Value-Decomposition Multi-Agent Actor-Critics

Jianyu Su, Stephen Adams, Peter A. Beling
University of Virginia
151 Engineer's Way
Charlottesville, Virginia, 22904
{js9wv, sca2c, pb3a}@virginia.edu

Abstract

The exploitation of extra state information has been an active research area in multi-agent reinforcement learning (MARL). QMIX represents the joint action-value using a non-negative function approximator and achieves the best performance on the StarCraft II micromanagement testbed, a common MARL benchmark. However, our experiments demonstrate that, in some cases, QMIX performs sub-optimally with the A2C framework, a training paradigm that promotes algorithm training efficiency. To obtain a reasonable trade-off between training efficiency and algorithm performance, we extend value-decomposition to actor-critic methods that are compatible with A2C and propose a novel actor-critic framework, value-decomposition actor-critic (VDAC). We evaluate VDAC on the StarCraft II micromanagement task and demonstrate that the proposed framework improves median performance over other actor-critic methods. Furthermore, we use a set of ablation experiments to identify the key factors that contribute to the performance of VDAC.

Many complex sequential decision making problems that involve multiple agents can be modeled as multi-agent reinforcement learning (MARL) problems, e.g. the coordination of semi-autonomous or fully autonomous vehicles (Hu et al. 2019) and the coordination of machines in a product line (Choo, Adams, and Beling 2017). A fully centralized controller that applies single-agent reinforcement learning will suffer from the exponential growth of the action space with the number of agents in the system. Learning decentralized policies that condition on the local observation history of individual agents is a viable way to attenuate this problem. Furthermore, partial observability and communication constraints, two common obstacles in multi-agent settings, also necessitate the use of decentralized policies.

In a laboratory or simulated setting, decentralized policies can be learned in a centralized fashion via enabling communication among agents or granting access to additional global state information. This *centralized training and decentralized execution* (CTDE) paradigm has attracted the attention of researchers. However, it remains an open research question how to best exploit centralized training. In particular, it is not obvious how to utilize joint action-value or global state value to train decentralized policies.

Breakthroughs in Q-learning have been made using joint action-value factorization techniques. *Value-decomposition networks* (VDN) represent joint action-value as a summation of local action-value conditioned on individual agents' local observation history (Sunehag et al. 2017). In (Rashid et al. 2018), a more general case of VDN is proposed using a mixing network that approximates a broader class of monotonic functions to represent joint action-values called QMIX. In (Son et al. 2019), a more complex factorization framework three modules, called QTRAN, is introduced and shown to have good performance on a range of cooperative tasks. While QMIX reports the best performance on the StarCraft micromanagement testbed (Samvelyan et al. 2019), we find that QMIX, in some StarCraft II compositions, has issues learning good policies that can consistently defeat enemies when using the A2C training paradigm (Mnih et al. 2016), which was originally introduced to enable algorithms to be executed efficiently.

On the other hand, on-policy actor-critic methods, such as *counterfactual multi-agent* (COMA) (Foerster et al. 2018), can leverage the A2C framework to improve training efficiency at the cost of performance. (Samvelyan et al. 2019) point out that there is a performance gap between the state-of-the-art actor-critic method, COMA, and QMIX on the StarCraft II micromanagement testbed.

To bridge the gap between multi-agent Q-learning and multi-agent actor-critic methods, as well as offer a reasonable trade-off between training efficiency and algorithm performance, we propose a novel actor-critic framework called value-decomposition actor-critic (VDAC). Let $V^a, \forall a \in \{1, \dots, n\}$ denote the local state value that is conditioned on agent a's local observation, and let V_{tot} denote the global state-value that is conditioned on the true state of the environment. VDAC takes an actor-critic approach but adds local critics, which share the same network with the actors and estimate the local state values V^a. The central critic learns the global state value V_{tot}. The policy is trained by following a gradient dependent on the central critic. Further, we examines two approaches for calculating V_{tot}.

VDAC is based on three main ideas. First, unlike QMIX, VDAC is compatible with a A2C training framework that enables game experience to be sampled efficiently. This is due to the fact that multiple games are rolled out independently during training. Second, similar to QMIX, VDAC enforces the following relationship between local state-values V^a and the global state-value V_{tot}:

$$\frac{\partial V_{tot}}{\partial V^a} \geq 0, \quad \forall a \in \{1, \ldots, n\}. \qquad (1)$$

This idea is related to *difference rewards* (Wolpert and Tumer 2002), in which each agent learns from a shaped reward that compares the global reward to the reward received when that agent's action is replaced with a default action. *Difference rewards* require that any action that improves an agent's local reward also improves the global reward, which implies the monotonic relationship between shaped local rewards and the global reward. While COMA (also inspired by *difference rewards*) focuses on customizing shaped rewards r^a from the global reward r_{tot} in a pairwise fashion $r^a = f(r_{tot})$, VDAC represents the global reward by all agents' shaped rewards $r_{tot} = f(r^1, ..., r^n)$. Third, VDAC is trained by following a rather simple policy gradient that is calculated from a temporal-difference (TD) advantage. We theoretically demonstrate that the proposed method is able to converge to a local optimum by following this policy gradient. Despite the fact that TD advantage policy gradients and COMA gradients are both unbiased estimates of a vanilla multi-agent policy gradients, our empirical study favors TD advantage policy gradients over COMA policy gradients.

This study strives to answer the following research questions:

- **Research question 1**: Is the TD advantage gradient sufficient to optimize multi-agent actor-critics when compared to a COMA gradient?
- **Research question 2**: Does applying state-value factorization improve the performance of actor-critics?
- **Research question 3**: Does VDAC provide a reasonable trade-off between training efficiency and algorithm performance when compared to QMIX?
- **Research question 4**: What are the factors that contribute to the performance of the proposed VDAC?

Related Work

MARL has benefited from recent developments in deep reinforcement learning, with the field moving away from tabular methods (Bu et al. 2008) to deep neural networks (Foerster et al. 2018). Our work is related to recent advances in CTDE deep multi-agent reinforcement learning.

The degree of training centralization varies in the literature on MARL. *Independent Q-learning* (IQL) (Tan 1993) and its deep neural network counterpart (Tampuu et al. 2017) train an independent Q-learning model for each agent. Those that attempt to directly learn decentralized policies often suffer from the non-stationarity of the environment induced by agents simultaneously learning and exploring. (Foerster et al. 2017; Usunier et al. 2016) attempt to stabilize learning under the decentralized training paradigm. (Gupta, Egorov, and Kochenderfer 2017) propose a training paradigm that alternates between centralized training with global rewards and decentralized training with shaped rewards.

Centralized methods, by contrast, naturally avoid the non-stationary problem at the cost of scalability. COMA (Foerster et al. 2016), takes advantage of CTDE, where actors are updated by following policy gradients that are tailored by their contributions to the system. *Multi-agent deep deterministic policy gradient* (MADDPG) (Lowe et al. 2017) extends *deep deterministic policy gradient* (DDPG) (Lillicrap et al. 2015) to mitigate the issue of high variance gradient estimates exacerbated in multi-agent settings. Based on MADDPG, (Wei et al. 2018) propose multi-agent soft Q-learning in continuous action spaces to tackle the issue of *relative overgeneralization*. *Probabilistic recursive reasoning* (Wen et al. 2019) is a method that uses a probabilistic recursive reasoning policy gradient that enables agents to recursively reason what others believe about their own beliefs.

More recently, value-based methods, which lie between the extremes of IQL and COMA, have shown great success in solving complex multi-agent problems. VDN (Sunehag et al. 2017), which represents joint-action value function as a summation of local action-value function, allows for centralized learning. However, it does not make use of extra state information. QMIX (Rashid et al. 2018) utilizes a non-negative mixing network to represent a broader class of value-decomposition functions. Furthermore, additional state information is captured by hypernetworks that output parameters for the mixing network. QTRAN (Son et al. 2019) is a generalized factorization method that can be applied to environments that are free from structural constraints. Other works, such as CommNet (Foerster et al. 2016), TarMAC (Das et al. 2019), ATOC (Jiang and Lu 2018), MAAC (Iqbal and Sha 2019), CCOMA (Su, Adams, and Beling 2020) and BiCNet(Peng et al. 2017) exploit inter-agent communication.

The proposed VDAC method is similar to QMIX and VDN in that it utilizes value-decomposition. However, VDAC is a policy-based method that decomposes global state-values whereas QMIX and VDN, which decompose global action-values, belong to the Q-learning family. (Nguyen, Kumar, and Lau 2018) address credit-assignment issue, however, under a different MARL setting, $\mathbb{C}$Dec-POMDP. COMA, which is also a policy gradient method inspired by *difference rewards* and has been tested on StarCraft II micromanage games, represents the work most closely related to this paper.

Background

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs): Consider a fully cooperative multi-agent task with n agents. Each agent identified by $a \in A \equiv \{1, \ldots, n\}$ takes an action $u^a \in U$ simultaneously at every timestep, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^a, \forall a \in \{1, \ldots, n\}$. The environment has a true state $s \in S$, a transition probability function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to S$, and a global reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$. In the partial observation setting, each agent draws an observations $z \in Z$ from the observation function $O(S, A) : S \times A \to Z$. Each agent conditions a stochastic policy $\pi(u^a|\tau^a) : T \times U \to [0, 1]$ on its observation-action history $\tau^a \in T \equiv Z \times U$. Throughout this paper, quantities in bold represent joint quantities over agents, and bold quantities with the superscript $-a$ denote joint quantities over agents other than a given agent a. MARL agents aim to maximize the discounted return $R_t = \sum_{l=1}^{\infty} \gamma^l r_{t+l}$. The joint value function $V^{\pi}(s_t) = \mathbb{E}[R_t | s_t = s]$ is the expected return for following the joint policy π from state s. The value-action function $Q^{\pi}(s, \mathbf{u}) = \mathbb{E}[R_t | s_t = s, \mathbf{u}]$ defines the expected return for selecting joint action $\mathbf{u}$ in state s and following the joint policy π.

Single-Agent Policy Gradient Algorithms: Policy gradient methods adjust the parameters θ of the policy in order to maximize the objective $J(\theta) = \mathbb{E}_{s \sim p^{\pi}, u \sim \pi}[R(s, u)]$ by taking steps in the direction of $\nabla J(\theta)$. The gradient with respect to the policy parameters is $\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi}[\nabla_{\theta} \log \pi_{\theta}(a|s) Q_{\pi}(s, u)]$, where p^{π} is the state transition by following policy π, and $Q_{\pi}(s, u)$ is an action-value.

To reduce variations in gradient estimates, a baseline b is introduced. In actor-critic approaches (Konda and Tsitsiklis 2000), an actor is trained by following gradients that are dependent on the critic. This yields the advantage function $A(s_t, u_t) = Q(s_t, u_t) - b(s_t)$, where $b(s_t)$ is the baseline ($V(s_t)$ or another constant is commonly used as the baseline). TD error $r_t + \gamma V(s_{t+1}) - V(s_t)$, which is an unbiased estimate of $Q(s_t, u_t)$, is a common choice for advantage functions. In practice, a TD error that utilizes an n-step return $\sum_{i=0}^{k-1} \gamma^i r_t + \gamma^k V(s_{t+k}) - V(s_t)$ yields good performance (Mnih et al. 2016).

Multi-Agent Policy Gradient (MAPG) Algorithms: Multi-agent policy gradient methods are extensions of policy gradient algorithms with a policy $\pi_{\theta_a}(u^a | o^a), a \in \{1, \cdots, n\}$. Compared with policy gradient methods, MAPG faces the issues of high variance gradient estimates (Lowe et al. 2017) and credit assignment (Foerster et al. 2018). Perhaps the simplest multi-agent gradient can be written as:

$$\nabla_{\theta} J = \mathbb{E}_{\pi} \left[\sum_{a} \nabla_{\theta} \log \pi_{\theta}(u^a | o^a) Q_{\pi}(s, \mathbf{u}) \right]. \quad (2)$$

Multi-agent policy gradients in the current literature often take advantage of CTDE by using a central critic to obtain extra state information s, and avoid using the vanilla multi-agent policy gradients (Equation 2) due to high variance. For instance, (Lowe et al. 2017) utilize a central critic to estimate $Q(s, (a_1, \ldots, a_n))$ and optimize parameters in actors by following a multi-agent DDPG gradient, which is derived from Equation 2:

$$\nabla_{\theta_a} J = \mathbb{E}_{\pi}[\nabla_{\theta_a} \pi(u^a | o^a) \nabla_{u^a} Q_{u^a}(s, \mathbf{u})|_{u^a = \pi(o^a)}]. \quad (3)$$

Unlike most actor-critic frameworks, (Foerster et al. 2018) claim to solve the credit assignment issue by applying the following counterfactual policy gradients:

$$\nabla_{\theta} J = \mathbb{E}_{\pi} \left[\sum_{a} \nabla_{\theta} \log \pi(u^a | \tau^a) A^a(s, \mathbf{u}) \right], \quad (4)$$

where $A^a(s, \mathbf{u}) = Q_{\pi}(s, \mathbf{u}) - \sum_{u^a} \pi_{\theta}(u^a | \tau^a) Q_{\pi}^a(s, (\mathbf{u}^{-a}, u^a))$ is the counterfactual advantage for agent a. Note that (Foerster et al. 2018) argue that the COMA gradients provide agents with tailored gradients, thus achieving credit assignment. At the same time, they also prove that COMA is a variance reduction technique.

Methods

In addition to the previously outlined research questions, our goal in this work is to derive RL algorithms under the following constraints: (1) the learned policies are conditioned on agents' local action-observation histories (the environment is modeled as Dec-POMDP), (2) a model of the environment dynamics is unknown (i.e. the proposed framework is task-free and model-free), (3) communication is not allowed between agents (i.e. we do not assume a differentiable communication channel such as (Das et al. 2019)), and (4) the framework should enable parameter sharing among agents (namely, we do not train different models for each agent as is done in (Tan 1993)). A method that met the above criteria would constitute a general-purpose multi-agent learning algorithm that could be applied to a range of cooperative environments, with or without communication between agents. Hence, the following methods are proposed.

Naive Central Critic Method

A naive central critic (naive critic) is proposed to answer the first research question: is a simple policy gradient sufficient to optimize multi-agent actor-critic methods. Naive critic's central critic shares a similar structure with COMA's critic. It takes (s_t, u_{t-1}) as the input and outputs $V(s)$. Actors follow a rather simple policy gradient, a TD advantage policy gradient that is common in the RL literature given by:

$$\nabla_{\theta} J = \mathbb{E}_{\pi} \left[\sum_{a} \nabla_{\theta} \log \pi(u^a | \tau^a) \big(Q(s, \mathbf{u}) - V(s) \big) \right], \quad (5)$$

where $Q(s, \mathbf{u}) = r + \gamma V(s')$. In the next section, we will demonstrate that policy gradients taking the form of Equation 5 under our proposed actor-critic frameworks are also unbiased estimates of the naive multi-agent policy gradients. The pseudo code is listed in Appendix.

Value Decomposition Actor-Critic

Difference rewards enable agents to learn from a shaped reward $D^a = r(s, \mathbf{u}) - r(s, (\mathbf{u}^{-a}, c^a))$ that is defined by a reward change incurred by replacing the original action u^a with a default action c^a. Any action taken by agent a that improves D^a also improves the global reward $r(s, \mathbf{u})$ since the second term in the difference reward equation does not depend on u^a. Therefore, the global reward $r(s, \mathbf{u})$ is monotonically increasing with D^a. Inspired by *difference rewards*, we propose to decompose state value $V_{tot}(s)$ into local states $V^a(o^a)$ such that the following relationship holds:

$$\frac{\partial V_{tot}}{\partial V^a} \geq 0, \quad \forall a \in \{1, \ldots, n\}. \quad (6)$$

With Equation 6 enforced, given that the other agents stay at the same local states by taking $\mathbf{u}^{-a}$, any action u^a that leads agent a to a local state o^a with a higher value will also improve the global state value V_{tot}.

Two variants of value-decomposition that satisfy Equation 6, VDAC-sum and VDAC-mix, are studied.

Table 1: Actor-Critics studied.

Algorithm	Central Critic	Value Decomposition	Policy Gradients
IAC (Foerster et al. 2018)	No	-	TD advantage
VDAC-sum	Yes	Linear	TD advantage
VDAC-mix	Yes	Non-linear	TD advantage
Naive Critic	Yes	-	TD advantage
COMA (Foerster et al. 2018)	Yes	-	COMA advantage

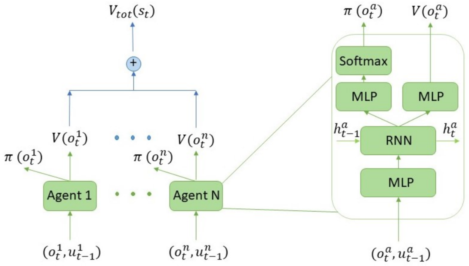


Figure 1: VDAC-sum

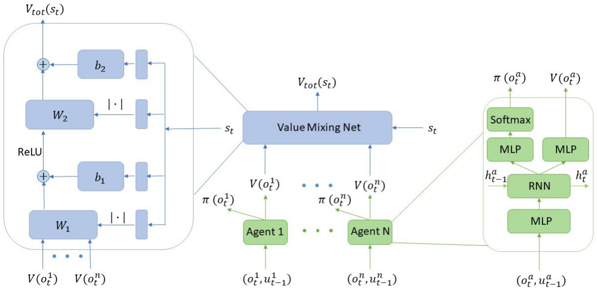


Figure 2: VDAC-vmix

VDAC-sum In VDAC-sum, the total state value $V_{tot}(s)$ is a summation of local state values $V^a(o^a)$: $V_{tot}(s) = \sum_a V^a(o^a)$. This linear representation is sufficient to satisfy Equation 6. VDAC-sum's structure is shown in Figure 1. Note that the actor outputs both $\pi_\theta(o^a)$ and $V_{\theta_v}(o^a)$. This is done by sharing non-output layers between distributed critics and actors. In this paper, θ_v denotes the distributed critics' parameters and θ denotes the actors' parameters for generality. The distributed critic is optimized by minibatch gradient descent to minimize the following loss:

$$\begin{aligned} L_t(\theta_v) &= \Big(y_t - V_{tot}(s_t)\Big)^2 \\ &= \Big(y_t - \sum_a V_{\theta_v}(o_t^a)\Big)^2, \end{aligned} \tag{7}$$

where $y_t = \sum_{i=t}^{k-t-1} \gamma^i r_i + \gamma^{(k-t)} V_{tot}(s_k)$ is bootstrapped from the last state s_k, and k is upper-bounded by T.

The policy network is trained using the following policy gradient $g = \mathbb{E}_\pi[\sum_a \nabla_\theta \log \pi(u^a|\tau^a) A(s, \mathbf{u})]$, where $A(s, \mathbf{u}) = r + \gamma V(s') - V(s)$ is a simple TD advantage. Similar to independent actor-critic (IAC), VDAC-sum does not make full use of CTDE in that it does not incorporate state information during training. Furthermore, it can only represent a limited class of centralized state-value functions.

VDAC-mix To generalize the representation to a larger class of monotonic functions, we utilize a feed-forward neural network that takes input as local state values $V_\theta(o^a), \forall a \in \{1, \dots, n\}$ and outputs the global state value V_{tot}. To enforce Equation 6, the weights (not including bias) of the network are restricted to be non-negative. This allows the network to approximate any monotonic function arbitrarily well (Dugas et al. 2009).

The weights of the mixing network are produced by separate hypernetworks (Ha, Dai, and Le 2016). Following the practice in QMIX (Rashid et al. 2018), each hypernetwork takes the state s as an input and generates the weights of one layer of the mixing network. Each hypernetwork consists of a single linear layer. An absolute activation function is utilized in the hypernetwork to ensure that the outputted weights are non-negative. The biases are not restricted to being non-negative. Hence, the hypernetworks that produce the biases do not apply an absolute non-negative function. The final bias is produced by a 2-layer hypernetwork with a ReLU activation function following the first layer. Finally, the hypernetwork outputs are reshaped into a matrix of appropriate size. Figure 2 illustrates the mixing network and the hypernetworks.

The whole mixing network structure (including hypernetworks) can be seen as a central critic. Unlike critics in (Foerster et al. 2018), this critic takes local state values $V^a(o^a), \forall a \in \{1, \dots, n\}$ as additional inputs besides global state s. Similar to VDAC-sum, the distributed critics are optimized by minimizing the following loss:

$$\begin{aligned} L_t(\theta_v) &= \Big(y_t - V_{tot}(s_t)\Big)^2 \\ &= \Big(y_t - f_{mix}(V_{\theta_v}(o_t^1), \dots, V_{\theta_v}(o_t^n))\Big)^2, \end{aligned} \tag{8}$$

where f_{mix} denotes the mixing network. Let θ^c denote parameters in the hypernetworks. The central critic is optimized by minimizing the same loss $L_t(\theta^c) = (y_t - V_{tot}(s_t))$. The policy network is updated by following the same policy gradient in Equation 5. The pseudo code is provided in Appendix.

Convergence of VDAC frameworks (Foerster et al. 2018) establish the convergence of COMA based on the convergence proof of single-agent actor-critic algorithms (Konda and Tsitsiklis 2000; Sutton et al. 2000). In the same manner, we utilize the following lemma to substantiate the convergence of VDACs to a locally optimal policy.

Lemma 1: For a VDAC algorithm with a compatible TD(1) critic following a policy gradient

$$g_k = \mathbb{E}_\pi\left[\sum_a \nabla_{\theta_k} \log \pi(u^a|\tau^a) A(s, \mathbf{u}))\right],$$

at each iteraction k, $\lim\inf_k ||\nabla J|| = 0 \quad w.p.1.$

Proof The VDAC gradient is given by:

$$g = \mathbb{E}_\pi\left[\sum_a \nabla_\theta \log \pi(u^a|\tau^a) A(s, \mathbf{u})\right], \tag{9}$$

$A(s, \mathbf{u}) = Q(s, \mathbf{u}) - V_{tot}(s)$. We first consider the expected distribution of the baseline V_{tot}:

$$\begin{aligned} g_b &= -\mathbb{E}_\pi\left[\sum_a \nabla_\theta \log \pi(u^a|\tau^a) V_{tot}(s)\right] \\ &= -\mathbb{E}_\pi\left[\nabla_\theta \log \prod_a \pi(u^a|\tau^a) V_{tot}(s)\right], \end{aligned} \tag{10}$$

where the distribution $\mathbb{E}_\pi$ is with respect to the state-action distribution induced by the joint policy π. Writing the joint policy as a product of independent actors $\pi(\mathbf{u}|s) = \prod_a \pi(u^a|\tau^a)$. The total value does not depend on agent actions and is given by $V_{tot}(s) = f(V_1(o^1), \ldots, V_n(o^n))$ where f is a non-negative function. This yields a single-agent actor-critic baseline: $g_b = -\mathbb{E}_\pi[\nabla_\theta \log \pi(\mathbf{u}|s) V_{tot}(s)]$.

Now let $d^{\pi(s)}$ be the discounted ergodic state distribution as defined by (Sutton et al. 2000):

$$\begin{aligned} g_b &= -\sum_s d^{\pi(s)} \sum_{\mathbf{u}} \nabla_\theta \log \pi(\mathbf{u}|s) V_{tot}(s) \\ &= -\sum_s d^{\pi(s)} V_{tot}(s) \nabla_\theta \sum_{\mathbf{u}} \log \pi(\mathbf{u}|s) \\ &= -\sum_s d^{\pi(s)} V_{tot}(s) \nabla_\theta 1 \\ &= 0 \end{aligned} \tag{11}$$

The remainder of the gradient is given by:

$$\begin{aligned} g &= \mathbb{E}_\pi\left[\sum_a \nabla_\theta \log \pi(u^a|\tau^a) Q(s, \mathbf{u})\right] \\ &= \mathbb{E}_\pi\left[\nabla_\theta \log \prod_a \pi(u^a|\tau^a) Q(s, \mathbf{u})\right], \end{aligned} \tag{12}$$

which yields a standard single-agent actor-critic policy gradient $g = \mathbb{E}_\pi[\nabla_\theta \log \pi(\mathbf{u}|s) Q(s, \mathbf{u})]$. (Konda and Tsitsiklis 2000) establish that an actor-critic that follows this gradient converges to a local maximum of the expected return J^π, subject to assumptions included in their paper.

In the naive critic framework, $V_{tot}(s)$ is evaluated by the central critic and does not depend on agent actions. Hence, by following the same proof in Equation 11, we can show that the expectation of naive critic baseline is also 0, thus proves naive critic also converges to a locally optimal policy.

Experiments

In this section, we benchmark VDACs against the baseline algorithms listed in Table 1 on a standardized decentralised StarCraft II micromanagement environment, SMAC (Samvelyan et al. 2019). SMAC consists of a set of StarCraft II micromanagement games that aim to evaluate how well independent agents are able to cooperate to solve complex tasks. In each scenario, algorithm-controlled ally units fight against enemy units controlled by the built-in game AI. An episode terminates when all units of either army have died or when the episode reached the pre-defined time limit. A game is counted as a win only if enemy units are eliminated. The goal is to maximize the win rate.

We consider the following maps in our experiments: 2s_vs_1sc, 2s3z, 3s5z, 1c3s5z, 8m, and bane_vs_bane. Note that all algorithms are trained under A2C framework where 8 episodes are rolled out independently during the training. Refer to Appendix for training details and map configuration.

We perform the following ablations to answer the corresponding research questions:

Ablation 1 Is the TD advantage gradient sufficient to optimize multi-agent actor-critics? The comparison between the naive critic and COMA will demonstrate the effectiveness of TD advantage policy gradients because the only significant difference between those two methods is that the naive critic follows a TD advantage policy gradient whereas COMA follows the COMA gradient (Equation 4).

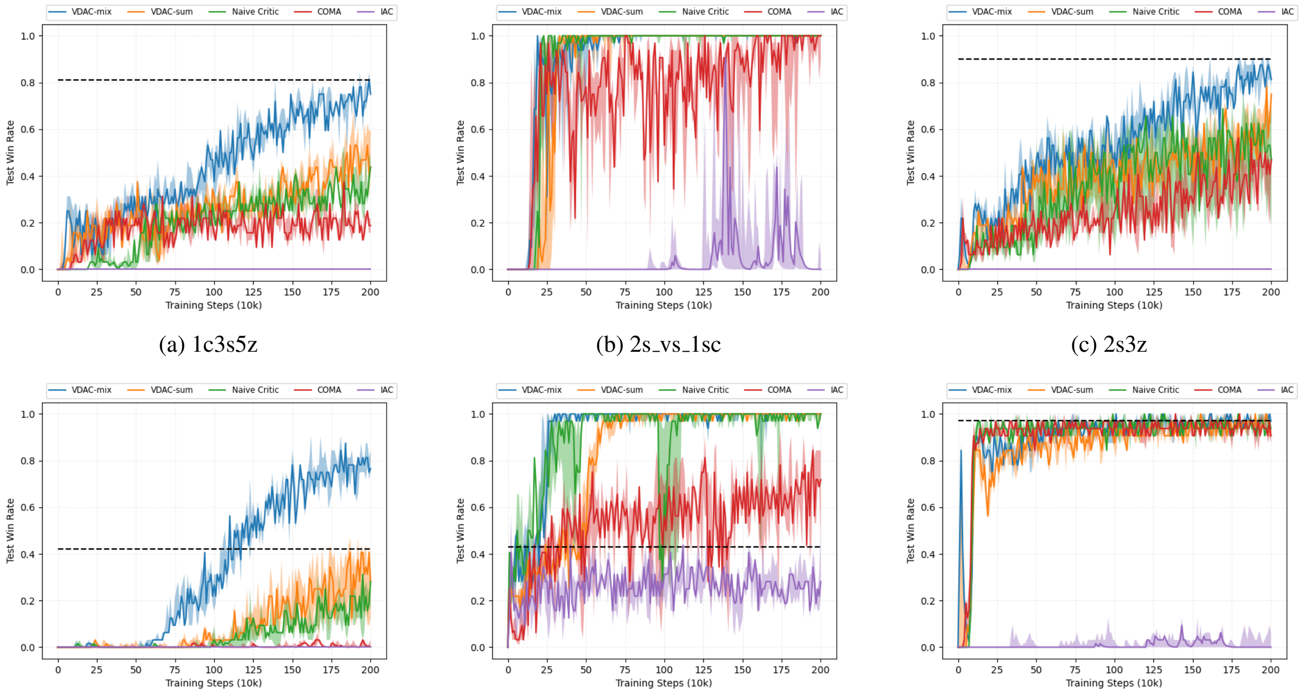
Ablation 2 Does applying state-value factorization improve the performance of actor-critic methods? VDAC-sum and IAC, both of which do not have access to extra state information, shares an identical structure. The only difference is that VDAC-sum applies a simple state-value factorization where the global state-value is a summation of local state values. The comparison between VDAC-sum and IAC will reveal the necessity of applying state-value factorization.

Ablation 3 Compared with QMIX, does VDAC provide a reasonable trade-off between training efficiency and algorithm performance? We train VDAC and QMIX under A2C training paradigm, which is proposed to promote training efficiency, and compare their performance.

Ablation 4 What are the factors that contribute to the performance of the proposed VDAC? We investigate the necessity of non-linear value-decomposition by removing the non-linear activation function in the mixing network. The resulting algorithm is called VDAC-mix (linear) and can be seen as VDAC-sum with access to extra state information.

Overall Results

As suggested in (Samvelyan et al. 2019), our main evaluation metric is the median win percentage of evaluation



(a) 1c3s5z

(b) 2s_vs_1sc

(c) 2s3z

(d) 3s5z

(e) bane_vs_bane

(f) 8m

Figure 3: Overall results: Win rates on a range of SC mini-games. Black dash line represents heuristic AI's performance

episodes as a function of environment steps observed over the 200k training steps. Specifically, the performance of an algorithm is estimated by periodically running a fixed number of evaluation episodes (in our implementation, 32) during the course of training, with any exploratory behaviours disabled. The median performance as well as the 25-75% percentiles are obtained by repeating each experiment using 5 independent training runs. Figure 3 demonstrates the comparison among actor-critics across 6 different maps.

In all scenarios, IAC fails to learn a policy that consistently defeats the enemy. In addition, its performance across training steps is highly unstable due to the non-stationarity of the environment and its lack of access to extra state information.

Noticeably, VDAC-mix consistently achieves the best performance across all tasks. On easy games (i.e, 8m), all algorithms generally perform well. This is due to the fact that a simple strategy implemented by the heuristic AI to attack the nearest enemies is sufficient to win. In harder games such as 3s5z and 2s3z, only VDAC-mix can match or outperform the heuristic AI. It is worth noting that VDAC-sum, which cannot access extra state information, matches the naive critic's performance on most maps.

Ablation 1 Consistent with (Lowe et al. 2017), the comparison between the naive critic and IAC demonstrates the importance to incorporate extra state information, which is also revealed by the comparison between COMA and IAC (Refer to Figure 3 for comparisons between naive critic and COMA across different maps.). As shown in Figure 3, naive

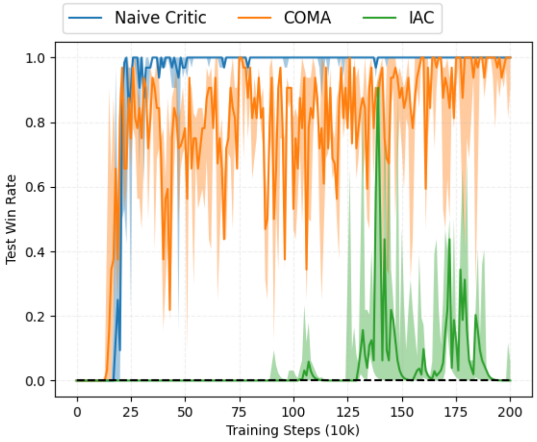


Figure 4: 2s_vs_1sc (Ablation 1)

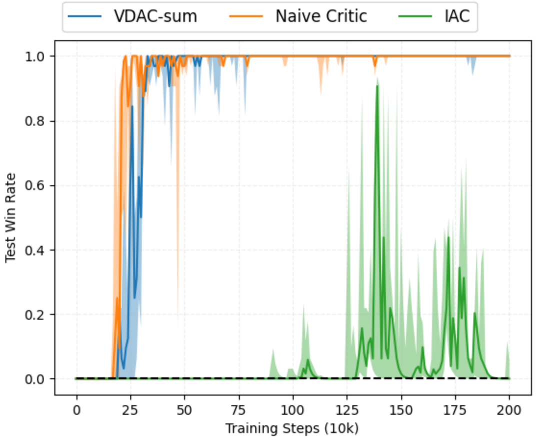


Figure 5: 2s_vs_1sc (Ablation 2)

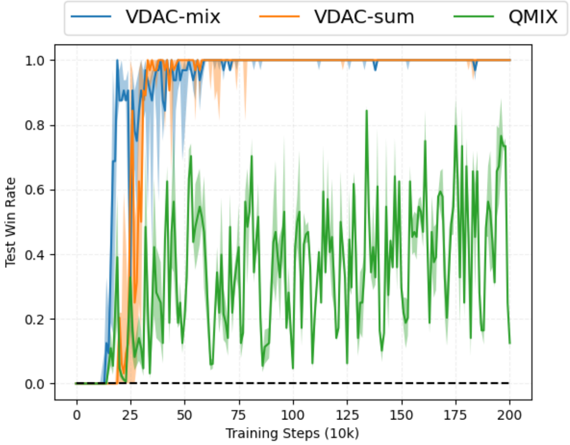


Figure 6: 2s_vs_1sc (Ablation 3)

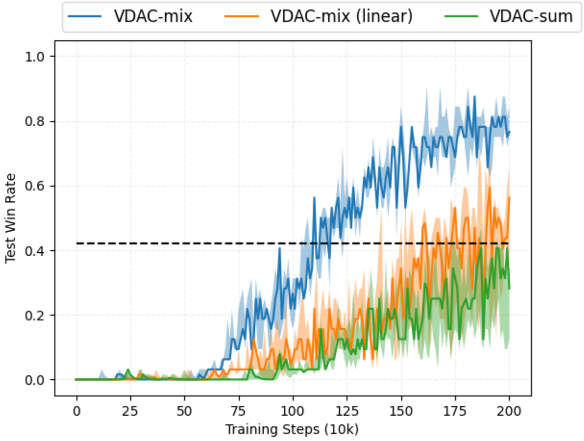


Figure 7: 3s5z (Ablation 4)

critic outperforms COMA across all tasks. It reveals that it is also viable to use a TD advantage policy gradients in multi-agent settings. In addition, COMA's training is unstable, as can be seen in Figure 4, which might arise dues to its inability to predict accurate counterfactual action-value $Q^a(s, (\mathbf{u}^{-a}, u^a))$ for un-taken actions.

Ablation 2 Despite the similarity in structure of VDAC-sum and IAC, VDAC-sum's median win rates at 2 million training step exceeds IAC's consistently across all maps (Refer to Figure 3 for comparisons between VDAC-sum and IAC across 6 different maps.). It reveals that, by using a simple relationship to enforce equation 6, we can drastically improve multi-agent actor-critic's performance. Furthermore, VDAC-sum matches naive critic on many tasks, as shown in Figure 5, demonstrating that actors that are trained without extra state information can achieve similar performance to naive critic by simply enforcing equation 6. In addition, it is noticeable that, compared with naive critic, VDAC-sum's performance is more stable across training.

Ablation 3 Figure 6 shows that, under the A2C training paradigm, VDAC-mix outperforms QMIX in map 2s_vs_1sc. It is also noticeable that QMIX's performance is unstable across the training steps in map 2s_vs_1sc. In easier games, QMIX's performance can be comparable to VDAC-mix. In harder games such as 3s5z, VDAC-mix's median test win rates at 2 million training step outnumber QMIX's by 71%. Refer to Appendix for complete comparisons between VDACs and QMIX.

Ablation 4 Finally, we introduced VDAC-mix (linear), which can be seen as a more general VDAC-sum that has access to extra state information. Consistent with our previous conclusion, the comparison between VDAC-mix (linear) and VDAC-sum shows that it is important to incorporate extra state information. In addition, the comparison between VDAC-mix and VDAC-mix (linear) shows the necessity of assuming the non-linear relationship between the global state value V_{tot} and local state values $V^a, \forall a \in \{1, \dots, n\}$. Refer to Appendix for comparisons between VDACs across all maps.

Conclusion

In this paper, we propose a new credit-assignment actor-critic framework that enforces the monotonic relationship between the global state-value and the shaped local state-value. Theoretically, we establish the convergence of the proposed actor-critic method to a local optimal. Empirically, benchmark tests on StarCraft micromanagement games demonstrate that our proposed actor-critic bridges the performance gap between multi-agent actor-critics and Q-learning, and our method provides a balanced trade-off between training efficiency and performance. Furthermore, we identify a set of key factors that contribute to the performance of our proposed algorithms via a set of ablation experiments. In future work, we aim to implement our framework in real-world applications such as highway on-ramp merging of semi or full self-driving vehicles.

Ethical Impact of Work

Large-scale multi-agent control problems are at the heart of a number of challenging problems facing society. For example in traffic management, there are over 300,000 accidents per year that occur during highway merging. The number of accidents could be significantly reduced if effective autonomous driving was broadly available in personal vehicles. MARL algorithms, like ones proposed in this paper, offer a possible solution to the autonomous driving task. Other areas of significant societal impact include healthcare, smart manufacturing, smart grids, and other transportation infrastructure.

References

Bu, L.; Babu, R.; De Schutter, B.; et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38(2): 156–172.

Choo, B. Y.; Adams, S.; and Beling, P. 2017. Health-aware hierarchical control for smart manufacturing using reinforcement learning. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 40–47. IEEE.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; and Pineau, J. 2019. TarMAC: Targeted Multi-Agent Communication. In *International Conference on Machine Learning*, 1538–1546.

Dugas, C.; Bengio, Y.; Bélisle, F.; Nadeau, C.; and Garcia, R. 2009. Incorporating Functional Knowledge in Neural Networks. *Journal of Machine Learning Research* 10(6).

Foerster, J.; Assael, I. A.; De Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, 2137–2145.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H.; Kohli, P.; and Whiteson, S. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887* .

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, 66–83. Springer.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* .

Hausknecht, M.; and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*.

Hu, Y.; Nakhaei, A.; Tomizuka, M.; and Fujimura, K. 2019. Interaction-aware Decision Making with Adaptive Strategies under Merging Scenarios. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 151–158.

Iqbal, S.; and Sha, F. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2961–2970.

Jiang, J.; and Lu, Z. 2018. Learning attentional communication for multi-agent cooperation. In *Advances in neural information processing systems*, 7254–7264.

Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, 6379–6390.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2018. Credit assignment for collective multiagent RL with global rewards. In *Advances in Neural Information Processing Systems*, 8102–8113.

Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069* 2.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485* .

Samvelyan, M.; Rashid, T.; Schroeder de Witt, C.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2186–2188. International Foundation for Autonomous Agents and Multiagent Systems.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1905.05408* .

Su, J.; Adams, S.; and Beling, P. A. 2020. Counterfactual Multi-Agent Reinforcement Learning with Graph Convolution Communication. *arXiv preprint arXiv:2004.00470* .

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo,

J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* .

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12(4): e0172395.

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

Usunier, N.; Synnaeve, G.; Lin, Z.; and Chintala, S. 2016. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993* .

Wei, E.; Wicke, D.; Freelan, D.; and Luke, S. 2018. Multiagent soft q-learning. In *2018 AAAI Spring Symposium Series*.

Wen, Y.; Yang, Y.; Luo, R.; Wang, J.; and Pan, W. 2019. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207* .

Wolpert, D. H.; and Tumer, K. 2002. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, 355–369. World Scientific.

Appendix

SMAC

In this paper, we use all the default settings in (Samvelyan et al. 2019). That includes: the game difficulty is set to level 7, *very difficult*, the shoot range, observe range, etc, are consistent with the default settings. The action space of agents consists of the following set of discrete actions: move[direction], attack[enemy id], stop, and no operation. Agents can only move in four directions: north, south, east, or west. A unit is allowed to perform the attack[enemy id] action only if the enemy is within its shooting range.

Each unit has a sight range that limits its ability to receive any information out of range. The sight range, which is bigger than shooting range, makes the environment partially observable from the standpoint of each agent. Agents can only observe other agents if they are both alive and located within the sight range. The global state, which is only available to agents during centralised training, encapsulates information about all units on the map.

The observation vector also follows the default implementation in (Samvelyan et al. 2019): It contains the following attributes for both allied and enemy units within the sight range: distance, relative x, relative y, health, shield, and unit type. In addition, the observation vector includes the last actions of allied units that are in the field of view. Lastly, the terrain features, in particular the values of eight points at a fixed radius indicating height and walkability, surrounding agents within the observe range are also included. The state vector includes the coordinates of all agents relative to the center of the map, together with units' observation feature vectors. Additionally, the energy of Medivacs and cooldown of the rest of the allied units are stored in the state vector. Finally, the last actions of all agents are attached to the state vector.

Table 2: Map Descriptions.

Map Name	Ally Units	Enemy Units
2s_vs_1sc	2 Stalkers	1 Spine Crawler
8m	8 Marines	8 Marines
2s3z	2 Stalkers & 3 Zealots	2 Stalkers & 3 Zealots
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots
1c3s5z	1 Colossus, 3 Stalkers & 5 Zealots	1 Colossus, 3 Stalkers & 5 Zealots
bane_vs_bane	20 Zerglings & 4 Banelings	20 Zerglings & 4 Banelings

Training Details and Hyperparameters

Experiments are obtained by using Nvidia RTX 2080 Ti graphics cards, with each independent run taking 1 to 3 hours depending on the scenario. Each independent run corresponds to a unique random seed that is randomly generalized at the beginning.

The agent networks of all algorithms resemble a DRQN (Hausknecht and Stone 2015) with a recurrent layer comprised of a GRU (Chung et al. 2014) with a 64-dimensional hidden state, with a fully-connected layer before and after. The exception is that IAC, VDAC-sum, and VDAC-mix agent networks contain an additional layer to output local state values and the policy network outputs a stochastic policy rather than action-values.

Algorithms are trained with RMSprop with learning rate 5×10^{-4}. During training, 8 games are initiated independently, from which episodes are sampled. Q-learning replay buffer stores the latest 5000 episodes for each independent game (In total, replay buffer has a size of $8 \times 5000 = 40000$). We set $\gamma = 0.99$ and $\lambda = 0.8$ (if needed). Target networks (if exists) are updated every 200 training steps.

The architecture of the COMA critic is a feedforward fully-connected neural network with the first 2 layers, each of which has 128 units, followed by a final layer of $|U|$ units. Naive central critic shares the same architecture with COMA critic with an exception that its final layer contains 1 units.

The mixing network in QMIX and VDAC-mix shares an identical structure. It consists of a single hidden layer of 32 units, whose parameters are outputted by hypernetworks. An ELU activation function follows the hidden layer in the mixing network. The hypernetworks consist of a feedforward network with a single hidden layer of 64 units with a ReLU activation function.

For naive central critic, IAC, and VDACs, $Q(s_t, \mathbf{u}_t)$ is given by:

$$Q(s_t, \mathbf{u}_t) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}), \tag{13}$$

where k can vary from state to state and is upper-bounded by T.

StarCraft II Results

Algorithm 1 Naive Central Critic

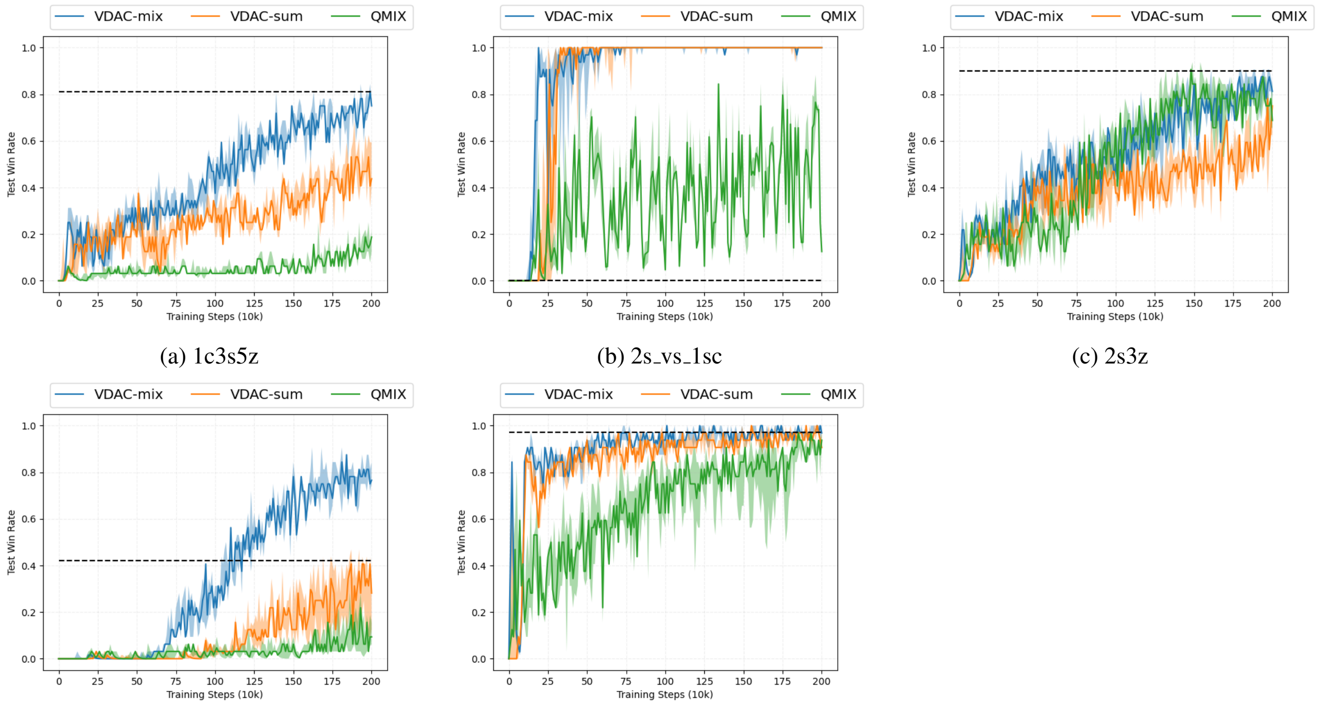
```
Initialize critic $\theta^c$, target critic $\hat{\theta}^c$, and actor $\theta$
**for** each training episode $e$ **do**
    Empty buffer
    **for** $e_c = 1$ to $\frac{\text{BatchSize}}{n}$ **do**
        $t = 0, h_o^a$ for each agent $a$
        **while** game not terminated **and** $t < T$ **do**
            $t = t + 1$
            **for** each agent $a$ **do**
                $h_t^a, \pi_t^a = \text{Actor}(o_t^a, h_{t-1}^a, u_{t-1}^a, a; \theta)$
                Sample action $u_t^a$ from $\pi_t^a$
            **end for**
            Get reward $r_t$ and next state $s_{t+1}$
        **end while**
        add experience to buffer
    **end for**
    Collate episodes in buffer into single batch
    **for** $t = 1$ to $T$ **do**
        Batch unroll RNN using states, actions and reward
        Calculate $y_t$ and $A_t$ using $\hat{\theta}^c$
    **end for**
    **for** $t = T$ down to $1$ **do**
        Calculate gradient wrt $\theta^c : \Delta\theta^c \leftarrow \nabla_{\theta^c} \big(y_t - V(s_t, \mathbf{u}_{t-1}; \theta^c)\big)^2$
        Update critic $\theta^c \leftarrow \theta^c - \alpha\Delta\theta^c$
        Every C steps update target critic $\hat{\theta^c} \leftarrow \theta^c$
    **end for**
    **for** $t = 1$ down to $T$ **do**
        Accumulate gradient wrt $\theta : \Delta\theta \leftarrow \Delta\theta + \nabla_\theta \log \pi(u_t^a | o_t^a) A_t$
    **end for**
    Update actor weights $\theta = \theta + \alpha\Delta\theta$
**end for**
```

Algorithm 2 Value Decomposition Actor-Critic (VDAC-sum)

```
Initialize actor network $\theta$
for each training episode $e$ do
    Empty buffer
    for $e_c = 1$ to $\frac{\text{BatchSize}}{n}$ do
        $t = 0, h_o^a$ for each agent $a$
        while game not terminated and $t < T$ do
            $t = t + 1$
            for each agent $a$ do
                $h_t^a, \pi_t^a, V_t^a = \text{Actor}(o_t^a, h_{t-1}^a, u_{t-1}^a, a; \theta)$
                Sample action $u_t^a$ from $\pi_t^a$
            end for
            Get reward $r_t$ and next state $s_{t+1}$
        end while
        add experience to buffer
    end for
    Collate episodes in buffer into single batch
    for $t = 1$ to $T$ do
        Batch unroll RNN using states, actions and reward
        Calculate $y_t$ and $A_t$ using $\theta$
        Accumulate gradient wrt $\theta : \Delta\theta_v \leftarrow \Delta\theta_v + \nabla_\theta \big(y_t - \sum_a V_t^a\big)^2$
    end for
    for $t = 1$ to $T$ do
        Accumulate gradient wrt $\theta : \Delta\theta_\pi \leftarrow \Delta\theta_\pi + \nabla_\theta \log \pi(u_t^a|o_t^a) A_t$
    end for
    Update actor weights $\theta = \theta + \alpha_\pi \Delta\theta_\pi - \alpha_v \Delta\theta_v$
end for
```

Algorithm 3 Value Decomposition Actor-Critic (VDAC-mix)

```
Initialize hypernetwork $\theta^c$, and actor network $\theta$
for each training episode $e$ do
    Empty buffer
    for $e_c = 1$ to $\frac{\text{BatchSize}}{n}$ do
        $t = 0, h_o^a$ for each agent $a$
        while game not terminated and $t < T$ do
            $t = t + 1$
            for each agent $a$ do
                $h_t^a, \pi_t^a, V_t^a = \text{Actor}(o_t^a, h_{t-1}^a, u_{t-1}^a, a; \theta)$
                Sample action $u_t^a$ from $\pi_t^a$
            end for
            Get reward $r_t$ and next state $s_{t+1}$
        end while
        add experience to buffer
    end for
    Collate episodes in buffer into single batch
    for $t = 1$ to $T$ do
        Batch unroll RNN using states, actions and reward
        Calculate $y_t$ and $A_t$ using $\theta^c$
        Accumulate gradient wrt $\theta^c : \Delta\theta^c \leftarrow \Delta\theta^c + \nabla_{\theta^c} \big(y_t - V_{tot}(V_t^1, \dots, V_t^n)\big)^2$
        Accumulate gradient wrt $\theta : \Delta\theta_v \leftarrow \Delta\theta_v + \nabla_\theta \big(y_t - V_{tot}(V_t^1, \dots, V_t^n)\big)^2$
    end for
    for $t = 1$ to $T$ do
        Accumulate gradient wrt $\theta : \Delta\theta_\pi \leftarrow \Delta\theta_\pi + \nabla_\theta \log \pi(u_t^a|o_t^a) A_t$
    end for
    Update actor weights $\theta = \theta + \alpha_\pi \Delta\theta_\pi - \alpha_v \Delta\theta_v$
    Update hypernet weights $\theta^c = \theta^c - \alpha \Delta\theta^c$
end for
```



(a) 1c3s5z

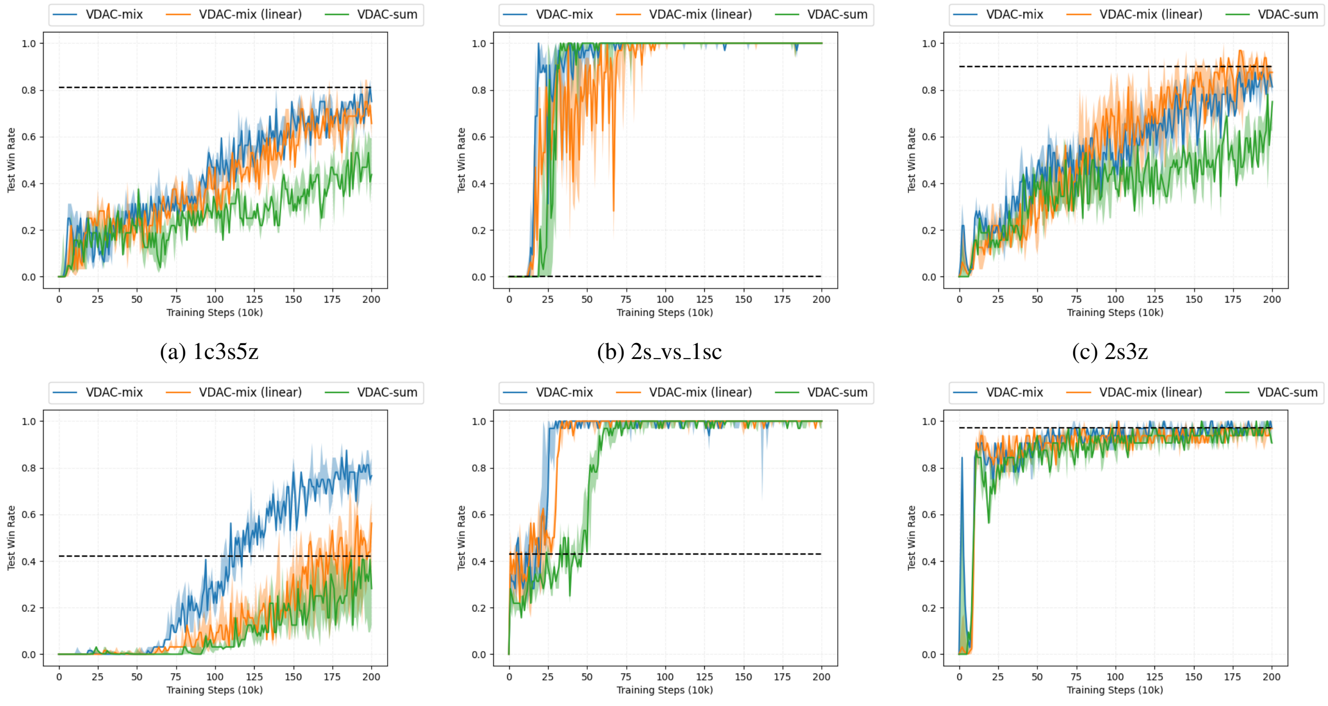
(b) 2s_vs_1sc

(c) 2s3z

(d) 3s5z

(e) 8m

Figure 8: Overall results: VDACs vs QMIX under A2C



(a) 1c3s5z

(b) 2s_vs_1sc

(c) 2s3z

(d) 3s5z

(e) bane_vs_bane

(f) 8m

Figure 9: Overall results: VDAC-mix vs VDAC-mix(linear) vs VDAC-sum