as independent PPO (IPPO) even with additional information in the centralized critic. In this paper, we propose a novel noise-injection method to regularize the policies of agents and mitigate the overfitting issue. We analyze the cause of policy overfitting in actor–critic MARL, and design two specific patterns of noise injection applied to the advantage function with random Gaussian noise to stabilize the training and enhance the performance. The experimental results on the Matrix Game and StarCraft II show the higher training efficiency and superior performance of our method, and the ablation studies indicate our method will keep higher entropy of agents' policies during training, which leads to more exploration.

Keywords: multi-agent reinforcement learning; proximal policy optimization; exploration; noise injection; advantage function

MSC: 68T07; 68T42; 68T99

Citation: Wang, S.; Chen, W.; Hu, J.; Hu, S.; Huang, L. Noise-Regularized Advantage Value for Multi-Agent Reinforcement Learning. *Mathematics* **2022**, *10*, 2728. https://doi.org/10.3390/math10152728

Academic Editor: Zhaobin Wang

Received: 21 June 2022
Accepted: 24 July 2022
Published: 2 August 2022

1. Introduction

Multi-agent reinforcement learning (MARL) has seen revolutionary breakthroughs with its successful applications to multi-agent cooperative tasks, such as robot swarms control [1], coordination of robots [2], autonomous vehicle coordination [3], computer games [4], and multi-agent differential games [5–7]. Since agents need to make a series of proper decisions to cooperate better to complete the tasks, many researchers have extended the deep reinforcement learning methods to the multi-agent field to solve these problems. An intuitive approach is to treat each agent as a separate individual and train the decentralized policies of agents with deep Q-Learning [8], i.e., independent Q learning (IQL) [9]; however, IQL cannot address the non-stationarity introduced by the fast-changing policies of agents during training; these policies could not even converge with enough exploration and local observations. Agents need additional information about each other to be aware of the others' policy changes; therefore, the *centralized training and decentralized execution (CTDE)* paradigm [10], which allows agents to access global information during training and execute based only on their local histories, is proposed to alleviate the non-stationary problem and stabilize training in some multi-agent cooperation scenarios.

Many CTDE algorithms, such as MADDPG [11], MAAC [12], and QMIX [13], have been proposed for different multi-agent tasks. These algorithms enable agents to treat the concatenated local observations as global state information, merge them with attention

mechanism, or incorporate them into the weights and bias of the delicately designed neural networks [14]. All these methods aim to enhance the observational representation of the individual agent by fusing more global information. Through the above methods, agents can integrate more information about policies and local observations from each other, so that they can make more appropriate decisions to cooperate better; however, the size of the joint action space increases with the number of agents. Even with the help of more accurate state information, it is extremely difficult to directly search for the optimal joint policy in a huge state-action space with multiple agents. This may lead us to take millions of samples to train the policies.

On the other hand, there are some ways to ensure that the training of agents progresses steadily from the perspective of policy updating. Considering the excellent properties of monotonically improving the policy, IPPO extends PPO [15] to the multi-agent setting to train the agents efficiently. In the case of the maximum entropy optimization target, IPPO ensures that the policy of each agent can be monotonically learned to find the optimal joint policy; however, MAPPO [16], which incorporates global information with IPPO, may not perform as well as IPPO. Although the global information can enrich the representation of observations, it also brings information redundancy to the centralized critic. This counter-intuitive phenomenon is described as *policy overfitting*, which would mislead the update of policy in the wrong direction with all agents sharing the parameters of networks. The inherent defect of *policy overfitting* raised by the centralized critic is difficult to improve by modifying the traditional Markov decision process (MDP) or changing the exploration method of policy networks. It also makes it harder to train the agents in an actor–critic architecture since there are policy and value networks that need to be trained.

To deal with the *policy overfitting* and reducing the information redundancy brought by the centralized critic, we propose a novel noise injection method to regularize the policies of actor–critic MARL algorithms. We focus on MAPPO, develop two patterns of the noise injection method applied to the advantage function, which is, respectively, inspired by the noisy net [17] and parameter space noise [18]. Different from replacing the exploration mechanism with a noisy policy network, as in [17,18], we inject the noise directly into the centralized value network to enrich the representation ability. Furthermore, we theoretically analyze the reason for *policy overfitting* in multi-agent actor–critic, and explain that the problem comes from updating the agents' policy by batch-sampling updating. Our experimental results on the Matrix Game and challenging StarCraft II micromanagement benchmark tasks (SMAC) [4] show that the injected noise can augment the variance of the centralized value function, and indirectly increase the entropy of agents' policies to obtain more exploration during training. In general, our main contributions are summarized in the following:

- We analyze the reason for *policy overfitting* in actor–critic MARL algorithms with the centralized value function, which is caused by the batch sampling mechanism in the training stage;
- We propose two patterns of noise injection to deal with the *policy overfitting* problem, and experimentally prove the noise injected into the centralized value function can maintain the entropy of agents' policies during training to alleviate the information redundancy and enhance performance;
- The experiments show our proposed method is able to archive comparable or much better performance than *state-of-the-art* results in most hard scenarios of SMAC compared to the current most trustworthy actor–critic MARL methods. Our code is open source and can be found at https://github.com/hijkzzz/noisy-mappo (accessd on 7 January 2022) for experimental verification and for future works.

2. Related Works

Multi-agent reinforcement learning approaches are mainly divided into four categories: analysis of emergent behaviors, learning communications, learning cooperation, and agent modeling agents [19]. Among them, the agent cooperation is the top priority in this field.

Recently, many MARL algorithms under the CTDE paradigm have been proposed to alleviate the non-stationarity during training. The straightforward idea is to consider all the agents in the same environment as a whole when we obtain all local observations and additional state information. This concept brings out plenty of algorithms under the CTDE paradigm. Value decomposition networks (VDN) [20] is the first attempt to factorize the joint critic into each individual agent. QMIX [13] takes a step forward and factorizes the centralized critic function by ensuring the consistency of *argmax* operator between Q_{tot} and each Q_i, which is able to effectively reduce the search space of the joint policy. QTRAN [21] learns the discrepancy between $\sum_{i=0}^{T} Q_i(o_i, a_i)$ and $Q_{tot}(\mathbf{o}, \mathbf{a})$ and compensates for this discrepancy through a state-based value function, which would elaborately factorize the centralized critic function and train all the agents in an end-to-end fashion. Then, QPLEX [22] factorizes the centralized critic into advantage value for each agent with transformed dueling architecture, and LICA [23] extends QMIX to continuous action spaces with the entropy of joint policy to constrain the training of agents.

On the other hand, actor–critic style algorithms are also shining in MARL. MADDPG [11], as a representative, extends the DDPG [24] to a multi-agent setting and trains agents with a centralized value function. MAAC [12] enriches the information of the centralized function in MADDPG with a self-attention mechanism [25]. COMA [26] trains agents with a centralized critic with counterfactual advantages, which is the embodiment of the credit assignment from the perspective of the expectation of a value function. FACMAC [27] combines the consistent constraint of QMIX with actor–critic algorithm for agents to improve the training efficiency of agents. Furthermore, researchers introduced independent PPO [16] to the multi-agent domain to monotonically update the policies of agents. They also equip the global state information with IPPO and propose MAPPO [16]. Yu [28] feeds agent-specific features to the centralized critic network of MAPPO, and proposes an information filtering method to mask the useless features, which would significantly improve MAPPO's performance in some cooperative scenarios. Still, Kuba [29] theoretically analyzes the information redundancy of centralized critic in MAPPO/MATRPO, then indicates the parameter sharing between agents will mislead the update of policies and deteriorate the performance.

Moreover, agents would also suffer from inefficient exploration in MARL setting even with enough state information. MAVEN [30] equips the value-based methods with *committed exploration* to persist the joint exploratory policy over an entire episode. ROMA [31] introduces a role concept-based regularizer to train agents more efficiently. Pan [32] proposes to use a synthetical *softmax* operator to update the Q-function under the CTDE paradigm, and [33] encourages the agent to maintain a common goal while all agents are exploring. Moreover, the traditional exploration, such as ϵ-greedy method, can be incorporated into a noise-injection network [17,18]. All the noises are injected into the parameters of policy network since designers expect to disturb the actions to explore other states, then make the joint value function jump out of the local optimal region.

All the methods above expect to efficiently train the policies with adequate information both from agents and environments, or other novel ways to increase exploration; however, these methods do not solve the information redundancy problem brought by centralizing the critics of agents together. Inspired by the noisy network [17], we explore the effect of the specific pattern of Gaussian noise directly injected in centralized advantage function of agents, which will correct the direction of policy update and regularize the training.

3. Preliminaries

Dec-POMDP We consider a cooperative task, which can be described as a decentralized partially observable Markov decision process (Dec-POMDP) [34], which is formally defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{R}, \mathcal{P}, n, \gamma)$. $\mathcal{S}$ represents the global state space, and $\tau_i = \mathcal{O}(s; i)$ is partially observation for each agent i at state s. $\mathcal{P}(s'|s, \mathbf{a})$ is the state transition probability in the environment given the joint action $\mathbf{a} = (a^1, \ldots, a^N) \in \mathcal{A}$. Each agent shares the same reward function $\mathcal{R}(s, \mathbf{a})$, and chooses sequential actions under partial observations.

N denotes the number of agents and $\gamma \in [0,1)$ is the discount factor. The whole team of agents attempt to learn a joint policy $\boldsymbol{\pi} = \langle \pi_1, ..., \pi_N \rangle$ that maximizes their expected discounted return in a complete trajectory as

$$V^{\boldsymbol{\pi}}(s_0) = \mathbb{E}_{a^i \sim \pi^i, t \sim T}\left[\sum_{t=0}^{\infty} \gamma_t r_t\left(s_t, a_t^1, \ldots, a_t^N\right)\right]. \tag{1}$$

Multi-Agent Policy Gradient (MAPG) Policy gradient (PG) is the cornerstone of actor–critic RL algorithms, which makes policy π closer to the actions which contain larger advantage values by gradient ascending. Since we can easily extend PG to multi-agent setting, the policies of N agents are trained with a shared advantage function which is introduced by the global state s as

$$\mathcal{G} = \sum_{i}^{N} \mathbb{E}_{s_t, \boldsymbol{a}_t \sim \boldsymbol{\pi}}\Big[\nabla \log \pi^i(a_t^i \mid \tau_t^i) \boldsymbol{A}^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t)\Big], \tag{2}$$

where $\boldsymbol{A}^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \boldsymbol{r}_t + \gamma \boldsymbol{V}^{\boldsymbol{\pi}}(s_{t+1}) - \boldsymbol{V}^{\boldsymbol{\pi}}(s_t)$ is estimated by a centralized value function with access to the global state s during training, and $\gamma \in [0,1]$ is the discount factor. The joint actions conform to the distribution of joint policies and the objective function is related to the gradient update direction fo all agents.

Multi-Agent Proximal Policy Optimization (MAPPO) Though it is easy to directly apply PPO to each agent in cooperative scenarios, the independent PPO [16] may also encounter non-stationarity since the policies of agents are updated simultaneously. MAPPO extends IPPO's independent critics to a centralized function with additional global information, and the learning target is derived as

$$J^{\text{mappo}} = \frac{1}{N} \sum_{i}^{N} \mathbb{E}_{\boldsymbol{\pi}_{\text{old}}}\Big[\min\Big(\rho^i \boldsymbol{A}^{\text{old}}, \mathrm{C}(\rho^i, \epsilon) \boldsymbol{A}^{\text{old}}\Big)\Big], \tag{3}$$

where $\rho^i = \frac{\pi^i(a^i|\tau^i)}{\pi_{\text{old}}^i(a^i|\tau^i)}$ is the importance sampling weight for each agent, $\boldsymbol{\pi}^{-i}$ denotes all the policies except for agent i, and $\boldsymbol{A}^{\text{old}}\left(s_t, a_t, \boldsymbol{\pi}_{\text{old}}^{-i}\right) = \boldsymbol{r}_t + \gamma \boldsymbol{V}^{\boldsymbol{\pi}}(s_{t+1}) - \boldsymbol{V}^{\boldsymbol{\pi}}(s_t)$. $\mathrm{C}(\rho^i, \epsilon) = \text{clip}(\rho^i, 1-\epsilon, 1+\epsilon)$ is a range-limiting function, which limits the ratio ρ^i in the interval $(1-\epsilon, 1+\epsilon)$.

Noise-injection Methods for Exploration Alongside the ϵ-greedy mechanism, the parametric noise in the weights of the policy network can also aid efficient exploration for agents. These parameters of noise are learned with gradient descent along with the remaining network weights [17], and induce the stochasticity of agents' policies to explore. The corresponding noisy linear layer is defined as:

$$y \overset{\text{def}}{=} (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b, \tag{4}$$

where the learnable parameters $\mu^w + \sigma^w \odot \varepsilon^w$ and $\mu^b + \sigma^b \odot \varepsilon^b$ replace w and b in the original layer $y = wx + b$, respectively. Meanwhile, the noisy-injection method with non-learnable parameters is also proposed [18]. It is worth noting that these methods aim to directly intervene in the selection of actions at execution, forcing agents to explore other states. Both methods affect the policy rather than critic network, which is different from our goal to alleviate the information redundancy of joint value function.

4. Method

In this section, we first describe and analyze the reason for the *policy overfitting* problem, then we propose the two patterns of noise-injection methods to deal with the information redundancy in the centralized value function.

4.1. Policy Overfitting

Let us start from MAPPO, the typical example of the actor–critic style algorithm, to illustrate the *policy overfitting* problem brought by the centralized value function and batched sampling mechanism. Since our ultimate goal is to update the policy of each agent, we can directly obtain the derivative of the objective function with respect to the policy of agent i. The expected policy gradient of MAPPO from (3) will be simplified as

$$\begin{aligned}\frac{\partial J}{\partial \pi^i(a_t^i \mid s_t)} &\propto \mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\pi}}[\boldsymbol{A}^{\boldsymbol{\pi}}\left(s_t, a_t^i, \boldsymbol{a}^{-i}\right)] \\ &= \mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\pi}}[\boldsymbol{r}(s_t, a_t^i, \boldsymbol{a}^{-i}) + \boldsymbol{V}^{\boldsymbol{\pi}}(s_{t+1}) - \boldsymbol{V}^{\boldsymbol{\pi}}(s_t)],\end{aligned} \tag{5}$$

as this gradient expects the actions of other agents rather than agent i itself; its value can represent the contribution of agent i. This marginal advantage function relies on the actions of all the other agents, which will be poorly estimated with the exploration-biased actions. Since we can access the global state information and global reward from the environment during training, the centralized advantage function can also be calculated in the bootstrap way. Still, similar results are also held in other PG-style MARL algorithms.

Furthermore, it is intractable to sweep the entire joint state-action space to obtain an unbiased gradient. Modern multi-agent algorithms including MAPG and MAPPO, estimate the gradient in (7) with sampled examples. These finite samples will enforce us to obtain a fluctuating gradient factorized by a mean value and deviations. These deviations may lead the policy update of agent i into a sub-optimal direction, preventing the exploration of trajectories with higher returns. Obviously, this inherent defect will persist as long as we use a batched sampling mechanism to update the gradients of policies.

Here, we illustrate it with a simple example. Assume we have just two cooperative agents with the centralized value function, we need to update the policy of both agents with trajectories sampled from the replay buffer or environment. As illustrated in Figure 1, assume this advantage value calculated by the centralized value function is dominated by the action of Agent 2 rather than Agent 1, so the global reward given by the environment is supposed to be assigned to Agent 2; however, the policy gradients with this shared advantage value under CTDE paradigm would improve the policies of both agents, i.e., $\frac{\partial \hat{J}}{\partial \pi^i\left(a_t^i|s_t\right)} \propto \boldsymbol{A}^{\boldsymbol{\pi}}\left(s_t, a^1, a^2\right)$. The additional information brought by the centralized critic would evenly distribute the global reward to each agent, which would assign the wrong credit to Agent 1. This credit misassignment will ultimately cause *policy overfitting* even in the same cooperative scenarios sometimes, which is contrary to the conventional wisdom that additional information would help agents act more carefully and further alleviate the non-stationarity caused by fast-changing policies of other agents. This credit misassignment here is essentially different from that in COMA, we have put the specific differences between them in Appendix A.

4.2. Noisy Advantage Values

Since the advantage values learned with finite samples are usually biased, our core motivation is to smooth these advantage values with the designed noise to prevent the *policy overfitting* and non-stationarity problems in the multi-agent setting. Inspired by noisy net [17] and parameter space noise [18], we propose two patterns of the noisy centralized value for policy regularization to address the aforementioned problems.

Noisy Advantage MAPPO (NA-MAPPO) Since the misguided policy is derived from the shared advantage values, we sample a Gaussian noise for each agent i, and then inject it into the advantage value $\boldsymbol{A}_b$ with a weight α as follows

$$\begin{aligned} x^i &\sim \mathcal{N}(0,1); \quad \forall i \in N, \\ \boldsymbol{A}_b^i &= (1-\alpha) \times \boldsymbol{A}_b + \alpha \times x^i; \forall i \in N, b \in B, \end{aligned} \tag{6}$$

as shown in Figure 2, this sampled noise in (6) aims to perturb the policy update of some agents unrelated to the global rewards and regularize the policy to some extent.

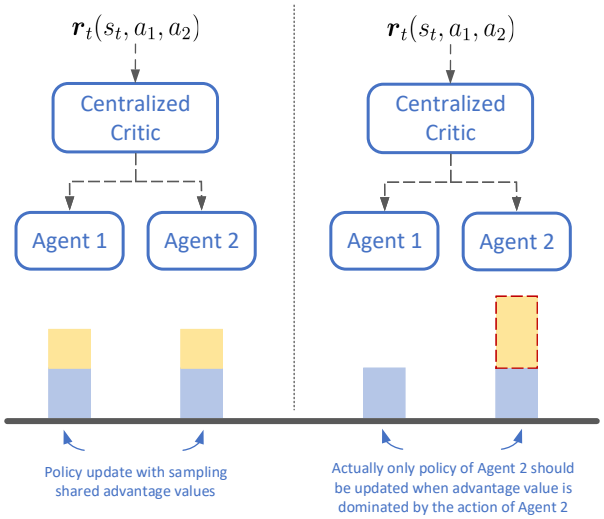


Figure 1. The schematic diagram of two agents with shared advantage function. The centralized critic would mislead the policy update to when we train the agents with finite samples as the left part of this figure, while it is correct by only updating the policy of Agent 2 as the right part.

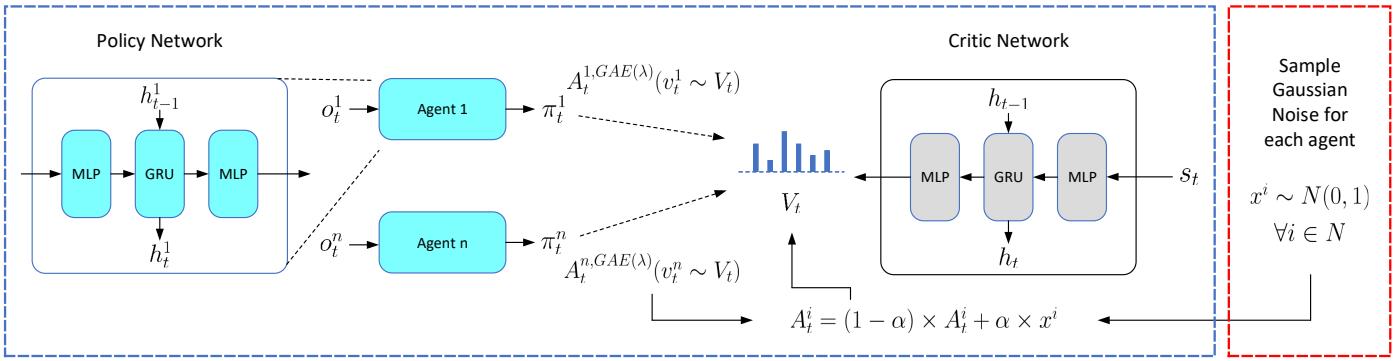


Figure 2. The framework of NA-MAPPO, blue-dash line region refers to the outline of MAPPO, red-dash line region refers to the designed Gaussian noise injected into the centralized critic function.

Noisy Value MAPPO (NV-MAPPO) The essence of NA-MAPPO is to directly disturb the centralized advantage function, which would make the change of policies too drastic. On the other hand, we gently incorporate the noise information into the joint advantage function. We sample a Gaussian noise $\boldsymbol{x}^i \sim \mathcal{N}(0, \sigma^2)$ for each agent i, where σ^2 is the variance that can be regarded as *noise intensity*, and concatenate these noises with state information s. We feed the concatenated vectors to the centralized critic and generate noisy value v^i for each agent i as shown in Figure A1,

$$v^i = V(concat(s, \boldsymbol{x}^i)), \forall i \in N. \tag{7}$$

This sampled noise $\boldsymbol{x}^i$ disturbs the centralized value network and propagates to $\boldsymbol{A}^i = \boldsymbol{r} + \gamma v^i(s_{t+1}) - v^i(s_t)$, for perturbing the shared advantage values. These advantage noises would bring the following benefits:

- The advantage noises prevent the multi-agent policies overfitting caused by the sampled advantage values with deviations and environmental non-stationarity.
- The policies trained by N noisy value networks of agents are similar to policies ensembling, which could enhance the generalization of the joint policy.

- The different noises x^i of each agent stimulate the gradients of policies to go in different directions, which encourage agents to explore diverse high-return trajectories.

In practice, we periodically shuffle these noises from Gaussian distribution, which is derived from noisy net [17], to keep the variety of noise and improve the stability of training. In general, we conclude the pseudo-code of NA-MAPPO in Algorithm 1, the model diagram and pseudo-code of NV-MAPPO is described in Appendix B for clarity.

Algorithm 1 NA-MAPPO.

Initialize parameters $\theta; \phi$; $\mathcal{D} \leftarrow \{\}$; batch size B; N agents; noise variance σ^2;
entropy loss weight η; λ for GAE(λ); Sample Gaussian noise $x^i \sim \mathcal{N}(0,1), \forall i \in N$;
for each episodic iteration **do**
  for episodic step t **do**
    $\vec{a}_t = [\pi^i_\theta(o^i_t), \forall i \in N]$; Execute actions $\vec{a}_t$, observe r_t, s_{t+1}, o_{t+1};
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}\{(s_t, \vec{o}_t, \vec{a}_t, r_t, s_{t+1}, \vec{o}_{t+1}\}$;
  end for
  Sample random batch $\mathcal{B}$ from $\mathcal{D}$;
  Compute advantage $\hat{A}_1, \ldots, \hat{A}_b$ and returns $\hat{R}_1, \ldots, \hat{R}_b$ via $GAE(\lambda)$;
  Mixing the noise with the normalized advantage values by (6);
  for each training epochs **do**
    Update critic by minimizing the loss $L(\phi)$: $L(\phi) = \frac{1}{B}\sum_{b=1}^{B}(v_b(\phi) - \hat{R}_b)^2$;
    Update policy by using PPO loss $L(\theta)$:

$$L(\theta) = \frac{1}{B \cdot N}\sum_{b=1}^{B}\sum_{i=1}^{N}\min[r^i_b(\theta)\hat{A}^i_b, C(r^i_b(\theta), \epsilon)\hat{A}^i_b - \eta\mathcal{H}(\pi^i_\theta(o^i_b))];$$

$$C(r^i_b(\theta), \epsilon) := clip(r^i_b(\theta), 1-\epsilon, 1+\epsilon); r^i_b(\theta) = \frac{\pi^i_\theta(a^i_b|o^i_b)}{\pi^i_{\theta_{old}}(a^i_b|o^i_b)}, \forall i \in N, b \in B;$$

  end for
end for

5. Experiments

In this section, we evaluate the performance of baseline algorithms IPPO and MAPPO [16] and their noisy centralized critic variants on non-monotonic Matrix Game and SMAC. We also compare the performance of NV-MAPPO and NA-MAPPO with the results of the agent-specific features enhanced MAPPO [28] to show the superiority of our noisy critic network. All the variant algorithms are implemented in the PyMARL framework [4], and all the hyperparameters would be kept the same as the baseline algorithms for the sake of fairness. We plot the median results for all experiments over 5 independent runs with random seeds and shade the 25–75% quartile.

Specifically, we give an out line of the two test environments, then list the necessary parameters of the algorithms in the following part of this section; the evaluation results and ablation studies will be presented at the end.

5.1. Testbeds

5.1.1. Non-Monotonic Matrix Game

The non-monotonic Matrix Game is a simple environment to test the cooperative ability of just two agents with three actions each, and the goal of cooperative agents is to take the optimal joint-action and capture the highest reward. The symmetric matrix game has the optimal joint action (A, A), all the agents share the same state information, and the pay-off matrix is shown in Table 1.

5.1.2. SMAC

It is a common and popular practice to test the training effect of multiple agents in a game environment. StarCraft II, as typical real-time strategy (RTS) game, offers a great opportunity to tackle different cooperative challenges in the multi-agent domain. SMAC [4] makes use of Blizzard's StarCraft II machine learning API and DeepMind's PySC2 as an interface for the autonomous agents to interact with the game environment. Each of our training agents can be controlled by an individual army unit in the testing scenarios, which is described as the *decentralized micromanagement* problem in StarCraft II. As in Figure 3, all the agents are trained to battle an opposing army of the game's built-in scripted AI, which can be set to different difficulty levels. Each agent can move in four discrete actions and take attack actions to cause damage to enemies within *shooting range*.

Table 1. Pay-off matrix of one-step game. Boldface means optimal/greedy actions from the state-action value. Both agents need to find the optimal joint actions as soon as possible.

a_1 \ a_2	A	B	C
A	***8***	−12	−12
B	−12	0	0
C	−12	0	0

(**a**) Payoff of matrix game 1

a_1 \ a_2	A	B	C
A	***12***	0	10
B	0	10	10
C	10	10	10

(**b**) Payoff of matrix game 2

Figure 3. Two typical cooperative scenarios in SMAC: 2c_vs_64zg & 3s5z_vs_3s6z.

5.2. Experimental Setup

We implemented all the algorithms in the PyMARL framework [35], and used the same network architecture and hyperparameters for those contrastive methods. Since we selected MAPPO as a representative of the MARL actor–critic style algorithm, the hyperparameters are heavily based on the recent paper [28], which fine-tunes the MAPPO (MAPPO: https://github.com/marlbenchmark/on-policy (accessed on 5 March 2021)). Yu [28] proposes several agent-specific features screening methods to enhance the performance of vanilla MAPPO. It is worth noting that we strip off these artificial features of MAPPO in [28] as testing baseline. Each agent is equipped with a DRQN [36] with a recurrent layer, which has a 64-dimensional hidden state. We set the discount factor $\gamma = 0.99$ and $lr = 5 \times 10^{-4}$ for all testing scenarios (there is a noticeable difference between the different versions of SMAC, we use SC2.4.10 version of SMAC through all the testing scenarios).

In order to speed up the convergence of policies, we adopted 8 individual processes to collect training trajectory. The clip coefficient $\epsilon = 0.2$ and scaling parameter λ in GAE module is 0.95. Still, we add an entropy term to the objective function and the entropy coefficient is 0.01. We paused the training every 10,000 time steps and test 32 episodes to test the cooperative ability. The difficult of game was set to the **Very Difficult** level as default, and we plot the median results for all experiments over 5 independent runs with random seeds and shade the 25–75% quartile.

We list the common hyperparameters of NV-MAPPO and NA-MAPPO in Table 2. For *noise shuffle interval*, we consider $\{0, 50, 100, 200\}$ and choose 100; for σ we consider

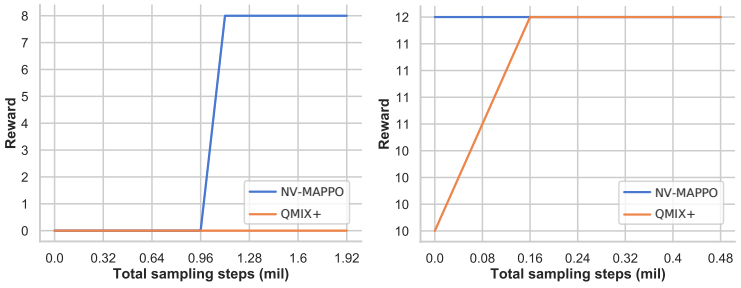
$\{0, 0.05, 0.5, 1, 3, 8, 10\}$, and for α we consider $\{0, 0.05, 0.06, 0.1\}$, then we select proper value for each scenario. As the positive correlation between the variance of noise and the exploration, we encourage researchers to select bigger σ in some other kind of cooperative scenarios to strengthen exploration of agents. Other training hyperparameters related to the test scenarios in SMAC can be found in Table A1 which is summarized in Appendix C, as well as the specific hyperparameters, i.e., σ and α for each scenario.

Table 2. Common hyperparameters used in MAPPO and its variants (NA-MAPPO and NV-MAPPO).

Hyperparameters	MAPPO & MAPG
Envs num	8
Buffer length	400
RNN hidden state dim	64
FC layer dim	64
Noise dim num	10
Adam *lr*	5×10^{-4}
GAE(λ)	0.95
Entropy coef	0.01
PPO clip	0.2
Noise shuffle interval (episodes)	100

5.3. Results

First, we evaluated NV-MAPPO on a non-monotonic Matrix Game to verify the expressive capacity of the centralized value function. These two different payoff matrices in Table 1 are both used to test the cooperative ability that deals with the *Relative Overgeneralization* problem. The results in Figure 4a,b show the superiority of the NV-MAPPO since it could find the optimal joint-action faster even when we implement the fine-tuned version of QMIX (QMIX : https://github.com/hijkzzz/pymarl2 (accessed on 8 August 2021)) [37] (represented by QMIX+ in figures below), which also achieves the *state-of-the-art* level in SMAC and other cooperative tasks.



(**a**) (**b**)

Figure 4. The test returns for NV-MAPPO and QMIX in two kinds of non-monotonic Matrix Games. (**a**) Learning results for Matrix Game 1; (**b**) learning results for Matrix Game 2.

Next, we compared the performance of MAPG and MAPPO with their noisy variants. As shown in Figure 5, both NV-MAPG and NV-MAPPO obtain superb performance in most scenarios, which indicates the dominant advantages of the proposed noise-injection method. Furthermore, we emphatically evaluate NV-MAPPO and NA-MAPPO on three hard-exploration scenarios, and both methods would surpass the baseline with a large margin as shown in Figure 6. We also found that the noisy advantage method could cause some instability in some scenarios, i.e., the results of the noisy advantage method have large variances during training. We speculate that it is too aggressive to directly inject the noises into the advantage function of each agent, which may introduce additional bias in training. Still, the performance of NA-MAPPO is comparable to NV-MAPPO in most scenarios of SMAC.

Moreover, we found that the performance of MAPPO degraded a lot when those agent-specific features in [28] are stripped off. We expect to know whether the performance of NV-MAPPO would come up to the *state-of-the-art* results of SMAC reported in [28]. We tested our methods on most of the Hard and Super-Hard scenarios and list the median results in Table 3. The results demonstrate that the performance of NV-MAPPO significantly exceeds that of MAPPO on most Hard scenarios, such as *5m_vs_6m* (**+65%**), *3s5z_vs_3s6z* (**+31%**), *6h_vs_8z* (**+87%**), and *corridor* (**+97%**). Even without a shared advantage critic, NV-IPPO still achieves extraordinary results in Super-Hard scenarios *3s5z_vs_3s6z* (**96%**) and *6h_vs_8z* (**94%**). We think that the noise injected into the independent critic of agents would also disturb the gradient direction and prevent IPPO from overfitting due to non-stationarity. The average performance of NV-MAPPO calculated from Hard and Super-Hard scenarios still surpasses that of MAPPO with agent-specific features.

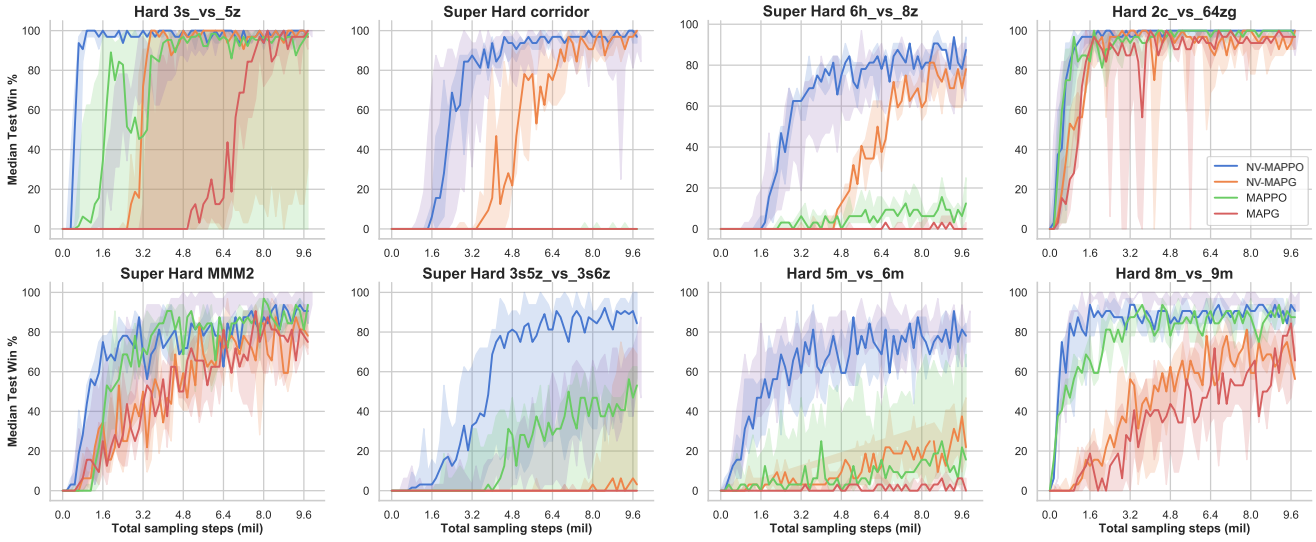


Figure 5. The training curves of MAPG and MAPPO, and their noisy variants in some Hard and Super-Hard testing scenarios in SMAC.

Table 3. Median test win rates of MARL algorithms in scenarios of SMAC. MAPPO^{+} represents the MAPPO with agent-specific features, and **Hard+** includes Hard and Super-Hard scenarios.

Scenarios	Difficulty	NV-MAPPO	NA-MAPPO	MAPPO^{+}	MAPPO	NV-IPPO	IPPO
2s3z	Easy	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
1c3s5z	Easy	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
3s5z	Easy	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
2s_vs_1sc	Easy	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
3s_vs_5z	Hard	**100%**	**100%**	**100%**	98%	**100%**	**100%**
2c_vs_64zg	Hard	**100%**	**100%**	**100%**	**100%**	**100%**	98%
5m_vs_6m	Hard	**89%**	85%	**89%**	25%	87%	87%
8m_vs_9m	Hard	**96%**	**96%**	**96%**	93%	**96%**	96%
MMM2	Super Hard	**96%**	**96%**	90%	**96%**	86%	86%
3s5z_vs_3s6z	Super Hard	87%	72%	84%	56%	**96%**	82%
6h_vs_8z	Super Hard	91%	90%	88%	15%	**94%**	84%
corridor	Super Hard	**100%**	**100%**	**100%**	3%	98%	98%
27m_vs_30m	Super Hard	**100%**	98%	94%	98%	72%	69%
Avg. Score	**Hard+**	**95.5%**	93.2%	93.4%	64.9%	**91.9%**	88.8%

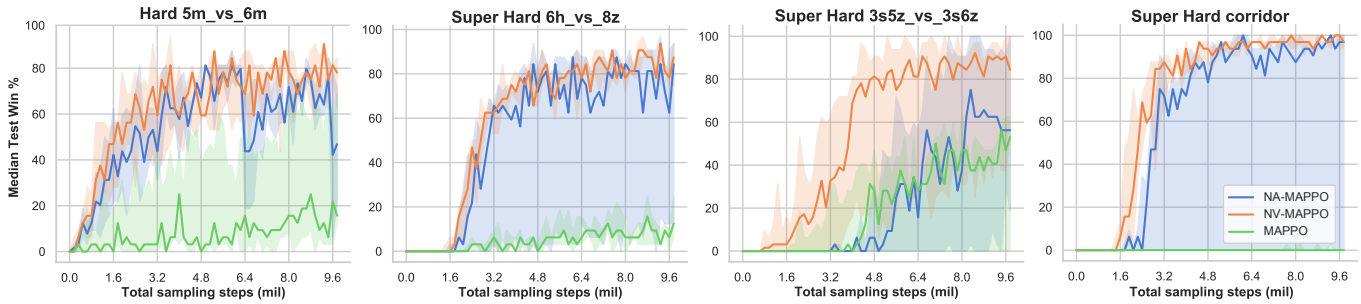
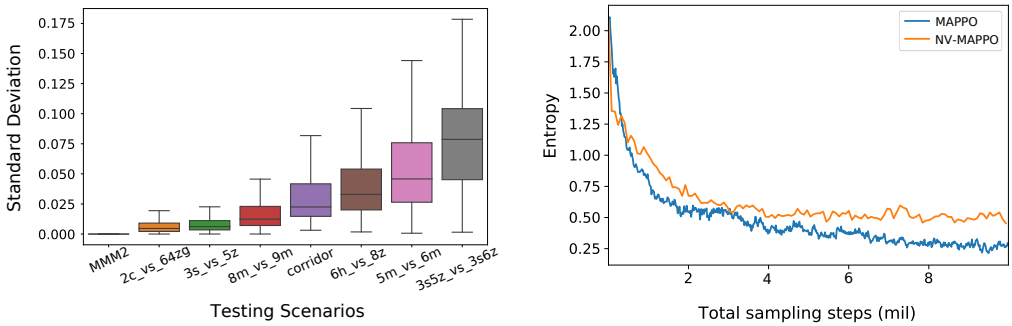


Figure 6. The training curves of proposed methods in some Hard and Super-Hard scenarios.

5.4. Ablations

All the results above indicate that the noisy advantage function would distinctly improve the performance of actor–critic style MARL algorithms. Since the proposed noise is relied upon to disturb the policy gradient direction, it would generally increase the fluctuation of centralized values. Furthermore, we calculate the standard deviation of centralized value function v^i on each agent dimension during training. As the box plot shown in Figure 7a, it seems that the fluctuations of the centralized critic are more dramatic in some scenarios that need more exploration and careful cooperation. Combined with the results in Table 3, we could find there is a positive correlation between the performance improvement and the standard deviation v^i, i.e., the larger fluctuation of v^i would bring the greater performance improvement. This conclusion reveals that the significantly improved results of NV-MAPPO indeed come from the noise we inject into the centralized critic.



(**a**) (**b**)

Figure 7. The increment of performance is positively correlated with the deviation of centralized value function, and injected noise can bring more exploration for agents. (**a**) Standard deviation of centralized value function on the agent dimension. (**b**) Average entropy of agents' policies in *3s5z_vs_3s6z* during training.

Since the proposed noises are not injected into policy network similar to [17,18], they will not directly affect the actions that agents take. Therefore, we explore the reason why these noises could regularize the policies of agents under CTDE paradigm. We calculate the average entropy of the policies of the agents in scenario *3s5z_vs_3s6z*, which has the largest variance across all the testing environments. As shown in Figure 7b, the entropy of MAPPO's policies continuously drops as the training goes on; however, for NV-MAPPO, the entropy would drop at the beginning of training as usual, and then keep fluctuating in a small range. This peculiarity would maintain the entropy in agents' policies higher than that of MAPPO, and hence give rise to more exploration. We think that the fluctuation of centralized critic induced by injected noise will indirectly regularize the policies through gradient backpropagation along with actor–critic architecture. This could explain why the proposed noise-injection method would greatly improve the performance of MAPPO in Super-Hard scenarios that are hard to explore the optimal joint actions of agents.

In the end, we mentioned before that the fixed sampling noise would show some instability of agents in some Hard or Super-Hard cooperative scenarios. Here, we compare the performance of the periodical shuffled Gaussian noise with the fixed sampling noise of NV-MAPPO on *5m_vs_6m* and Super-Hard *corridor*, *6h_vs_8z*. As shown in Figure 8, the performance of these two noise-sampling methods is comparable, but the shuffled Gaussian noise is more stable during training. We implemented the shuffled sampling noise across all the cooperative scenarios in this paper.

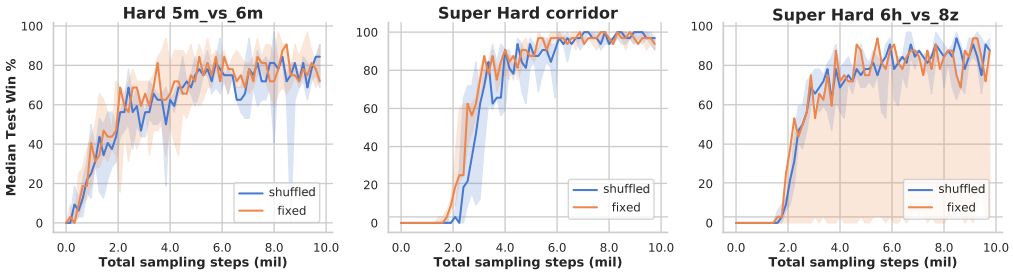


Figure 8. The performance comparison of fixed sampled noise vectors and shuffled noise vectors.

6. Conclusions

In this paper, we depict the *policy overfitting* of actor–critic pattern in MARL and propose a noise-injection method to mitigate this issue. The injected noise can be directly implemented in shared advantage or value function under CTDE paradigm. We theoretically analyze the credit misassignment caused by *policy overfitting* with finite samples and compare its essential differences with COMA. The experimental results show that our proposed method would obtain extraordinarily high win rates and achieve *state-of-the-art* in SMAC, even without artificial agent-specific features. Our work indicates that perturbation induced by the noisy advantage values would effectively improve the performance of multi-agent actor–critic algorithms, and the injected noise would also regularize the policies by maintaining relative higher entropy, then encourage more exploration during training. For future work, it is worth exploring the adaptive noise in both policy and critic networks, or designing a task-specific noise-selecting mechanism. Moreover, it is also worth exploring the regularization method on centralized value network itself with DropOut or other methods to strengthen the representation ability for general MARL algorithms.

Author Contributions: Conceptualization, S.W.; Data curation, S.H.; Methodology, J.H.; Supervision, W.C.; Writing—original draft, S.W.; Writing—review & editing, L.H. All authors have read and agreed to the published version of the manuscript.

Funding: There is no funding support for the work of this paper.

Data Availability Statement: Algorithms and code openly available in a public repository.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In this part, we give out the discrepancy of *policy overfitting* and the explicit credit misassignment problem in COMA. The gradient of COMA [26] is given by

$$g = \mathbb{E}_{\pi}\left[\sum_{i} \nabla_{\theta} \log \pi^{i}\left(a^{i} \mid \tau^{i}\right) A^{i}(s, \mathbf{a})\right], \tag{A1}$$

$$A^{i}(s, \mathbf{u}) = Q(s, \mathbf{a}) - \sum_{a'^{i}} \pi^{i}\left(a'^{i} \mid \tau^{i}\right) Q\left(s, \left(a'^{i}, \mathbf{a}^{-i}\right)\right). \tag{A2}$$

From the equations above, we can conclude that COMA computes an advantage function that compares the Q-value for the current action a^{i} with a counterfactual baseline that marginalizes out a^{i}, while keeping the other agents' actions fixed. This advantage

value relies heavily on action a^i, and we have to exhaustively sample the actions to obtain an accurate counterfactual baseline. On the other hand, *policy overfitting* is caused by the inaccurate estimation of the centralized advantage function brought by the batch sampling mechanism in training. The marginal advantage function relies on the biased exploring actions of the other agents, which is significantly different from the credit misassignment in COMA. This conclusion still holds in other PG-style MARL algorithms.

Appendix B

We describe the model diagram and the omitting pseudo-code of NV-MAPPO in this section, the complete code can be found in https://github.com/hijkzzz/noisy-mappo (accessd on 7 January 2022).

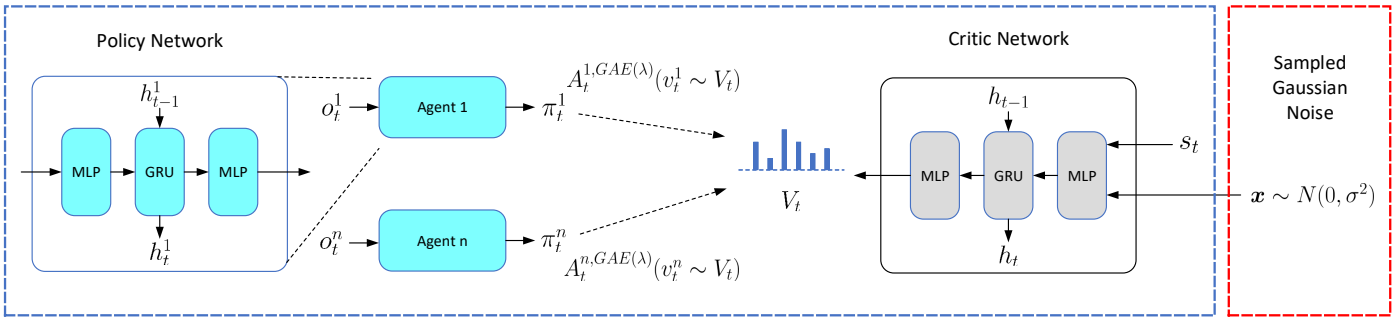


Figure A1. The framework of NV-MAPPO, whose noise-injection is different from NA-MAPPO.

Algorithm A1 NV-MAPPO.

Initialize parameters $\theta; \phi$; $\mathcal{D} \leftarrow \{\}$; batch size B; N agents; noise variance σ^2;
entropy loss weight η; λ for GAE(λ); Sample random noise vectors $\vec{x}^i \sim \mathcal{N}(0, \sigma^2), \forall i \in N$;
for each episodic iteration **do**
  for episodic step t **do**
    $\vec{a}_t = [\pi^i_\theta(o^i_t), \forall i \in N]$; Execute actions $\vec{a}_t$, observe r_t, s_{t+1}, o_{t+1};
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}\{(s_t, \vec{o}_t, \vec{a}_t, r_t, s_{t+1}, \vec{o}_{t+1}\}$;
  end for
  if episodes % shuffle interval **then**
    Shuffle the noise vectors $\vec{x}^i$ on each agent dimension.
  end if
  Sample random batch $\mathcal{B}$ from $\mathcal{D}$; Inject noise into value function for each agent by (7);
  Compute $\hat{A}^i_1, \ldots, \hat{A}^i_b$ and returns $\hat{R}^i_1, \ldots, \hat{R}^i_b$ via $GAE(\lambda)$ with $v^i_b(\phi), \forall i \in N, b \in B$;
  for each training epochs **do**
    Update critic by minimizing the loss $L(\phi)$: $L(\phi) = \frac{1}{B \cdot N} \sum_{b=1}^{B} \sum_{i=1}^{N} (v^i_b(\phi) - \hat{R}^i_b)^2$;
    Update policy by using PPO loss $L(\theta)$:

$$L(\theta) = \frac{1}{B \cdot N} \sum_{b=1}^{B} \sum_{i=1}^{N} \min[r^i_b(\theta)\hat{A}^i_b, C(r^i_b(\theta), \epsilon)\hat{A}^i_b - \eta \mathcal{H}(\pi^i_\theta(o^i_b))];$$

$$C(r^i_b(\theta), \epsilon) := clip(r^i_b(\theta), 1-\epsilon, 1+\epsilon); r^i_b(\theta) = \frac{\pi^i_\theta(a^i_b|o^i_b)}{\pi^i_{\theta_{old}}(a^i_b|o^i_b)}, \forall i \in N, b \in B;$$

  end for
end for

Appendix C

As described in the main text, NV-MAPPO and NA-MAPPO are heavily dependent on MAPPO [16]. To be fair, we develop our NV-MAPPO and NA-MAPPO code based on the project of [16], the omitting hyperparameters for each scenario are listed in Table A1.

Table A1. Hyperparameters for IPPO and MAPPO and their noisy variants in SMAC.

Map	PPO Epochs	Mini-Batch	Gain	Network	Stacked Frames	NV-MAPPO	NV-MAPG	NV-IPPO	NA-MAPPO
						σ	σ	σ	α
2s3z	15	1	0.01	rnn	1	1	1	0.05	0.05
1c3s5z	15	1	0.01	rnn	1	1	1	0.05	0.05
3s5z	5	1	0.01	rnn	1	1	1	0.05	0.05
2s_vs_1sc	15	1	0.01	rnn	1	1	1	0.05	0.05
3s_vs_5z	15	1	0.01	mlp	4	1	1	1	0.05
2c_vs_64zg	5	1	0.01	rnn	1	1	1	1	0.05
5m_vs_6m	10	1	0.01	rnn	1	8	3	0	0.05
8m_vs_9m	15	1	0.01	rnn	1	1	0.05	1	0.05
corridor	5	1	0.01	mlp	1	3	1	1	0.06
MMM2	5	2	1	rnn	1	0	0.5	0	0
3s5z_vs_3s6z	5	1	0.01	rnn	1	10	1	8	0.05
6h_vs_8z	5	1	0.01	mlp	1	1	1	1	0.06
27m_vs_30m	5	1	0.01	rnn	1	1	1	1	0

References

1. Hüttenrauch, M.; Šošić, A.; Neumann, G. Guided deep reinforcement learning for swarm systems. *arXiv* **2017**, arXiv:1709.06011.
2. Kušić, K.; Ivanjko, E.; Vrbanić, F.; Gregurić, M.; Dusparic, I. Spatial-Temporal Traffic Flow Control on Motorways Using Distributed Multi-Agent Reinforcement Learning. *Mathematics* **2021**, *9*, 3081. [CrossRef]
3. Cao, Y.; Yu, W.; Ren, W.; Chen, G. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Trans. Ind. Inform.* **2012**, *9*, 427–438. [CrossRef]
4. Samvelyan, M.; Rashid, T.; De Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.; Hung, C.M.; Torr, P.H.; Foerster, J.; Whiteson, S. The starcraft multi-agent challenge. *arXiv* **2019**, arXiv:1902.04043.
5. Tatari, F.; Naghibi-Sistani, M.B.; Vamvoudakis, K.G. Distributed optimal synchronization control of linear networked systems under unknown dynamics. In Proceedings of the 2017 American Control Conference (ACC), Seattle, WA, USA, 24–26 May 2017; pp. 668–673. [CrossRef]
6. Vamvoudakis, K.G.; Lewis, F.L.; Hudas, G.R. Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality. *Automatica* **2012**, *48*, 1598–1611. [CrossRef]
7. Jiao, Q.; Modares, H.; Xu, S.; Lewis, F.L.; Vamvoudakis, K.G. Multi-agent zero-sum differential graphical games for disturbance rejection in distributed control. *Automatica* **2016**, *69*, 24–34. [CrossRef]
8. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
9. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the ICML, Amherst, MA, USA, 27–29 June 1993; pp. 330–337.
10. Oliehoek, F.A.; Spaan, M.T.; Vlassis, N. Optimal and approximate Q-value functions for decentralized POMDPs. *J. Artif. Intell. Res.* **2008**, *32*, 289–353. [CrossRef]
11. Lowe, R.; WU, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390.
12. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the ICML, Long Beach, CA, USA, 9–15 June 2019; pp. 2961–2970.
13. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the ICML, Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.
14. Ha, D.; Dai, A.; Le, Q.V. Hypernetworks. *arXiv* **2016**, arXiv:1609.09106.
15. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
16. de Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv* **2020**, arXiv:2011.09533.
17. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Hessel, M.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; et al. Noisy Networks for Exploration. In Proceedings of the ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.
18. Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R.Y.; Chen, X.; Asfour, T.; Abbeel, P.; Andrychowicz, M. Parameter Space Noise for Exploration. In Proceedings of the ICLR, Vancouver, BC, Canada, 30 April–3 May 2018.
19. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
20. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In Proceedings of the AAMAS, Stockholm, Sweden, 10–15 July 2018; pp. 2085–2087.
21. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the ICML, Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.
22. Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; Zhang, C. Qplex: Duplex dueling multi-agent q-learning. *arXiv* **2020**, arXiv:2008.01062.

23. Zhou, M.; Liu, Z.; Sui, P.; Li, Y.; Chung, Y.Y. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *arXiv* **2020**, arXiv:2007.02529.
24. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the ICLR (Poster), San Juan, Puerto Rico, 2–4 May 2016.
25. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *A*dv. Neural Inf. Process. Syst. **2017**, *30*, 5998–6008.
26. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI, New Orleans, LO, USA, 2–7 February 2018; Volume 32.
27. Peng, B.; Rashid, T.; de Witt, C.A.S.; Kamienny, P.A.; Torr, P.H.; Böhmer, W.; Whiteson, S. FACMAC: Factored Multi-Agent Centralised Policy Gradients. *arXiv* **2020**, arXiv:2003.06709.
28. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv* **2021**, arXiv:2103.01955.
29. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust region policy optimisation in multi-agent reinforcement learning. In Proceedings of the ICLR, Virtual, 25–29 April 2022.
30. Mahajan, A.; Rashid, T.; Samvelyan, M.; Whiteson, S. MAVEN: Multi-agent variational exploration. In Proceedings of the NeuIPS, Vancouver, Canada, 8–14 December 2019; pp. 7613–7624.
31. Wang, T.; Dong, H.; Lesser, V.; Zhang, C. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In Proceedings of the ICML, Virtual, 13–18 July 2020; pp. 9876–9886.
32. Pan, L.; Rashid, T.; Peng, B.; Huang, L.; Whiteson, S. Regularized Softmax Deep Multi-Agent Q-Learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 1365–1377.
33. Liu, I.J.; Jain, U.; Yeh, R.A.; Schwing, A. Cooperative exploration for multi-agent deep reinforcement learning. In Proceedings of the ICML, Virtual, 18–24 July 2021; pp. 6826–6836.
34. Ong, S.C.; Png, S.W.; Hsu, D.; Lee, W.S. POMDPs for robotic tasks with mixed observability. In Proceedings of the Robotics: Science and Systems, Seattle, WA, USA, 28 June–1 July 2009; Volume 5, p. 4.
35. Böhmer, W.; Kurin, V.; Whiteson, S. Deep coordination graphs. In Proceedings of the International Conference on Machine Learning, Virtual , 13–18 July 2020; pp. 980–991.
36. Hausknecht, M.; Stone, P. Deep recurrent q-learning for partially observable mdps. In Proceedings of the 2015 AAAI Fall Symposium Series, Austin, TX, USA, 25–30 January 2015.
37. Hu, J.; Jiang, S.; Harding, S.A.; Wu, H.; Liao, S.W. Revisiting the Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning. *arXiv* **2021**, arXiv:2102.03479.