MDPI

Article

Diversified Experience Replay for Multi-Agent Reinforcement Learning

Guangchong Zhou [1,2], Feng Hong [3], Zeren Zhang [1,2] and Guoliang Fan [1]

[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] Cooperative Medianet Innovation Center, Shanghai Jiao Tong University
{zhouguangchong2021, zhangzeren2021, guoliang.fan}@ia.ac.cn, feng.hong@sjtu.edu.cn

Abstract: To enhance the capability of off-policy multi-agent reinforcement learning (MARL), previous research has extensively investigated the agents' decision-making and credit assignment. However, the role of experience replay is largely overlooked, with related works limited to prioritizing samples based on their TD-errors. Despite their improvements, more accurate Q-value estimations do not guarantee better decisions, as the relative advantage of different situations is more crucial for the greedy policy. To this end, we propose **Div**ersified **E**xperience **R**eplay (DivER), which increases the experience diversity in the sampled mini-batch. Agents are guided to learn behaviors that transition to more superior situations, thus enhancing training efficiency. DivER is compatible with any off-policy MARL methods and has been experimentally proven to be effective across various tasks and algorithms.

Keywords: Off-policy MARL; Prioritized experience replay; Sample diversity

0. Introduction

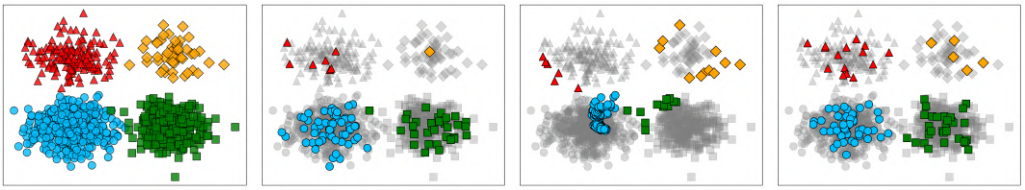
Multi-agent reinforcement learning (MARL) has recently demonstrated remarkable capability in handling various real-world tasks, such as flocking control [1,2], autonomous driving [3,4], and traffic light control [5]. To deal with the partial observability and the vast joint spaces of agents, most MARL algorithms follow the *centralized training and decentralized execution* (CTDE) paradigm [6] where global information is only accessible during training. Especially, off-policy methods [7,8] enjoy high sample efficiency and great scalability, ultimately achieving superior performance in popular benchmarks.

Off-policy methods still encounter challenges including insufficient exploration, behavioral homogeneity, and inaccurate credit assignment. To address them, a number of recent studies have focused on improving decision-making mechanisms [9–13] and modifying the calculation of value mixing [14–17]. However, the experience replay, a technique widely used by off-policy methods to boost sample efficiency, has been severely overlooked. A novel research is MAC-PO [18] that prioritizes the samples in replay buffer via regret minimization. To our knowledge, no other research on multi-agent experience replay has gained recognition recently, and Prioritized Experience Replay (PER) [19] and its variants remain the most widely used approaches. PER prioritizes the samples with high magnitude of TD-errors and subsequently obtains more accurate global Q-value estimations.

Figure 1 shows the results of different mini-batch sampling strategies on a toy replay buffer. It can be seen that the default uniform sampling strategy does not guarantee comprehensive coverage of all experiences, while the outcomes of PER based on TD-error

Received:
Revised:
Accepted:
Published:

Citation: Zhou, G.; Hong, F.; Zhang, Z.; Fan, G. Diversified Experience Replay for Multi-Agent Reinforcement Learning. *Journal Not Specified* **2025**, *1*, 0.



(a) Full data in replay buffer (b) Uniform (c) TD-error PER (d) DivER

Figure 1. The comparison of different sampling methods on a toy replay buffer, in which the transition data is distributed in four regions. (a) The visualization of all data in the replay buffer. (b)(c)(d) The mini-batch derived by the Uniform, PER, and DivER sampling strategies.

exhibit significant homogeneity and imbalance. We argue that, as agents in most off-policy methods are value-based and employ greedy policies, the relative superiority among different states can better guide the agents' decision-making towards superior states rather than precise state value estimations. Therefore, at each training step, a mini-batch with greater sample diversity is more conducive to the agent's understanding of the relative merits of different experiences, thereby accelerating policy improvement. To this end, we propose **Div**ersified **E**xperience **R**eplay (DivER), a novel and flexible method to improve the sample diversity of the mini-batch at each training step. DivER learns a representation model to transform the entire episode of states into a compact vector, based on which it measures the discrepancy between experiences in the replay buffer and generates the mini-batch of diversified samples. Our main contributions can be summarized as follows:

- We explore the multi-agent experience replay which is largely overlooked by previous research, and are the very first to take the sample diversity of the mini-batch into consideration, thereby furnishing a novel perspective for subsequent studies.
- We propose a prioritized experience replay method called DivER, which improves the sample diversity of the mini-batch at each training step. As DivER only modifies the experience replay mechanism without altering the MARL framework, it can serve as a plug-and-play technique for any off-policy MARL methods and achieve stable improvements.

1. Background

1.1. Dec-POMDP

A fully cooperative multi-agent system (MAS) is typically represented by a decentralized partially observable Markov decision process (Dec-POMDP) [20], which is composed of a tuple $G = \langle \mathcal{S}, \boldsymbol{\mathcal{U}}, \mathcal{P}, \mathcal{Z}, r, \mathcal{O}, n, \gamma \rangle$. At each time-step, the current global state of the environment is denoted by $s \in \mathcal{S}$, while each agent $a \in \mathcal{A} := \{1, \ldots, n\}$ only receives a unique local observation $z_a \in \mathcal{Z}$ generated by the observation function $\mathcal{O}(s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$. Subsequently, every agent a selects an action $u_a \in \mathcal{U}$, and all individual actions are combined to form the joint action $\boldsymbol{u} = [u_1, \ldots, u_n] \in \boldsymbol{\mathcal{U}} \equiv \mathcal{U}^n$. The interaction between the joint action $\boldsymbol{u}$ and the current state s leads to a change in the environment to state s' as dictated by the state transition function $\mathcal{P}(s'|s, \boldsymbol{u}) : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$. All agents in the Dec-POMDP share the same global reward function $r(s, \boldsymbol{u}) : \mathcal{S} \times \boldsymbol{\mathcal{U}} \to \mathbb{R}$, and $\gamma \in [0, 1)$ represents the discount factor.

1.2. One-shot Coreset Selection

In supervised learning, let us consider a classification task with a training dataset containing N examples drawn i.i.d. from an underlying distribution P. The dataset is denoted as $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ is the data and y_i is the ground-truth label. Given a

pruning rate $\alpha \in (0,1)$, the goal of one-shot coreset selection is to select a training subset $\mathcal{B}$ from the whole dataset to maximize the accuracy of models trained on this subset, which can be formulated as the optimization problem below [21]:

$$\min_{\mathcal{B}\subset\mathcal{D}, \frac{|\mathcal{B}|}{|\mathcal{D}|}\leq 1-\alpha} \mathbb{E}_{(\boldsymbol{x},y)\sim P}[l(\boldsymbol{x}, y; h_{\mathcal{B}})], \tag{1}$$

where $l(\cdot,\cdot)$ is the loss function, and $h_{\mathcal{B}}$ is the model trained on the subset $\mathcal{B}$. To find the optimal subset, previous methods typically rank the examples based on their importance and select the most important $|\mathcal{B}|$ examples to form the subset. Different metrics, including prediction error [22,23], gradient norm [24], area under the margin (AUM) [25], and EL2N score [26], are proposed to measure the importance of each sample.

Off-policy MARL methods commonly employ the replay buffer, from which a mini-batch of samples are selected for subsequent model updates. However, the training process of MARL differs significantly from supervised learning, and the design of effective coreset selection (known as prioritized experience replay) methods for MARL requires further investigation.

2. DivER

2.1. Motivation

In this section, we mainly focus on value-based methods and use $\mathcal{D}$ and $\mathcal{B}$ to denote the replay buffer and sampled mini-batch in MARL, respectively. Most existing methods directly extend PER [19] to MARL, which prioritizes the samples with high magnitude of TD-errors. Therefore, these methods still follow the supervised learning target in Equation 1 with the following loss function:

$$l(s_t, \boldsymbol{u}_t, r_t, s_{t+1}) = \left(r_t + \gamma \max_{\boldsymbol{u}_{t+1}} Q^{-}_{tot}(s_{t+1}, \boldsymbol{u}_{t+1}) - Q_{tot}(s_t, \boldsymbol{u}_t)\right)^2,$$

in which s_t and $\boldsymbol{u}_t$ are the global state and joint action at time step t, Q^{-}_{tot} is a periodically updated target network of Q_{tot}. As the experiment results in Figure 2 show, the absolute value of TD-error during the training process of QMIX [8] is significantly reduced with the integration of PER (named QMIX-PER), indicating that PER effectively improves the accuracy of Q-value estimation. However, a more accurate value estimation does not necessarily lead to the policy improvement and the increase in the overall return of the system. Considering the training paradigm of MARL, we modify the objective for mini-batch selection at each training step to Equation 2:

$$\begin{aligned}\max_{\mathcal{B}\subset\mathcal{D}} J(\pi') - J(\pi) &= \max_{\mathcal{B}\subset\mathcal{D}} \mathbb{E}_{s_0}[V^{\pi'}(s_0)] - \mathbb{E}_{s_0}[V^{\pi}(s_0)] \\ &= \max_{\mathcal{B}\subset\mathcal{D}} \mathbb{E}_{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t [r(s_t, a_t) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)]\right],\end{aligned} \tag{2}$$

where π' is the updated policy based on π after trained on mini-batch $\mathcal{B}$ and $V(\cdot)$ is the state value function. We assert that, rather than precisely predicting the system value in every state, understanding the relative superiority among states could be more helpful in guiding the agents to learn behaviors that transition to superior states, thus facilitating efficient policy improvement. To this end, we propose **Div**ersified **E**xperience **R**eplay (DivER), which enhances the training efficiency by increasing the experience diversity in the sampled mini-batch.

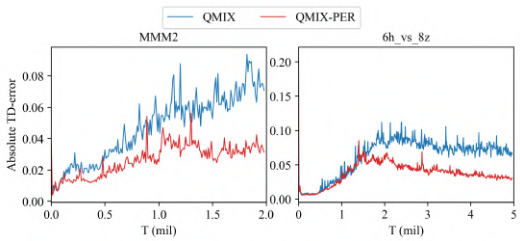


Figure 2. The absolute value of TD-error during the training processes of QMIX with and w/o PER on two SMAC scenarios.

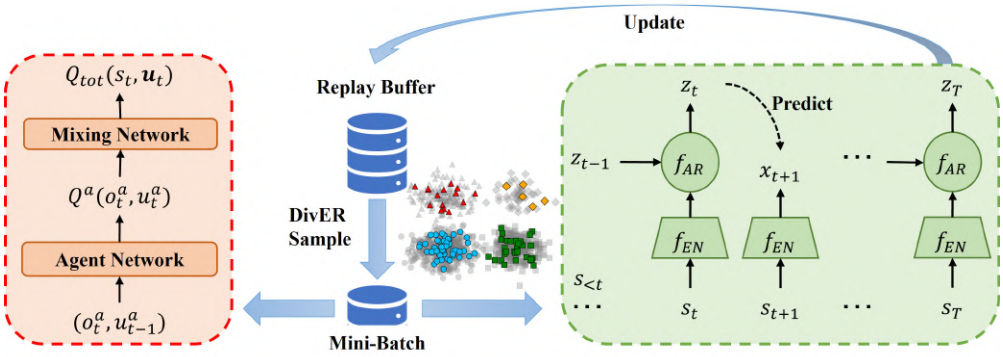


Figure 3. Overview of DivER architecture. Red dashed box depicts the off-policy MARL method. Green dashed box is the representation model in DivER. f_{EN} and f_{AR} denote the state encoder and the autoregressive model respectively.

2.2. Learning Episode Representation

Generally, each piece of data in MARL includes information from all time steps of an entire episode from start to end. Therefore, a single state at a specific time step cannot sufficiently represent the whole episode, nor can it be directly used to compare similarities with other episodes. Instead, we treat the sequence of states in an episode as a trajectory τ of the system, and summarize them into a compact vector z. The mutual information between τ and z measures the reduction of the uncertainty of trajectory prediction after knowing the representation z, which is defined as Equation 3. By maximizing the mutual information, we extract the underlying latent information that the trajectory and its representation have in common.

$$\mathcal{I}(\tau; z) = \sum_{\tau, z} p(\tau, z) \log \frac{p(\tau|z)}{p(\tau)} \tag{3}$$

Next, it is necessary to design a learnable network architecture based on the training objectives. A severe challenge is that a powerful generative model for $p(\tau|z)$ in Equation 3 is computationally intense and would waste capacity at modeling the complex relationships between the states in the sequence. Besides, unimodal losses like mean squared error and cross-entropy are not sufficient for predicting high-dimensional data. Inspired by Oord et al. [27], we do not predict the future trajectory directly with a generative model, but we model a density ratio which preserves the mutual information between the next trajectory τ_{t+1} and the representation of current trajectory z_t as:

$$f(x_{t+1}, z_t) \propto \frac{p(x_{t+1}|z_t)}{p(x_{t+1})}, \tag{4}$$

in which $\tau_{t+1} = [\tau_t, x_{t+1}]$ is simplified to x_{t+1}, since τ_t is already known when calculating z_t. To guarantee the output of f is a positive real score, we implement it with a simple log-bilinear function:

$$f(x_{t+1}, z_t) = \exp\Big(x_{t+1}^T W z_t\Big), \tag{5}$$

where x_{t+1} is the encoded state at next step. More complex networks can also be used to construct f. With the density ratio as an alternative object, the networks are relieved from modeling the high dimensional trajectory. Though the distribution $p(\tau)$ or $p(\tau|z)$ cannot be evaluated by the model, we can estimate them using samples from the replay buffer instead. The overall architecture of DivER is depicted in Figure 3, and it can be clearly seen that the workflow of DivER is fully decoupled from the MARL framework. At each time step, the state is encoded and fed into the autoregressive model to generate the trajectory representation z_t, which helps predict the encoded state x_{t+1} at next step.

Consider a mini-batch $\mathcal{B}$ of N episodes with length T. For each episode, z_t has one positive label $p(x_{t+1}^i|z_t)$ and $N-1$ negative samples $\{x_{t+1}^j|j = 1, ..., i-1, i+1, ..., N\}$ from other episodes. By referring to contrastive learning [27,28], we learn the representation model by maximizing the target in Equation 6 as the following lemma holds:

Lemma 1. *Equation 6 is a lower bound for mutual information $\mathcal{I}(x_{t+1}, z_t)$, and maximizing it leads to $f(x_{t+1}, z_t)$ approximating the density ratio in Equation 4.*

$$\mathcal{J}_D = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \frac{f(x_{t+1}^i, z_t^i)}{\sum_{j=1}^{N} f(x_{t+1}^j, z_t^j)} \tag{6}$$

Proof. See Appendix B.1. □

With the learned representation model, we can summarize the whole state sequence of length T into a compact vector z_T for downstream tasks.

2.3. Sampling Methodology

Next, we need to sample episodes from the replay buffer to obtain the mini-batch based on their trajectory representations. Previous methods for prioritized experience replay commonly measure the importance of samples using metrics such as training errors and gradients, then rank the samples and select the most significant ones. However, the ranking approach is not applicable to DivER, as "diversity" is not an attribute possessed by a single agent but rather a global metric for the overall system. Identifying a fixed-size mini-batch with the optimal diversity from the replay buffer constitutes an NP problem, making it both impractical and unworthy to solve. Therefore, the sampling strategy of DivER needs to strike a balance between diversity and computational efficiency. Given a replay buffer $\mathcal{D}$ and sampled mini-batch $\mathcal{B}$, we first employ the concept of "coverage radius" to interpret the diversity:

Definition 2 (r-cover and coverage radius)**.** A set of points $\mathcal{D}$ are distributed in a metric space (X, d). We say mini-batch $\mathcal{B} \subset \mathcal{D}$ is a r-cover of $\mathcal{D}$ if:

$$\mathcal{D} \subseteq \bigcup_{x \in \mathcal{B}} B(x, r),$$

where $B(x, r) = \{x' \in X | d(x', x) < r\}$ is an open ball of radius r centered at point x. The lower bound of r is called the coverage radius.

When the mini-batch size is fixed, a smaller coverage radius indicates that the mini-batch encompasses diverse episodes and represents the entire replay buffer more sufficiently.

While existing methods like uniform sampling and PER can not guarantee a small coverage radius due to randomness and distribution imbalance, we propose a new sampling methodology for DivER presented in Algorithm A1 in Appendix C.

Simply speaking, we first cluster the trajectory representations z_T of different episodes in the latent space to obtain multiple spherical clusters. Next, DivER uniformly samples an equal proportion of episodes from each cluster, which are then merged to form the final mini-batch. The episodes from different clusters help maintain a low coverage radius and high sample diversity for the mini-batch $\mathcal{B}$, while the uniform sampling within each cluster guarantees the computational efficiency of DivER.

2.4. Implementation

Overall learning algorithm. The overall algorithm of DivER after integrating with the off-policy MARL framework is displayed in Algorithm A2 in Appendix C. On the basis of the original MARL method, DivER only introduces an additional representation model during training and fully preserves the efficiency of the decision-making process. The parameter updates of the MARL framework and the representation model in DivER are independent and do not interfere with each other, enabling DivER to be easily integrated with different frameworks and achieve stable performance.

Hyperparameter settings. The clustering algorithm and the number of clusters are two main hyperparameters for DivER. DivER employs the K-Means clustering algorithm by default in our implementation but is also compatible with various other algorithms such as DBSCAN [29] and Gaussian Mixture Models, and choosing a clustering algorithm that aligns with the true sample distribution can improve the effectiveness of DivER. A larger number of clusters requires higher computational costs but also leads to a smaller size for each cluster, which means a corresponding reduction in the coverage radius and improved sample diversity. For fairness, we set the number of clusters to 8 for algorithms that specify this hyperparameter, and employ the default hyperparameters for other algorithms.

3. Experiments

3.1. Experimental Setup

Environments. We select the most popular environment for MARL, StarCraft II Multi-Agent Challenges (SMAC) [30], as the main testbed to facilitate an intuitive comparison with previous off-policy MARL methods. SMAC contains two armies of units, and each ally is controlled by a decentralized agent that can only act based on its local observation while the enemy units are controlled by built-in handcrafted heuristic rules. To address SMAC's lack of stochasticity and get rid of open-loop policies, we further conduct experiments on the more challenging SMACv2 [31], which severely restricts the observability of agents and randomly initializes the scenarios. In our experiments, the difficulty level of built-in AI is set to 7 (very hard), and the version of StarCraft II engine is 4.6.2 instead of the simpler 4.10. Please note that results from different versions are not always comparable.

Baselines. In our experiments, we select the most popular and recognized MARL framework QMIX [8] as backbone, on which we conduct three different sampling strategies, including uniform, PER [19] and MAC-PO [18], as baselines. Uniform and PER are two common sampling strategies, while MAC-PO prioritizes samples in MARL via regret minimization. All baselines are implemented using the source codes in their original papers.

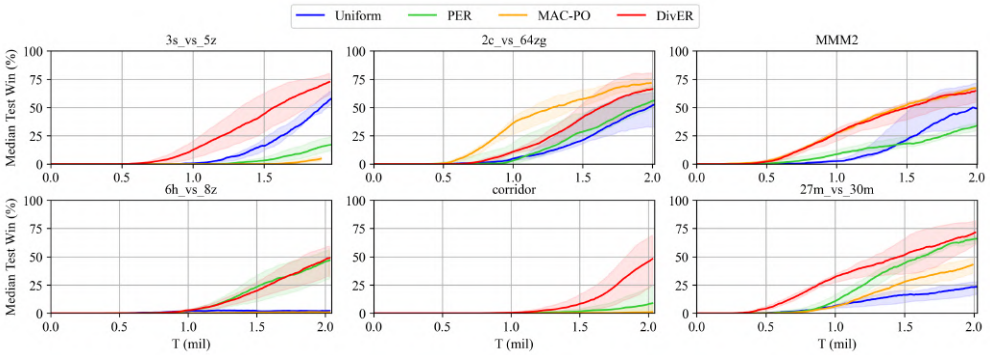


Figure 4. Experiment results of DivER and baselines on SMAC.

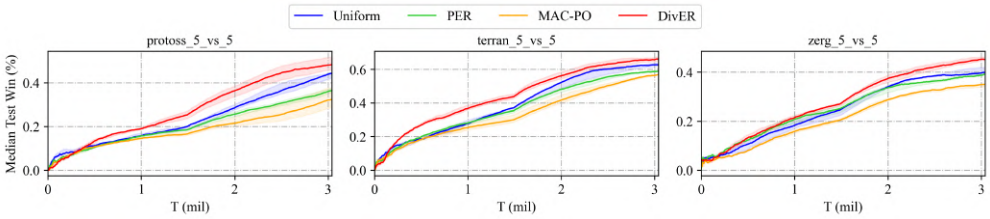


Figure 5. Experiment results of DivER and baselines on SMACv2.

3.2. Experiment Results on SMAC

We present experiment results on 6 *Hard* and *Super Hard* SMAC scenarios in Figure 4. The solid line represents the median win rates during training, with the 25-75% percentiles being shaded.

Although RODE and LDSA achieve higher win rates than DivER in *3s_vs_5z*, both of them promote multi-agent cooperation heuristically by increasing behavioral diversity of agents, leading to unstable performances across scenarios. Meanwhile, DivER focuses on sample diversity in the mini-batch, which can more broadly adapted to different scenarios. As we can see, DivER reaches the best performances in 4 out of all 6 scenarios with both the highest win rates and the earliest rise-ups, indicating its great sample efficiency. Since DivER is implemented on the basis of QMIX, we can observe that its performance significantly surpasses that of QMIX in all scenarios. Besides, DivER also outperforms the other prioritized experience replay method QMIX-PER. These evidences demonstrate the effectiveness and superiority of our proposed method.

3.3. Experiment Results on SMACv2

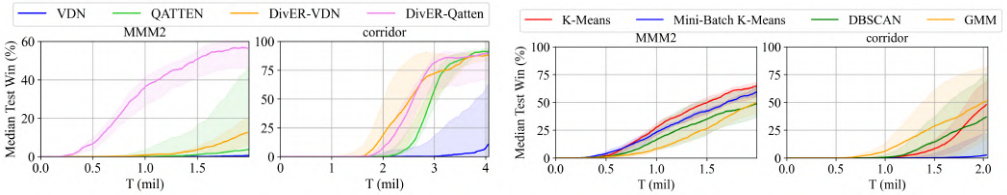
Ellis et al. [31] argue that the agent in SMAC may learn open-loop policy conditioned on the time step rather than the observation, thus can not adapt to diverse situations. Therefore, we further test DivER in the more random and difficult SMACv2 environment, and results are shown in Figure 5. Meanwhile, PER sometimes even degrades the performance of QMIX, and methods based on behavioral diversity exhibit severe performance decline. Meanwhile, DivER makes stable improvements on the basis of QMIX, and its learning curves are positioned above and to the left of the baselines in all three scenarios. This suggests the high sample efficiency and stable performance of DivER, highlighting the broad effectiveness and great reliability of our proposed mini-batch sampling strategy based on sample diversity.

3.4. Ablation Study

In this subsection, we conduct ablation studies to investigate two issues: (a) Can DivER be migrated to other frameworks? (b) Is DivER compatible with different clustering

algorithms? Two challenging *Super Hard* tasks, *MMM2* and *corridor*, are employed as testbeds.

The wide applicability of DivER. As previously asserted, DivER can be integrated with any off-policy MARL framework. To better prove this claim, we further test the efficacy of integrating DivER with VDN [7] and Qatten [16] by comparing their performances with the original algorithms, and the results are in Figure 6(a). VDN is the simplest value-decomposition method and could not solve the two tasks at all, but its capability is significantly enhanced with the integration of DivER. In *MMM2*, DivER also significantly enhances the performance of Qatten. Meanwhile, although the final win rates of Qatten with and without DivER are very close in *corridor*, DivER still notably accelerates the learning process. This shows that DivER can be widely used in different MARL frameworks and maintain great effectiveness.



(a) DivER with other off-policy MARL methods. (b) DivER with different clustering algorithms.

Figure 6. The results of DivER's variants.

Compatibility with different clustering algorithms. As different clustering algorithms fit in different sample distributions and have distinct computational costs, DivER needs to be compatible with diverse algorithms to adapt to various tasks. Besides the default K-Means algorithm, we have also investigated Mini-Batch K-Means, DBSCAN, and Gaussian Mixture Models (GMM), and results are shown in Figure 6(b). While GMM is not suitable for the *MMM2* scenario, it exhibits the best performance in *corridor*. Mini-Batch K-Means enjoys high computational efficiency and performs close to K-Means in *MMM2*, but the inaccurate clustering also leads to failure in *corridor*. Therefore, choosing an appropriate clustering algorithm is of great importance for improving both the speed and performance of DivER.

4. Conclusion

Previous works have extensively investigated approaches to enhance off-policy MARL algorithms' performance, but few have considered the experience replay. Most related works still follow the prioritized experience replay techniques in the single-agent domain, which rank all experiences in the replay buffer based on their "importance scores" and select the most significant ones. In this work, we argue that a mini-batch with a higher sample diversity can help the model learn the relative advantage of different situations, thus accelerating the improvement of greedy policies. To this end, we propose a novel technique called **Div**ersified **E**xperience **R**eplay (DivER) to improve the sample diversity of the mini-batch in every training step. DivER first learns a representation model to embed the information of the entire sequence of states, based on which it then clusters the experiences in the replay buffer into groups and selects diversified experiences for the mini-batch. This sample-diversity perspective demonstrates broad effectiveness across a variety of tasks. Besides, the workflow of DivER is independent of the MARL framework and does not affect the efficiency of agents' decision-making, making it a plug-and-play technique compatible with any off-policy MARL framework. We hope DivER can serve as a versatile and stable method for addressing general multi-agent tasks, as well as providing a new perspective for further research.

Appendix A Related Works

Appendix A.1 Off-policy MARL

In off-policy multi-agent reinforcement learning (MARL), agents learn from past experiences collected using a policy different from the current one, thereby improving the data efficiency. Value-based MARL algorithms are almost exclusively off-policy. Independent Q-learning [32] trains independent action-value functions for each agent, which is later combined with deep learning techniques by Tampuu et al. [33]. Under the CTDE framework, value decomposition is widely used to achieve credit assignment. VDN [7] and QMIX [8] estimate the optimal joint action-value function by combining mentioned utilities via a summation function and a learned state-dependent monotonic function, respectively. QTRAN [34] and QPLEX [15] further loose the monotonicity constraints in QMIX and extend the class of value functions. Weighted-QMIX [14] introduces a weighting mechanism into the projection of monotonic value factorization to place more importance on better joint actions. QPro [17] casts the factorization problem as regret minimization over the projection weights of different state-action values. For policy-based off-policy methods, MADDPG [6] utilizes the ensemble of policies for each agent that leads to more robust multi-agent policies, showing strength in cooperative and competitive scenarios. The extensions [35–37] of MADDPG have been proposed to realize further improvements on the original algorithm.

Appendix A.2 Experience Replay

Online reinforcement suffers from two issues: (a) strongly correlated transitions that break the i.i.d. assumption in deep learning, and (b) the rapid forgetting of possibly rare experiences that would be useful later on. Experience replay [38] stores the collected experiences in a replay buffer for subsequent reuse, which effectively mitigates the above issues and improves the sample efficiency. While most off-policy methods uniformly sample transitions from the replay buffer, Schaul [19] asserts that some experiences are more valuable for model training than others and proposes prioritized experience replay (PER), which replays transitions with higher magnitude of temporal-difference (TD) error more frequently. Subsequently, further modifications and improvements have been made to the PER algorithm. Lee et al. [39] design a sampling technique which updates the transitions backward from a whole episode. PSER [40] prioritizes sequences of experience instead of single transitions. ERO [41] learns a replay policy to optimize the prioritization function. To favor recent transitions and abandon the outdated ones, Sun et al. [42] sample transitions according to the similarities between their states and the agent's state, while Novati and Koumoutsakos [43] control the similarity between the replay behaviors and the current policy. Liu et al. [44] use the regret minimization method to design the prioritized experience replay scheme. MaPER [45] improves experience replay by using a model-augmented critic network and modifying the rule of priority. So far, most PER works are designed for single-agent reinforcement learning. To extend PER to multi-agent scenarios, some research directly migrates previous works to off-policy MARL methods [46,47], while a limited number of studies consider the adaptation to multi-agent systems [18,48].

Appendix B Proofs

Appendix B.1 The Learning Target for DivER Model

Lemma A1. *The Equation below is a lower bound for mutual information $\mathcal{I}(x_{t+1}, z_t)$, and maximizing it leads to $f(x_{t+1}, z_t)$ approximating the density ratio in Equation 4.*

$$\mathcal{J}_D = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \frac{f(x_{t+1}^i, z_t^i)}{\sum_{j=1}^{N} f(x_{t+1}^j, z_t^j)}$$

Proof. This Equation can be viewed as the categorical cross-entropy of classifying the positive sample in contrastive learning, with $\frac{f}{\sum f}$ being the predicted probability of the model. We rewrite the optimal probability as $p(d = i|X_{t+1}, z_t^i)$ with $d = i$ being the indicator that x_{t+1}^i is the positive sample, and $X_{t+1} = \{x_{t+1}^1, ..., x_{t+1}^N\}$ being the set of all encoded state at time $t+1$ in mini-batch $\mathcal{B}$. The probability that x_{t+1}^i was drawn from the conditional distribution $p(x_{t+1}|z_t^i)$ rather than the prior distribution $p(x_{t+1})$ can be derived as below:

$$\begin{aligned} p(d = i|X, z_t^i) &= \frac{p(x_{t+1}^i|z_t^i) \prod_{x_{t+1}^k \in X_{t+1} \setminus \{x_{t+1}^i\}} p(x_{t+1}^k)}{\sum_{j=1}^{N} p(x_{t+1}^j|z_t^i) \prod_{x_{t+1}^k \in X_{t+1} \setminus \{x_{t+1}^i\}} p(x_{t+1}^k)} \\ &= \frac{\frac{p(x_{t+1}^i|z_t^i)}{p(x_{t+1}^i)}}{\sum_{j=1}^{N} \frac{p(x_{t+1}^j|z_t^i)}{p(x_{t+1}^j)}} \end{aligned}$$

According to the above results, the optimal value for $f(x_{t+1}^i, z_t^i)$ in Equation 6 is proportional to $\frac{p(x_{t+1}^i|z_t)}{p(x_{t+1}^i)}$, and is independent of the choice of the mini-batch size N.

Replace the term $f(x_{t+1}, z_t)$ in $\mathcal{J}_{\mathrm{D}}$ with $\frac{p(x_{t+1}|z_t)}{p(x_{t+1})}$, we have:

$$\begin{aligned} \mathcal{J}_{\mathrm{D}} &= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \left[\frac{\frac{p(x_{t+1}^i|z_t^i)}{p(x_{t+1}^i)}}{\frac{p(x_{t+1}^i|z_t^i)}{p(x_{t+1}^i)} + \sum_{x_{t+1}^j \in X_{t+1} \setminus \{x_{t+1}^i\}} \frac{p(x_{t+1}^j|z_t^i)}{p(x_{t+1}^j)}} \right] \\ &= -\sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \left[1 + \frac{p(x_{t+1}^i)}{p(x_{t+1}^i|z_t^i)} \sum_{x_{t+1}^j \in X_{t+1} \setminus \{x_{t+1}^i\}} \frac{p(x_{t+1}^j|z_t^i)}{p(x_{t+1}^j)} \right] \\ &\approx -\sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \left[1 + \frac{p(x_{t+1}^i)}{p(x_{t+1}^i|z_t^i)} (N-1) \mathbb{E}_{x_{t+1}^j} \frac{p(x_{t+1}^j|z_t^i)}{p(x_{t+1}^j)} \right] \\ &= -\sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \left[1 + \frac{p(x_{t+1}^i)}{p(x_{t+1}^i|z_t^i)} (N-1) \right] \\ &\leq -\sum_{i=1}^{N} \sum_{t=1}^{T-1} \log \left[\frac{p(x_{t+1}^i)}{p(x_{t+1}^i|z_t^i)} N \right] \\ &= \mathcal{I}(x_{t+1}, z_t) - \log N \end{aligned}$$

Therefore, $\mathcal{I}(x_{t+1}, z_t) \geq \log N + \mathcal{J}_{\mathrm{D}}$, which means $\mathcal{J}_{\mathrm{D}}$ is a lower bound for mutual information $\mathcal{I}(x_{t+1}, z_t)$. It is worth noting that the inequality approximation in the above formula becomes more accurate as N increases, so increasing the mini-batch size is helpful for the training of DivER. □

Appendix B.2 The Improved Sample Diversity of DivER

Lemma A2. *Let $\mathcal{D}$ be a finite set of N points in a metric space (X, d). Let $\mathcal{B}$ be a mini-batch of b points sampled from $\mathcal{D}$. The coverage radius of $\mathcal{B}$ for $\mathcal{D}$ is $R(\mathcal{B}) = \sup_{y\in\mathcal{D}} \min_{x\in\mathcal{B}} d(y, x)$. We have $\mathbb{E}[R(\mathcal{B}_{div})] \leq \mathbb{E}[R(\mathcal{B}_{uni})]$, with strict inequality being typical.*

Notation:

- *$\mathcal{B}_{uni}$: A mini-batch of b points selected by random uniform sampling from $\mathcal{D}$.*
- *$\mathcal{C} = \{C_1, \ldots, C_k\}$: A partition of $\mathcal{D}$ into k non-empty clusters.*
- *$N_j = |C_j|$: The number of points in cluster C_j, so $\sum_{j=1}^{k} N_j = N$.*
- *b_j: The number of points sampled from cluster C_j in stratified sampling. Proportional allocation means $b_j \approx b \cdot (N_j/N)$. We assume $\sum_{j=1}^{k} b_j = b$.*
- *$\mathcal{B}_{div}$: A mini-batch formed by sampling b_j points from each C_j (typically uniformly at random within C_j).*

Proof. Let $d_y(\mathcal{B}) = \min_{x\in\mathcal{B}} d(y, x)$. So $R(\mathcal{B}) = \max_{y\in\mathcal{D}} d_y(\mathcal{B})$. A direct proof of $E[\max d_y]$ is complex. Instead, we can argue using the concept of stochastic dominance or by considering the probability of a point being poorly covered. Let $F_{div}(r) = P(R(\mathcal{B}_{div}) \leq r)$ and $F_{uni}(r) = P(R(\mathcal{B}_{uni}) \leq r)$ be the cumulative distribution functions (CDFs) of the coverage radii. If $F_{div}(r) \geq F_{uni}(r)$ for all r (and strictly greater for some r), then R_{div} is stochastically smaller than R_{uni}, which implies $E[R_{div}] \leq E[R_{uni}]$.

Consider any point $y \in \mathcal{D}$. Suppose $y \in C_j$. For stratified sampling (assuming $b_j \geq 1$ for the cluster C_j containing y): The b_j points sampled from C_j provide "local" coverage for points in C_j. The remaining $b - b_j$ points are sampled from other clusters. $P(d_y(\mathcal{B}_{div}) > r) = P(\text{all } b \text{ points in } \mathcal{B}_{div} \text{ are further than } r \text{ from } y)$. Since b_j points are drawn from C_j, if r is larger than the effective radius of C_j achievable with b_j points, this probability becomes small.

For uniform random sampling: Let K_j be the (random) number of points in $\mathcal{B}_{uni}$ that are drawn from cluster C_j. K_j follows a hypergeometric distribution $H(N, N_j, b)$. The expectation $E[K_j] = b \cdot N_j/N \approx b_j$. However, K_j can be 0. $P(K_j = 0) = \frac{\binom{N-N_j}{b}}{\binom{N}{b}}$ if $b \leq N - N_j$. $P(d_y(\mathcal{B}_{uni}) > r) = \sum_{m=0}^{b_j^*} P(d_y(\mathcal{B}_{uni}) > r \mid K_j = m)P(K_j = m)$, where b_j^* is $\min(b, N_j)$. If $P(K_j = 0)$ is substantial (e.g., if C_j is small or b is small relative to N/N_j), and C_j is somewhat isolated from other clusters, then the term $P(d_y(\mathcal{B}_{uni}) > r \mid K_j = 0)$ can be large. This is because y must be covered by points sampled from $\mathcal{D} \setminus C_j$. Stratified sampling (assuming $b_j \geq 1$) ensures $K_j = b_j \geq 1$, eliminating the possibility of cluster C_j being entirely unrepresented in the sample (if it was chosen to receive samples). This directly reduces the probability that $y \in C_j$ is far from its closest point in $\mathcal{B}_{div}$.

More formally, $R(\mathcal{B}) = \max_{y\in\mathcal{D}} d_y(\mathcal{B})$. The uniform random sampling strategy allows for higher variability in the spatial dispersion of points in $\mathcal{B}_{uni}$. Some realizations of $\mathcal{B}_{uni}$ will have points clustered in one region of $\mathcal{D}$, leaving other regions poorly covered, leading to a large $R(\mathcal{B}_{uni})$. Stratified sampling, by forcing b_j samples from each stratum C_j (for j where $b_j \geq 1$), ensures a degree of spatial representativeness. This limits the occurrence of very large values of $R(\mathcal{B}_{div})$. If a sampling strategy S_1 is more likely to produce "extreme" bad configurations than strategy S_2, then we expect $E[f(S_1)] \geq E[f(S_2)]$ if f measures the "badness". The set of points $\mathcal{D}$ can be thought of as a finite probability space where each point has mass $1/N$. The coverage radius $R(\mathcal{B})$ indicates how well the empirical measure induced by $\mathcal{B}$ covers the true measure of $\mathcal{D}$ in a geometric sense. Stratified sampling generally yields empirical measures that are "closer" (in various senses, like reduced variance for means) to the population measure. While a full proof of $F_{div}(r) \geq F_{uni}(r)$ is intricate for general cases, the intuition is that stratified sampling curtails the right tail of the distribution of $R(\mathcal{B})$ by preventing samples that are very poorly spread relative to the

cluster structure. This leads to $E[R(\mathcal{B}_{div})] \leq E[R(\mathcal{B}_{uni})]$. Strict inequality typically holds if clustering is meaningful (strata differ in location/density) and uniform sampling has a non-negligible chance of missing or undersampling certain regions that stratified sampling covers by design. □

Appendix C Pseudocode of Algorithms

Algorithm A1 DivER Sampling

Inputs:
Replay buffer $\mathcal{D}$, mini-batch size N

Cluster samples in $\mathcal{D}$ based on z_T,
$\mathcal{D}' \leftarrow \{\mathbb{D}_i | \mathbb{D}_i$ contains samples from the i-th cluster$\}$,
$\mathcal{B} \leftarrow \emptyset$.
while $\mathcal{D}' \neq \emptyset$ **do**
  $\mathbb{D}_{\min} \leftarrow \arg\min_{\mathbb{D}_i \in \mathcal{D}'} |\mathbb{D}_i|$;
  $N_B = \lceil \frac{|\mathbb{D}_{\min}|}{|\mathcal{D}'|} N \rceil$;
  $\mathbb{B} \leftarrow$ uniformly sample N_B episodes from $\mathbb{D}_{\min}$;
  $\mathcal{B} \leftarrow \mathcal{B} \cup \mathbb{B}$;
  $\mathcal{D}' \leftarrow \mathcal{D}' \setminus \mathbb{D}_{\min}$;
  $N \leftarrow N - N_B$;
end while
Return $\mathcal{B}$

Algorithm A2 Diversified Experience Replay (DivER)

Parameters: MARL framework θ, target network $\theta^- = \theta$, representation model ϕ.

Initialize an empty replay buffer $\mathcal{D}$.
while *training* **do**
  for *episode* $\leftarrow 1$ **to** M **do**
    Initialize $\boldsymbol{E} = \emptyset$.
    for each time step t **do**
      for each agent k **do**
        Select an action based on the policy.
        Store the transition in $\boldsymbol{E}$.
      end for
      Calculate z_t with the representation model.
    end for
    Calculate the final representation z_T,
    Add the episodic data $\boldsymbol{E}$ and vector z_T to $\mathcal{D}$.
  end for
  Sample a mini-batch $\mathcal{B}$ according to Algorithm A1.
  Follow the MARL method to update θ,
  $\phi \leftarrow \phi + \lambda_\phi \nabla_\phi J_D$,
  Update z_T of each sample in $\mathcal{B}$ with new ϕ,
  Update target network $\theta^- \leftarrow \theta$ periodically,
  Update the cluster labels of all data in $\mathcal{D}$ periodically.
end while

Appendix D Environment Details

Appendix D.1 SMAC

SMAC is a simulation environment for research in collaborative multi-agent reinforcement learning (MARL) based on Blizzard's StarCraft II RTS game. It provides various

micro-battle scenarios and also supports customized scenarios for users to test the algorithms. The goal in each scenario is to control different types of ally agents to move or attack to defeat the enemies. The enemies are controlled by a heuristic built-in AI with adjustable difficulty level between 1 to 7. In our experiments, the difficulty of the game AI is set to the highest (the 7th level). The version of StarCraft II is 4.6.2 (B69232) in our experiments, and it should be noted that results from different client versions are not always comparable. Table A1 presents the details of selected scenarios in our experiments.

Table A1. Information of selected challenges.

Challenge	Ally Units	Enemy Units	Type	Level of Difficulty
3s_vs_5z	3 Stalkers	5 Zealots	Homogeneous Asymmetric	Hard
2c_vs_64zg	2 Colossi	64 Zerglings	Homogeneous Asymmetric	Hard
MMM2	1 Medivac 2 Marauders 7 Marines	1 Medivac 3 Marauders 8 Marines	Heterogeneous Asymmetric	Super Hard
6h_vs_8z	6 Hydralisks	8 Zealots	Homogeneous Asymmetric	Super Hard
corridor	6 Zealots	24 Zerglings	Homogeneous Asymmetric	Super Hard
27m_vs_30m	27 Marines	30 Marines	Homogeneous Asymmetric	Super Hard

Appendix D.2 SMACv2

SMACv2 [31] is proposed to address SMAC's lack of stochasticity. In SMACv2, the sight range of the agents is narrowed, and the attack ranges of different unit types are no longer the same. The team compositions and agent start positions are generated randomly at the beginning of each episode. These modifications make SMACv2 extremely challenging. It is worth noting that the disparity between the lineups of the two sides can be substantial in some episodes due to the randomness in initialization. In these episodes, the outcomes are almost determined at the initialization stage but are little affected by the policies learned by the algorithms. We sifted out such episodes during testing to prevent blurring the gap between algorithms. The version of the StarCraft II engine is also 4.6.2 (B69232) in our experiments.

Table A2. The hyperparameter settings of DivER.

Description	Value
Type of value mixer	QMIX
Dimension of trajectory embedding	32
Dimension of hidden states in RNN	64
Dimension of the mixing network	32
Dimension of hypernetworks	64
Batch size	128
Trajectories sampled per run	4
Replay buffer size	2500
Discount factor γ	0.99
Probability of random action (ϵ)	1.0$\sim$0.05
Anneal time for ϵ	100000
Type of optimizer	Adam
Clustering algorithm	K-Means
Number of clusters	8
Learning rate for DivER	0.001
Learning rate for MARL framework	0.001
Target network update interval	200

Appendix E Implementation Details

Appendix E.1 Settings of Hyperparameters

We list the hyperparameters of the DivER in Table A2. The hyperparameters of the baselines in our experiments remain the same as their official implementations.

Appendix E.2 Experiments Compute Resources

We conducted our experiments on a platform with 2 Intel(R) Xeon(R) Platinum 8280 CPU 2.70GHz processors, each with 26 cores. Besides, we use a GeForce RTX 3090 GPU to facilitate the training procedure. The time of execution varies by scenario.

Appendix F Further Experiments

The extra model parameters introduced by DivER. Since DivER is implemented on the basis of QMIX, one may doubt if the improvement of DivER is due to the increased model size. As Figure 3 shows, the structure of DivER is relatively independent and does not interfere with the decision-making process of QMIX. We further expand the model size of QMIX (denoted as QMIX_Large) to be comparable with that of DivER, and the experiment results are displayed in Figure A1. As we can see, an increase in model size does not yield a significant performance improvement. Therefore, we can deduce that the methodology of DivER rather than the increased parameters is the pivotal factor contributing to its superiority.

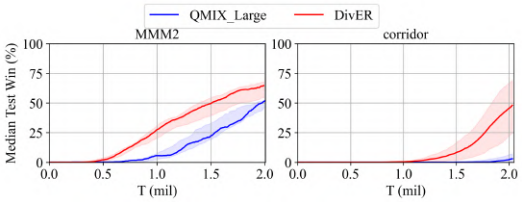


Figure A1. The ablation study results about the model size.

References

1. Xu, Z.; Lyu, Y.; Pan, Q.; Hu, J.; Zhao, C.; Liu, S. Multi-vehicle flocking control with deep deterministic policy gradient method. In Proceedings of the 2018 IEEE 14th International Conference on Control and Automation (ICCA). IEEE, 2018, pp. 306–311.
2. Gu, S.; Kuba, J.G.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; Yang, Y. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence* **2023**, *319*, 103905.
3. Shamsoshoara, A.; Khaledi, M.; Afghah, F.; Razi, A.; Ashdown, J. Distributed cooperative spectrum sharing in uav networks using multi-agent reinforcement learning. In Proceedings of the 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2019, pp. 1–6.
4. Zhang, Z.; Han, S.; Wang, J.; Miao, F. Spatial-temporal-aware safe multi-agent reinforcement learning of connected autonomous vehicles in challenging scenarios. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023, pp. 5574–5580.
5. Wu, T.; Zhou, P.; Liu, K.; Yuan, Y.; Wang, X.; Huang, H.; Wu, D.O. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology* **2020**, *69*, 8243–8256.
6. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* **2017**, *30*.
7. 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. *arXiv preprint arXiv:1706.05296* **2017**.
8. 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 International conference on machine learning. PMLR, 2018, pp. 4295–4304.
9. Mahajan, A.; Rashid, T.; Samvelyan, M.; Whiteson, S. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems* **2019**, *32*.

10. Li, C.; Wang, T.; Wu, C.; Zhao, Q.; Yang, J.; Zhang, C. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* **2021**, *34*, 3991–4002.
11. Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; Zhang, C. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523* **2020**.
12. Zhou, G.; Xu, Z.; Zhang, Z.; Fan, G. SORA: Improving Multi-agent Cooperation with a Soft Role Assignment Mechanism. In Proceedings of the International Conference on Neural Information Processing. Springer, 2023, pp. 319–331.
13. Zeng, X.; Peng, H.; Li, A. Effective and Stable Role-Based Multi-Agent Collaboration by Structural Information Principles. *arXiv preprint arXiv:2304.00755* **2023**.
14. Rashid, T.; Farquhar, G.; Peng, B.; Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems* **2020**, *33*, 10199–10210.
15. Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; Zhang, C. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062* **2020**.
16. Yang, Y.; Hao, J.; Liao, B.; Shao, K.; Chen, G.; Liu, W.; Tang, H. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939* **2020**.
17. Mei, Y.; Zhou, H.; Lan, T. Projection-Optimal Monotonic Value Function Factorization in Multi-Agent Reinforcement Learning. In Proceedings of the AAMAS, 2024, pp. 2381–2383.
18. Mei, Y.; Zhou, H.; Lan, T.; Venkataramani, G.; Wei, P. Mac-po: Multi-agent experience replay via collective priority optimization. *arXiv preprint arXiv:2302.10418* **2023**.
19. Schaul, T. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952* **2015**.
20. Oliehoek, F.A.; Amato, C. *A concise introduction to decentralized POMDPs*; Springer, 2016.
21. Sener, O.; Savarese, S. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489* **2017**.
22. Loshchilov, I.; Hutter, F. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343* **2015**.
23. Jiang, A.H.; Wong, D.L.K.; Zhou, G.; Andersen, D.G.; Dean, J.; Ganger, G.R.; Joshi, G.; Kaminksy, M.; Kozuch, M.; Lipton, Z.C.; et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762* **2019**.
24. Katharopoulos, A.; Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. In Proceedings of the International conference on machine learning. PMLR, 2018, pp. 2525–2534.
25. Pleiss, G.; Zhang, T.; Elenberg, E.; Weinberger, K.Q. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems* **2020**, *33*, 17044–17056.
26. Paul, M.; Ganguli, S.; Dziugaite, G.K. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems* **2021**, *34*, 20596–20607.
27. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* **2018**.
28. Gutmann, M.; Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Proceedings of the Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.
29. Fang-Ming, B.; Wei-Kui, W.; Long, C. DBSCAN: Density-based spatial clustering of applications with noise. *Journal of Nanjing University(Natural Sciences)* **2012**, *48*, 491–498.
30. 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 preprint arXiv:1902.04043* **2019**.
31. Ellis, B.; Cook, J.; Moalla, S.; Samvelyan, M.; Sun, M.; Mahajan, A.; Foerster, J.N.; Whiteson, S. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning, 2023, [arXiv:cs.LG/2212.07489].
32. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.
33. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* **2017**, *12*, e0172395.
34. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.; Yi, Y. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. *CoRR* **2019**, *abs/1905.05408*, [1905.05408].
35. Iqbal, S.; Sha, F. Actor-Attention-Critic for Multi-Agent Reinforcement Learning, 2019, [arXiv:cs.LG/1810.02912].
36. Su, J.; Adams, S.; Beling, P. Value-decomposition multi-agent actor-critics. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2021, Vol. 35, pp. 11352–11360.
37. Gogineni, K.; Wei, P.; Lan, T.; Venkataramani, G. Scalability Bottlenecks in Multi-Agent Reinforcement Learning Systems. *arXiv preprint arXiv:2302.05007* **2023**.
38. Lin, L.J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* **1992**, *8*, 293–321.
39. Lee, S.Y.; Sungik, C.; Chung, S.Y. Sample-efficient deep reinforcement learning via episodic backward update. *Advances in neural information processing systems* **2019**, *32*.
40. Brittain, M.; Bertram, J.; Yang, X.; Wei, P. Prioritized sequence experience replay. *arXiv preprint arXiv:1905.12726* **2019**.

41. Zha, D.; Lai, K.H.; Zhou, K.; Hu, X. Experience replay optimization. *arXiv preprint arXiv:1906.08387* **2019**.
42. Sun, P.; Zhou, W.; Li, H. Attentive experience replay. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2020, Vol. 34, pp. 5900–5907.
43. Novati, G.; Koumoutsakos, P. Remember and forget for experience replay. In Proceedings of the International Conference on Machine Learning. PMLR, 2019, pp. 4851–4860.
44. Liu, X.H.; Xue, Z.; Pang, J.; Jiang, S.; Xu, F.; Yu, Y. Regret minimization experience replay in off-policy reinforcement learning. *Advances in neural information processing systems* **2021**, *34*, 17604–17615.
45. Oh, Y.; Shin, J.; Yang, E.; Hwang, S.J. Model-augmented prioritized experience replay. In Proceedings of the International Conference on Learning Representations, 2022.
46. Fan, S.; Song, G.; Yang, B.; Jiang, X. Prioritized experience replay in multi-actor-attention-critic for reinforcement learning. In Proceedings of the Journal of Physics: Conference Series. IOP Publishing, 2020, Vol. 1631, p. 012040.
47. Wang, Y.; Zhang, Z. Experience selection in multi-agent deep reinforcement learning. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019, pp. 864–870.
48. Ahilan, S.; Dayan, P. Correcting experience replay for multi-agent communication. *arXiv preprint arXiv:2010.01192* **2020**.