Semi-On-Policy Training for Sample Efficient Multi-Agent Policy Gradients

Bozhidar Vasilev
Department of Computer Science
University of Oxford
bozhidar.vasilev1@gmail.com

Tarun Gupta
Department of Computer Science
University of Oxford
tarun.gupta@cs.ox.ac.uk

Bei Peng
Department of Computer Science
University of Oxford
bei.peng@cs.ox.ac.uk

Shimon Whiteson
Department of Computer Science
University of Oxford
shimon.whiteson@cs.ox.ac.uk

ABSTRACT

Policy gradient methods are an attractive approach to multi-agent reinforcement learning problems due to their convergence properties and robustness in partially observable scenarios. However, there is a significant performance gap between state-of-the-art policy gradient and value-based methods on the popular StarCraft Multi-Agent Challenge (SMAC) benchmark. In this paper, we introduce semi-on-policy (SOP) training as an effective and computationally efficient way to address the sample inefficiency of on-policy policy gradient methods. We enhance two state-of-the-art policy gradient algorithms with SOP training, demonstrating significant performance improvements. Furthermore, we show that our methods perform as well or better than state-of-the-art value-based methods on a variety of SMAC tasks.

KEYWORDS

Reinforcement Learning; Policy Gradient; Multi-Agent; SMAC

1 INTRODUCTION

Many real-world learning problems, such as autonomous vehicle coordination [1] and network packet delivery [34], can be naturally modelled as cooperative multi-agent reinforcement learning (MARL) problems. Reinforcement learning (RL) algorithms designed for single-agent settings are typically unsuitable for multi-agent scenarios, because the joint action space grows exponentially with the number of agents. Furthermore, even when the joint action space is not prohibitively large, communication constraints may make it impossible to apply single-agent algorithms directly.

This often makes it necessary to learn *decentralised* policies, where the agents make decisions independently, based only on their own local observations. However, if the agents learn independently, this makes the environment non-stationary from the viewpoint of any agent, so learning might not converge. To address this issue, one often adopts the framework of *centralised training with decentralised execution* (CTDE) [7, 16], assuming a simulated or controlled setting. The CTDE framework allows the algorithm to use any extra information available during training. However, at test time each agent must make decisions based only on its private observations.

The StarCraft Multi-Agent Challenge (SMAC) [21] is a challenging MARL benchmark in which agents must learn complex coordination behaviours in an environment with significant partial observability. In recent years SMAC has become a popular environment for evaluating cooperative MARL algorithms [10, 19, 30, 35]. Despite the progress in multi-agent policy gradient (MAPG) methods, most of them still struggle to compete with state-of-the-art value-based methods on SMAC. Rashid et al. [20] show that COMA [3], a state-of-the-art on-policy MAPG algorithm, severely underperforms even simple value-based methods. Papoudakis et al. [17] go as far as using 10 times more data for MAPG methods when comparing against value-based ones. They show that state-of-the-art on-policy MAPG methods, such as CentralV [3] and COMA, suffer from poor performance even on simple SMAC scenarios.

While off-policy algorithms are generally more sample efficient than on-policy ones, state-of-the-art off-policy MAPG methods, such as MADDPG [9] and Multi-Agent Soft Actor-Critic (MASAC) [4, 6], also perform poorly on SMAC [19]. The recently proposed stochastic decomposed policy gradient (DOP) [30] algorithm mixes off-policy and on-policy training, by adapting the tree backup technique [13, 18] to the multi-agent setting and taking advantage of a factored critic representation. The authors claim their method is the first MAPG algorithm that outperforms state-of-the-art value-based methods on SMAC. However, they only present results for a small number of SMAC maps.

In this paper, we improve COMA by fixing a representation issue with the centralised critic, which produces inconsistent value estimates for some joint actions due to its network architecture. We further improve COMA by changing its training scheme to use the entire batch of data, rather than minibatches. We find that our version, COMA with a consistent critic (COMA-CC), consistently outperforms the original COMA on SMAC, and provide ablation studies to evaluate the effect of each change.

Furthermore, we introduce semi-on-policy (SOP) training as an effective and computationally efficient way to address the sample inefficiency of on-policy policy gradient algorithms. Key to our method is using recent off-policy data for training, based on an eligibility criterion. We apply SOP training to CentralV and COMA-CC, and demonstrate significant performance improvements. Our methods achieve similar or better performance and sample efficiency, compared to state-of-the-art value-based methods, on a variety of challenging SMAC tasks. Additionally, our best performing method often outperforms other state-of-the-art MAPG methods, including DOP.

2 RELATED WORK

Value-based methods. A natural value-based MARL method for learning decentralised policies is independent Q-learning [29], where each agent learns an individual action-value function and acts based on it; this is typically done using DQN [12, 28]. To exploit CTDE, Sunehag et al. [26] propose value decomposition networks (VDN), where a centralised but factored action-value function is learned, which is the sum of the individual agent utilities. The QMIX algorithm [20] expands this idea by learning a factored joint action-value function that is only constrained to be monotonic with respect to each agent's utility. Recent works, like Weighted-QMIX [19] and Qatten [33], aim to address the limitations imposed by the monotonicity constraint of QMIX, while still using a monotonic factorisation. On the other hand, works like QTRAN [25] address the monotonic restriction of QMIX by learning a VDN-factored joint action-value function along with an unrestricted centralised state-value function, and QTRAN++ [24] address the gap between the theoretical and empirical results of QTRAN.

Policy gradient methods. The most successful policy gradient methods have been actor-critic methods, where both a policy and a value function are learned during training, and the estimates of the value function are used to train the policy [27]. If each agent learns independently, we get the independent actor-critic (IAC) algorithm, which is an adaptation of the advantage actor-critic (A2C) [11] algorithm to the multi-agent setting. The CentralV algorithm [3] is another adaptation of A2C, but using a centralised critic that makes use of global state information for learning. To address the issue of multi-agent credit assignment, the COMA algorithm [3] uses a counterfactual baseline, which marginalises out an agent's action when computing the advantage of their action. Zhou et al. [35] address the same problem with their technique of learning implicit credit assignment (LICA) by formulating a centralised critic as a hypernetwork. To improve sample efficiency, the stochastic decomposed policy gradients algorithm (DOP) [30] uses a mixture of on-policy and off-policy data by adapting the tree backup technique [13, 18] to the multi-agent setting and taking advantage of a factored critic representation. Iqbal and Sha [6] adapt the soft actor-critic (SAC) [4] algorithm to the multi-agent setting and combine it with an attention mechanism and a centralised critic. The MADDPG [9] algorithm is an adaptation of the DDPG algorithm [8] to the multi-agent setting; while originally the algorithm was used to learn deterministic policies in continuous domains using off-policy data, it has since been adapted to discrete domains as well [6, 19].

SMAC. The StarCraft Multi-Agent Challenge (SMAC) [21] is a popular cooperative multi-agent reinforcement learning benchmark. The challenges require agents to learn complex cooperative behaviours, such as *kiting* and *focus fire*. For a long time, value-based methods have significantly outperformed policy gradient methods on SMAC. Some recent works, such as LICA [35] and DOP [30] have shown that actor-critic methods can sometimes outperform value-based methods on SMAC. However, they only focus on a limited set of maps, and their algorithms are much more complex compared to previous work.

Our work focuses on improving the sample efficiency of on-policy policy gradient algorithms by directly using off-policy data during training, based on an eligibility criterion. In principle, this makes it applicable to any such method. In contrast to methods that correct for off-policy data using importance sampling or returns corrections [2, 5, 14, 18, 30, 31], we simply require the generating policy for all training data to be closely related to the current behaviour policy. This removes the associated computational cost of applying corrections to the data. Furthermore, in contrast to proximal policy optimisation (PPO) [23], which uses a "surrogate" objective to constrain policy updates over multiple training iterations on a fixed set of on-policy data, our method optimises the true objective of the base algorithm and uses a rolling-window replay buffer that contains recent off-policy data, along with on-policy data. In contrast to [30, 35], we show that a simple base algorithm like CentralV suffices for many challenging SMAC tasks, with our method of training.

3 BACKGROUND

Dec-POMDPs. We consider a *fully cooperative multi-agent task*, which can be modelled as a *Decentralised Partially Observable Markov Decision Process* (Dec-POMDP) [15] consisting of a tuple $G = \langle S, U, P, r, Z, O, A, \gamma \rangle$. The state of the environment at each time step is $s \in S$. Each of n agents $a \in A \equiv \{1, \dots, n\}$ chooses an action $u^a \in U \equiv \{1, \dots, m\}$, forming the joint action $\mathbf{u} \in \mathbf{U} \coloneqq U^n$. The environment dynamics are modelled by a state transition function $P : S \times \mathbf{U} \times S \to [0, 1]$, where $P(s'|\mathbf{u}, s)$ is the probability that the environment transitions to state s' given that the current state is s and the joint action is $\mathbf{u}$. All agents share a global reward function $r : S \times \mathbf{U} \to \mathbb{R}$. We consider partially observable scenarios, where an observation function $O : S \times A \to Z$ individually determines each agent's observation z^a at each time step. Each agent has a private action-observation history $\tau^a \in T \coloneqq (Z \times U)^*$, on which it conditions a stochastic policy $\pi^a : U \times T \to [0, 1]$. The joint policy $\boldsymbol{\pi} : \mathbf{U} \times T^n \to [0, 1]$ admits a joint state-value function $V^{\boldsymbol{\pi}}(s_t) \coloneqq \mathbb{E}_{\boldsymbol{\pi}}[R_t|s_t]$ and action-value function $Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) = \mathbb{E}_{\boldsymbol{\pi}}[R_t|s_t, \mathbf{u}_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the *discounted return*.

CentralV and COMA. Both CentralV and COMA [3] learn decentralised policies π^a for each agent a, which induce a joint policy $\boldsymbol{\pi}$. During training they maximise the objective $J = V^{\boldsymbol{\pi}}(S_1) = \mathbb{E}_{\boldsymbol{\pi}}[R_1]$, following the policy gradient:

$$\nabla J = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla \log \pi^a(u^a|\tau^a) A_a^{\boldsymbol{\pi}}(s, \mathbf{u}) \right], \tag{1}$$

where $A_a^{\boldsymbol{\pi}}(s, \mathbf{u})$ is an *advantage* function, which, intuitively, estimates how much better or worse a joint action $\mathbf{u}$ is compared to the agents' expected performance under joint policy $\boldsymbol{\pi}$. As in most cooperative MARL, the agents share policy parameters [3, 20, 21].

In CentralV, the critic approximates the joint state-value function $V_\psi(s_t) \approx V^{\boldsymbol{\pi}}(s_t)$, which is used to estimate the advantage for all agents:

$$A_a^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) = r_t + \gamma V^{\boldsymbol{\pi}}(s_{t+1}) - V^{\boldsymbol{\pi}}(s_t) \tag{2}$$

$$\approx r_t + \gamma V_\psi(s_{t+1}) - V_\psi(s_t) \tag{3}$$

In COMA, the critic approximates the joint action-value function $Q_\psi(s_t, \mathbf{u}_t) \approx Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t)$, which is used to estimate a counterfactual

advantage for each agent:

$$A_a^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) \tag{4}$$

$$= Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) - \sum_{u' \in U} \pi^a(u'|\tau^a) Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t^{-a}, u') \tag{5}$$

$$\approx Q_\psi(s_t, \mathbf{u}_t) - \sum_{u' \in U} \pi^a(u'|\tau^a) Q_\psi(s_t, \mathbf{u}_t^{-a}, u'), \tag{6}$$

where $Q_\psi(s_t, \mathbf{u}_t^{-a}, u')$ is the action-value function evaluated by keeping the actions of all agents except a fixed, and replacing agent a's action with u'. The quantity:

$$\sum_{u' \in U} \pi^a(u'|\tau^a) Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t^{-a}, u') \tag{7}$$

is called the *counterfactual* baseline.

In both CentralV and COMA, the critic f_ψ is trained to minimise the loss $\mathcal{L}(\psi) = (y^{(\lambda)} - f_\psi(\cdot_t))^2$, using a variant of TD($\lambda$) [27]. The targets $y^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$ are a mixture of n-step returns:

$$G_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n f_{\psi'}(\cdot_{t+n}), \tag{8}$$

which are calculated using bootstrapped values from a target network [12]. Periodically, the target network parameters are copied from ψ.

4 METHOD

In this section, we present our semi-on-policy (SOP) training framework and our COMA-CC algorithm.

4.1 Semi-on-policy training

The key insight behind our method is that, if the policy of the agents only changes slightly after a step of training, then the data generated by the previous policy can be reused in the next training step. In particular, we assume that if an episode e was generated following a policy π', and $KL(\pi \parallel \pi') \leq$ `kl_threshold`, then we can still use the episode e for training our policy π.[1]

Thus, if the base on-policy algorithm induces gradual policy updates, we can train it using the most recent episodes, despite the fact that some of them were generated by a different policy. To take advantage of this, we propose permissive semi-on-policy training (Algorithm 1). Our results on SMAC show that this is an effective and reliable way of improving the sample efficiency of on-policy algorithms. For cases when the policy might diverge significantly during one update, we propose strict semi-on-policy training (Algorithm 2), which enforces the KL constraint between policies in the replay buffer. Because our analysis in Section 7.1 shows that permissive SOP training (henceforth simply called SOP training) is generally stable on SMAC, we defer evaluating strict SOP training to future work.

4.1.1 Implications on batch size. Typical implementations of value-based MARL algorithms use a batch size of 32 episodes on SMAC [10, 19, 20]. Because value-based methods generally use Q-learning [32] to train their value-function estimator [10, 19, 20], they can reuse their old data. On-policy algorithms, however, must discard their data after training, so they typically opt for a smaller batch size

[1]The choice of using the KL-divergence between policies to quantify similarity is somewhat arbitrary, and other criterions could also be used.

Algorithm 1: Permissive semi-on-policy training

```
Let b be the batch-size.
Initialise the current policy π and replay buffer buf of size b.
Fill buf with b episodes.
while there is still time to train do
    Train π using all data in buf.*
    Discard the least recent episode from buf.
    Sample a new episode from the environment following
      the current policy π and insert it into buf.
// (*) this step depends on how the particular policy gradient
   algorithm performs policy updates. For actor-critic
   algorithms: also train the critic.
```

Algorithm 2: Strict semi-on-policy training

```
Let b be the batch-size.
Let kl_threshold be a hyperparameter.
Initialise the current policy π, replay buffer buf of size b and
   corresponding policy buffer pbuf.
// Maintain the invariant that buf[i] was generated by
   following the policy pbuf[i].
while there is still time to train do
    Sample new episodes from the environment following the
      current policy π and insert them into buf until it is full.
      Fill corresponding locations of pbuf with π.
    Train π using all data in buf.*
    Discard the least recent episode from buf and all
      episodes generated by a sufficiently different policy.**
// (*) this step depends on how the particular policy gradient
   algorithm performs policy updates. For actor-critic
   algorithms: also train the critic.
// (**) we say π' is sufficiently different from π if
   KL(π ∥ π') > kl_treshold.
```

[20], a massive increase in the number of episodes sampled from the environment [35], or both [17]. If policy updates are stable across a small number of training steps, our method of training allows us to perform the same number of training steps as the currently popular off-policy value-based MARL methods, without requiring more data.

4.2 COMA-CC

In this section we describe the improvements our COMA-CC algorithm makes to the original COMA.

4.2.1 Fixing the critic inconsistency. The COMA critic computes the counterfactual Q-function for each agent:

$$Q^a(s_t, \mathbf{u}_t) := \left\langle Q(s_t, u_t^a = 1, \mathbf{u}_t^{-a}), \ldots, Q(s_t, u_t^a = |U|, \mathbf{u}_t^{-a}) \right\rangle \tag{9}$$

varying the action of agent a, while keeping other agents' actions fixed. In particular it estimates:

$$Q^a(s_t, \mathbf{u}_t) \approx Q_\psi(s_t, z_t^a, \mathbf{u}_{t-1}, \mathbf{u}_t^{-a}, a) \tag{10}$$

where Q_ψ is a neural network with parameters ψ. Here s_t is the central state, z_t^a is the observation of agent a at time step t, $\mathbf{u}_{t-1}$ is the joint action at time step $(t-1)$, $\mathbf{u}_t^{-a}$ is the joint action at time step t with the action of agent a masked out, and a is the agent id.

Consider a simple scenario where we have two agents, a_1, a_2, and they can perform a single action, u. Let $Q^1 = f(s_t, z_t^1), Q^2 = f(s_t, z_t^2)$ be the (neural network) estimates of the two agents' action-value functions. Since, in general, $z_t^1 \neq z_t^2$, this means that, in general, $Q^1(u,u) \neq Q^2(u,u)$. Thus, the agents' estimates for the *actual* joint action taken at that time step are inconsistent with each other. The observation difference among agents is one high-level inconsistency that causes this issue, while another is having the agent id as input. The way that $\mathbf{u}_t^{-a}$ is implemented (as a concatenation of one-hot action-vectors with one of the actions masked out) has a similar impact.

One of the intuitive issues this causes is that two agents might disagree about whether the same joint action was good or bad. Thus, when they compute their respective share of the gradient, one will move the policy parameters towards making the joint action more likely, while the other will move away from it. Since the agents share parameters, this can cause convergence issues.

To address this issue, COMA-CC uses an alternative critic that estimates:

$$Q(s_t, \mathbf{u}_t) \approx Q_\phi(s_t, z_t^1, \ldots, z_t^n, \mathbf{u}_{t-1}, \mathbf{u}_t) \tag{11}$$

We can use this critic to compute each component of $Q^a(s_t, \mathbf{u}_t)$ by inputting the respective counterfactual joint actions. Furthermore, this critic always receives the same inputs when estimating the same joint action, resolving the issue of inconsistent estimates.

The new critic can compute all necessary counterfactual Q-values in a single forward pass; this is done by building the input for each counterfactual Q-value computation and running the concatenated inputs through the network. For each counterfactual Q-value computation, the COMA critic requires n inputs, one for each agent, and the COMA-CC critic requires nm inputs, one for each agent and counterfactual joint action; therefore, if the internal network architectures are similar for both critics, the COMA-CC critic must perform $O(m)$ times more multiplications than the COMA critic. To ensure this, the concatenated observations $(z_t^1, \ldots, z_t^n)$ must be compressed before they are used as inputs to the critic; this can be done with an encoding network. Since the running time of our experiments was essentially the same for both the COMA and COMA-CC critics, we do not include this optimisation in our implementation.[2]

4.2.2 Critic training. The critic in COMA is trained according to the procedure given in Algorithm 3. This way of training means that each training iteration uses only `batch_size` steps of agent-environment interaction to compute the gradients: for example, if the `batch_size` is 8 episodes, and each episode has length 128, COMA will train on minibatches of 8 steps until it trains on all data from the episodes. Our ablative study in Section 7.2 indicates that this destabilises learning; therefore, we train COMA-CC on the whole batch at once. The change necessary is to the for-loop, and is given in Algorithm 4.

[2] A more thorough analysis of the computational complexity of the COMA-CC critic is included in Appendix B.

Algorithm 3: COMA Critic Training [3]

Initialise the critic parameters ϕ, target critic parameters $\hat{\phi} = \phi$, and policy parameters θ
...
[Let T be the maximum length of an episode.]
// For all agents $a \in A$ simultaneously:
Calculate TD(λ) targets y_t^a using the target critic $\hat{\phi}$
for $t = T$ **to** *1* **do**
  $\Delta Q_t^a = y_t^a - Q_\phi(s_t, \mathbf{u}_t)$
  $\Delta\phi = \nabla_\phi (\Delta Q_t^a)^2$ // calculate critic gradient
  $\phi = \phi - \alpha \Delta\phi$ // update critic parameters
  Every C steps reset $\hat{\phi} = \phi$
...

Algorithm 4: Alternative COMA Critic Training

$\Delta\phi = 0$
for $t = T$ **to** *1* **do**
  $\Delta Q_t^a = y_t^a - Q_\phi(s_t, \mathbf{u}_t)$
  $\Delta\phi = \Delta\phi + \nabla_\phi (\Delta Q_t^a)^2$ // accumulate critic gradient
$\phi = \phi - \alpha \Delta\phi$ // update critic parameters
Every C steps reset $\hat{\phi} = \phi$

5 EXPERIMENTAL SETUP

We evaluate all methods on the challenging SMAC benchmark [21], using a batch size of either 8 or 32 episodes. In all comparisons, all methods use the same batch size; we indicate this in the figure notes. We run 4 random seeds per map for each method.

For QMIX and DOP we use the author-provided open-source implementations.[3] We create fresh implementations of CentralV and COMA-CC, based on the existing CentralV and COMA implementations.[4] The changes we make are outlined in Section 4. Additionally, we set $\gamma = 1$ when computing the advantage function for CentralV, which we find helps stabilise policy updates; we include results for the version without this change in Appendix C.2.

All experiments, including baselines, use StarCraft II Version `2.4.10`.[5] We use the same hyperparameters, network architectures, and evaluation methodology as in the original SMAC paper [21], unless stated otherwise. We list these in detail in Appendix A. Doing so, we have attempted to eliminate all possible confounding factors in our COMA-CC ablation study and in our comparisons of CentralV and COMA-CC. As such, our results should be interpreted as purely comparative, and we believe that better absolute results are achievable in practice, with proper tuning.

We focus on 6 maps in this section: 3 easy maps `2s3z`, `3s5z`, `1c3s5z`, 2 hard maps `bane_vs_bane`, `2c_vs_64zg`, and 1 super-hard map `MMM2`. The easy maps were chosen because of the

[3] https://github.com/oxwhirl/pymarl and https://github.com/TonghanWang/DOP.
[4] https://github.com/oxwhirl/pymarl.
[5] At the time of writing, the latest version of the PyMARL framework provided by the original SMAC authors uses `SC2.4.10`. Older versions of the framework used `SC2.4.6.2.69232`. The SMAC authors note that results are not always comparable between versions. We have retested all baselines to ensure fair comparisons.

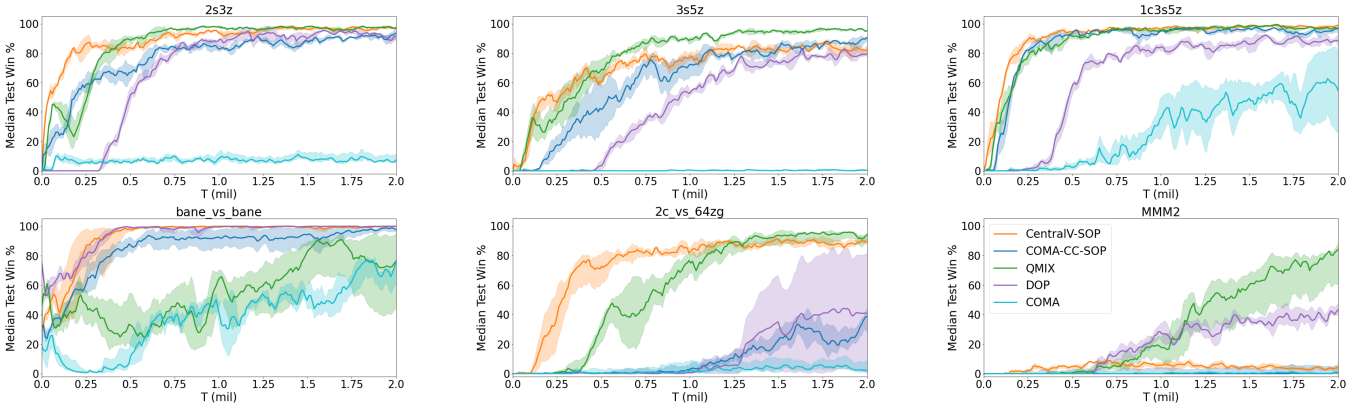

Figure 1: Comparing MAPG methods with QMIX. Batch size 32. Easy maps (top): `[2s3z, 3s5z, 1c3s5z]`, Hard maps (bottom): `[bane_vs_bane, 2c_vs_64zg]`, Super-hard maps (bottom): `[MMM2]`.

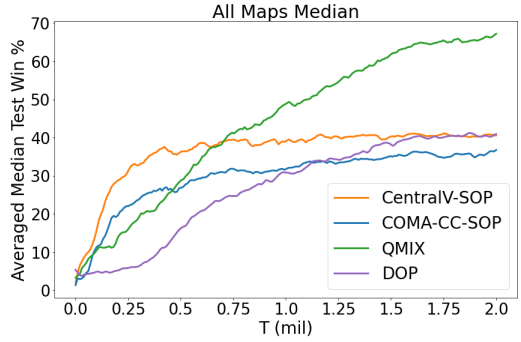

Figure 2: Averaged median test win rate for policy gradient algorithms and QMIX across all 14 SMAC scenarios.

surprisingly low performance of CentralV and COMA reported previously in the literature [17, 20, 21, 30].[6] The hard maps include a challenging symmetric and asymmetric scenario, and the super-hard map tests the limits of our method. For each map, we report the median test win-rate across all seeds, and we shade the area between the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentile. For aggregate results, we plot the averaged median win-rate across all seeds and all 14 SMAC maps for the given algorithm. Finally, when we refer to the entire SMAC challenge, we mean the 14 maps explicitly mentioned in the original paper [21]; in particular, we do not consider the additional maps distributed with the package. The algorithms that use semi-on-policy training are suffixed with "-SOP".

6 RESULTS

Figure 1 shows that CentralV-SOP and COMA-CC-SOP perform on par with the state-of-the-art value-based method QMIX on various SMAC maps, and even outperform it on some maps. In particular, QMIX fails to converge on `bane_vs_bane`, whereas all MAPG methods manage to solve the scenario. Additionally, CentralV-SOP achieves the same performance as QMIX on `2c_vs_64zg`, outperforming all other MAPG methods and converging much faster. The best performing MAPG method is CentralV-SOP, which outperforms other MAPG methods on most maps. Furthermore, COMA-CC-SOP significantly outperforms the original COMA. Results for all 14 SMAC maps are included in Appendix C.

Figure 2 shows the aggregate results on the entire SMAC challenge. QMIX performs the best on average. However, as previously discussed, it is not the best on all maps. CentralV-SOP and DOP achieve approximately the same performance at the end of training, with CentralV-SOP converging much faster. One significant difference between CentralV and COMA or DOP is that the former learns a centralised V-function, while the latter learn a centralised Q-function, which could be more difficult to learn in the multi-agent setting due to the exponentially sized joint-action space.

Figures 3 and 4 show the results comparing SOP training to on-policy training for CentralV and COMA-CC respectively. We see that the algorithms trained with SOP data consistently outperform the corresponding on-policy versions, demonstrating better sample efficiency and stable learning.

7 ANALYSIS

7.1 SOP Training

In this section we provide more insight into when SOP training works and how to best utilise it. Figure 5 shows a plot of the maximum KL divergence between the current policy π and any other policy in the buffer for various maps. To estimate the KL divergence, we use an unbiased estimator [22]:

$$KL(p \parallel q) = \mathbb{E}_{x\sim p}\left[\frac{q(x)}{p(x)} - 1 - \log\frac{q(x)}{p(x)}\right] \tag{12}$$

As before, we plot the median across seeds and shade the area between the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentile. In the map `3s5z` we can see that the divergence is extremely low: less than 4 in most cases, and going up to at most 10 for the 75$^{\text{th}}$ percentile. This fits our intuitive description that we can use past data, as long as the current policy has not diverged too much from the policy that generated it. However, even divergences as high as 150 can still yield stable

[6] Some of these papers use the older version of StarCraft II. However, our experiments indicate that the low performance is not due to version number alone.

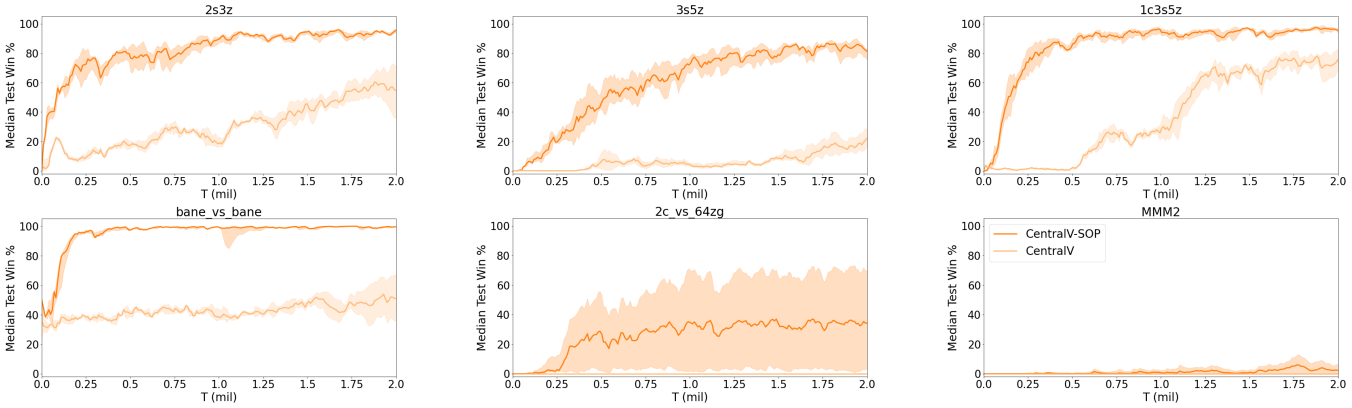

Figure 3: Comparing CentralV with SOP training and with purely on-policy training. Batch size 8.

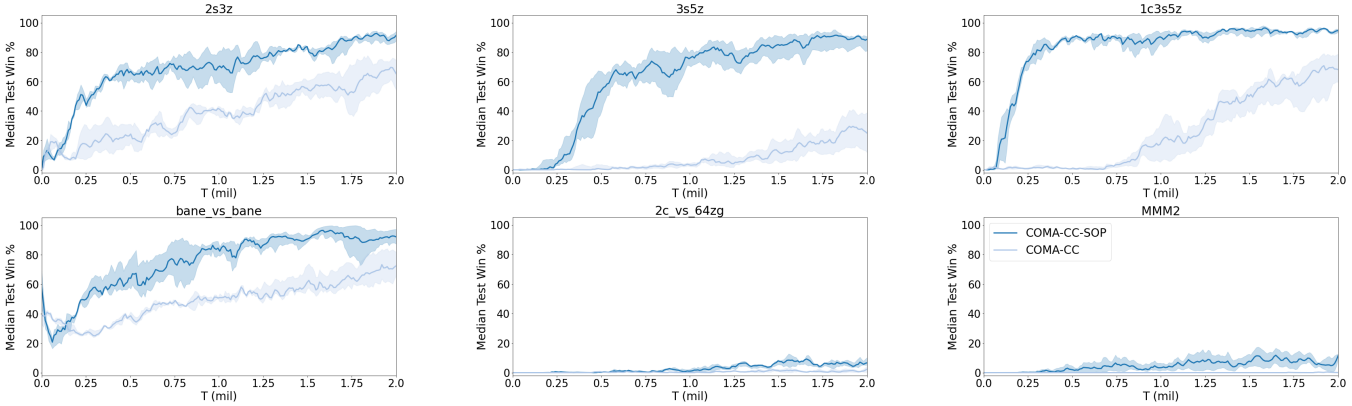

Figure 4: Comparing COMA-CC with SOP training and with purely on-policy training. Batch size 8.

training, as the results for `2c_vs_64zg` indicate. The reason is that the action space in `2c_vs_64zg` is much larger than most other maps, and there are many actions with a similar effect; thus, while the KL divergence may rise, the underlying behaviour remains approximately the same.

In practice we find that sudden big changes to the current policy lead to instability when using SOP training. There are different ways to remedy this: for example modifying critic training or advantage estimation, lowering the learning rate, or using strict SOP training (Algorithm 2) with an appropriate `kl_threshold`. We hypothesise that, in most cases, permissive SOP training (Algorithm 1) would be preferable, since, if the policy changes are unstable, strict SOP training would prune too much of the old data and essentially degenerate to on-policy training. The KL estimator is also an important factor to consider when determining appropriate values of `kl_threshold`. Additionally, our results from Figure 5 indicate that, if used as a hyperparameter, it should be tuned on a per-application basis (i.e., per map in the case of SMAC). However, our results from Section 6 indicate that permissive SOP training suffices for SMAC.

7.2 COMA Ablations

We now analyse the effect of each change we made to the COMA algorithm. Figure 6 shows the averaged median test win rate across all 14 SMAC scenarios. We use the same naming convention as before, with the addition of the `COMA-WholeBatch` algorithm, which incorporates new training from Section 4.2.2, but does not use the new critic proposed in Section 4.2.1.

We find that COMA-CC-SOP outperforms all ablations. Training on the entire batch at once and using SOP training are crucial to achieving the best results with COMA. We find that it is important to have both additions, as `COMA-SOP` (just doing SOP training) and `COMA-CC` (just doing whole-batch training), produce far weaker results compared to the ablations that incorporate both changes. Fixing the critic representation also yields some improvements, though they are less significant than we expected, as `COMA-WholeBatch-SOP` performs similarly to `COMA-CC-SOP`. Additionally, we do not find significant differences between using a batch size of 32 episodes, compared to using a batch size of 8 episodes. Overall, COMA-CC-SOP achieves a more than 200% higher average median test win-rate compared to the original COMA on the SMAC benchmark. Additionally, it exhibits more stable learning.

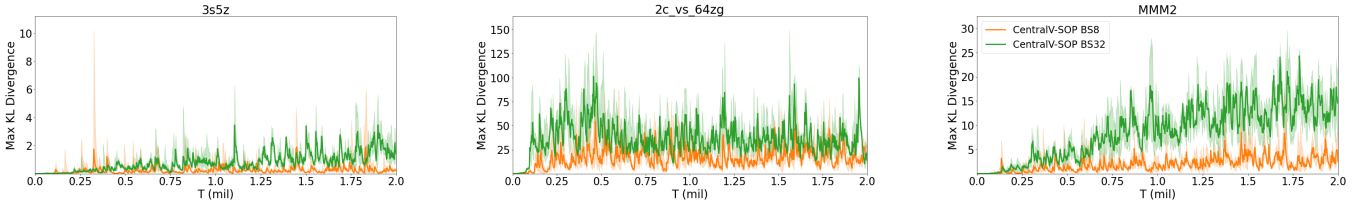

Figure 5: Maximum estimated KL divergence between the current policy π and any other policy π' in the buffer. Median over all seeds. BS indicates batch size.

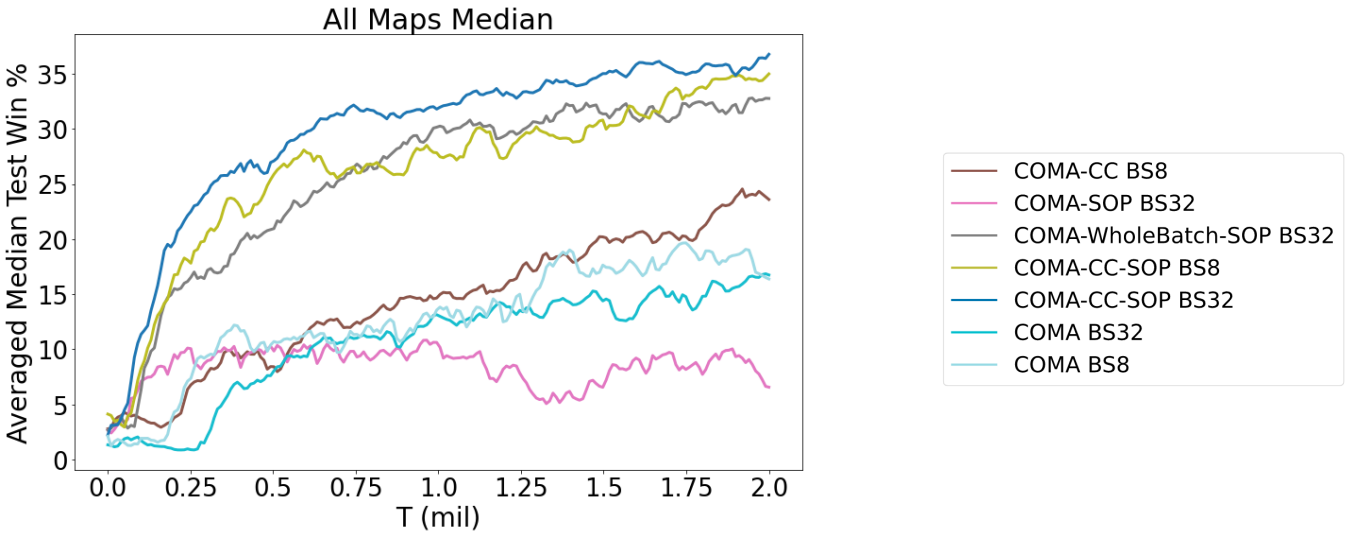

Figure 6: Averaged median test win rate for COMA-CC ablations across all 14 SMAC scenarios. BS indicates batch size.

8 CONCLUSION

In this paper we proposed a simple to implement enhancement that yielded significant improvements to the performance and sample efficiency of both multi-agent policy gradient algorithms we examined: namely, using semi-on-policy training. With this enhancement, we established CentralV-SOP as a strong benchmark for multi-agent policy gradient algorithms on SMAC, showing that it is competitive with state-of-the-art value-based and policy gradient methods. The fact that CentralV-SOP often outperforms COMA and DOP, two methods that address the multi-agent credit assignment problem, indicates that credit assignment might be an insignificant factor on SMAC.

9 ACKNOWLEDGEMENTS

We thank the reviewers for their comments and suggestions.

REFERENCES

[1] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2012. An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination. *IEEE Transactions on Industrial Informatics* 9 (07 2012). https://doi.org/10.1109/TII.2012.2219061

[2] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1146–1155. http://proceedings.mlr.press/v70/foerster17b.html

[3] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2974–2982.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*. 1856–1865.

[5] Anna Harutyunyan, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. 2016. Q(λ) with Off-Policy Corrections. *CoRR* abs/1602.04951 (2016). arXiv:1602.04951

[6] Shariq Iqbal and Fei Sha. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, Long Beach, California, USA, 2961–2970.

[7] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG]

[9] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017), 6382–6393.

[10] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems*. 7611–7622.

[11] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1928–1937.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (01 Feb 2015), 529–533. https://doi.org/10.1038/nature14236

[13] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc., 1054–1062.

[14] Remi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc.

[15] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs* (1st ed.). Springer Publishing Company, Incorporated.

[16] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[17] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2020. Comparative Evaluation of Multi-Agent Deep Reinforcement Learning Algorithms. arXiv:2006.07869 [cs.LG]

[18] Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 759–766.

[19] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 10199–10210.

[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research* 21, 178 (2020), 1–51.

[21] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2186–2188.

[22] John Schulman. 2020. *Approximating KL Divergence*. http://joschu.net/blog/kl-approx.html

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[24] Kyunghwan Son, Sungsoo Ahn, Roben Delos Reyes, Jinwoo Shin, and Yung Yi. 2020. QTRAN++: Improved Value Transformation for Cooperative Multi-Agent Reinforcement Learning. arXiv:2006.12010 [cs.LG]

[25] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5887–5896.

[26] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.

[27] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). MIT Press, Cambridge, MA, USA.

[28] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE* 12, 4 (04 2017), 1–15. https://doi.org/10.1371/journal.pone.0172395

[29] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *In Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, 330–337.

[30] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. 2020. Off-Policy Multi-Agent Decomposed Policy Gradients. arXiv:2007.12322 [cs.LG]

[31] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample Efficient Actor-Critic with Experience Replay. *CoRR* abs/1611.01224 (2016). arXiv:1611.01224

[32] Christopher Watkins. 1989. *Learning From Delayed Rewards*. Ph.D. Dissertation. King's College, Cambridge.

[33] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. arXiv:2002.03939 [cs.MA]

[34] Dayong Ye, Minjie Zhang, and Yun Yang. 2015. A multi-agent framework for packet routing in wireless sensor networks. *Sensors (Basel, Switzerland)* 15, 5 (28 Apr 2015), 10026–10047. https://doi.org/10.3390/s150510026 25928063[pmid].

[35] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. 2020. Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. arXiv:2007.02529 [cs.LG]

A EXPERIMENTAL SETUP

All environment configurations for SMAC are kept at their default values. We use the same network architecture and hyperparameters as in [21], unless stated otherwise. The configuration is the same for both CentralV and COMA-CC. We present it here for completeness.

A.1 Network architecture

The policy network is a recurrent network which contains a GRU of hidden dimension 64 with a fully-connected linear layer before and after. To speed up learning, parameters are shared across agents. The critic architecture for both algorithms is a fully-connected feed-forward neural network, with 2 hidden layers of 128 units each, and a final layer of 1 unit; as discussed in Section 4.2.1, the original COMA critic has a final layer of $|U|$ units. On each training iteration, we first update the critic, and then the actor, performing a single gradient descent step in both cases; the original implementations of these algorithms used minibatch gradient descent to update their critics, as in Algorithm 3 from Section 4.2.2. Both networks apply a ReLU non-linearity between all but the final layer.

A.2 Hyperparameters

For both CentralV and COMA-CC (including ablations) we use the following hyperparameters: for TD(λ) we set $\lambda = 0.8$ for all maps, with discount factor $\gamma = 0.99$. Both algorithms update the target critic every 200 training iterations. The actor and critic are optimised separately using an RMSProp optimiser; in both cases we set the optimiser parameters as follows: learning rate = 0.005, $\alpha = 0.99$, and $\varepsilon =$ 1e-5. To enforce exploration, both algorithms use an ϵ-floor scheme (as in [3]) for the agents' policies, with ϵ linearly annealed from 0.5 to 0.01 over 100k time steps. As in [3, 21], at test time, the algorithms perform greedy action selection, choosing the actions with highest probability.

B COMPUTATIONAL COMPLEXITY OF THE COMA-CC CRITIC

Multiplying a $n \times k$ matrix with a $k \times m$ matrix requires $O(n \times k \times m)$ multiplications. The COMA critic is a neural network, which takes i_1 inputs, has two hidden layers of size h, and produces m outputs. The COMA-CC critic takes i_2 inputs, has two hidden layers of size h, and produces a single output. Thus, on a single input, the COMA critic requires $O(hi_1) + O(h^2) + O(hm)$ multiplications to compute its outputs and the COMA-CC critic requires $O(hi_2) + O(h^2) + O(h)$ multiplications.

Note that h is typically a fixed constant. Furthermore, note that $m = O(\min\{i_1, i_2\})$, since the joint action is included in the input. Thus, we get that the COMA critic requires $O(i_1)$ multiplications and the COMA-CC critic requires $O(i_2)$ multiplications. For each baseline computation, the COMA critic takes n inputs, one for each agent, so requires $O(n \times i_1)$ multiplications; the COMA-CC critic takes nm inputs, one for each agent and counterfactual joint action, so requires $O(n \times m \times i_2)$ multiplications. Therefore, assuming $i_1 = \Theta(i_2)$, the COMA-CC critic network performs $O(m)$ times as many multiplications as the COMA critic.

C FULL RESULTS

In this section, we present our full set of results.

C.1 Actor-critic algorithms vs QMIX

The per-map comparisons of all algorithms we examine are presented in Figure 9.

C.2 CentralV-default

Here we show the results for using SOP training on CentralV without using the $\gamma = 1$ trick when computing advantages. We refer to this version as CentralV-default. The results are shown in Figure 7. SOP training still manages to improve the performance of the algorithm. The increased training instability can be explained by unstable policy updates, as shown in Figure 8.

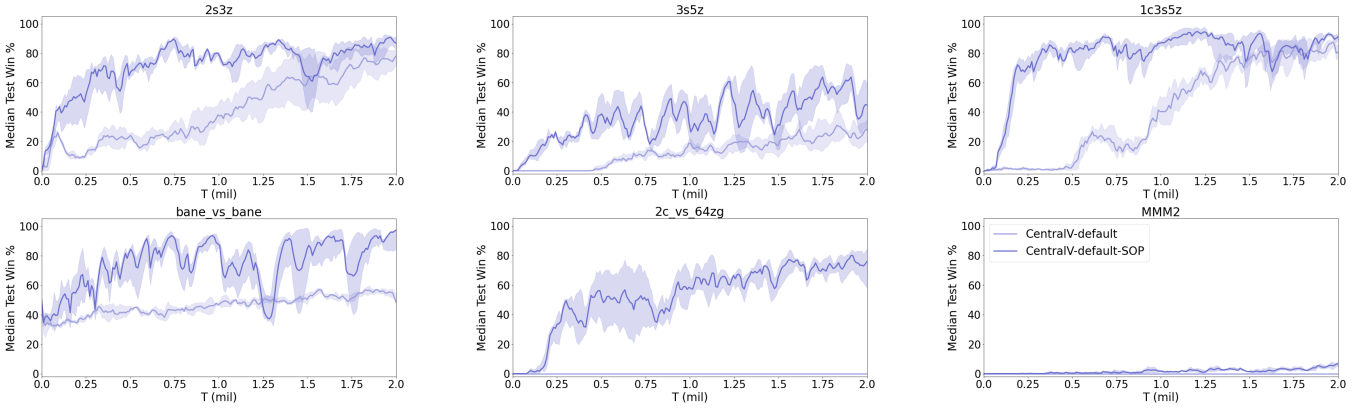

Figure 7: Comparing CentralV-default with SOP training and with purely on-policy training. Batch size 8.

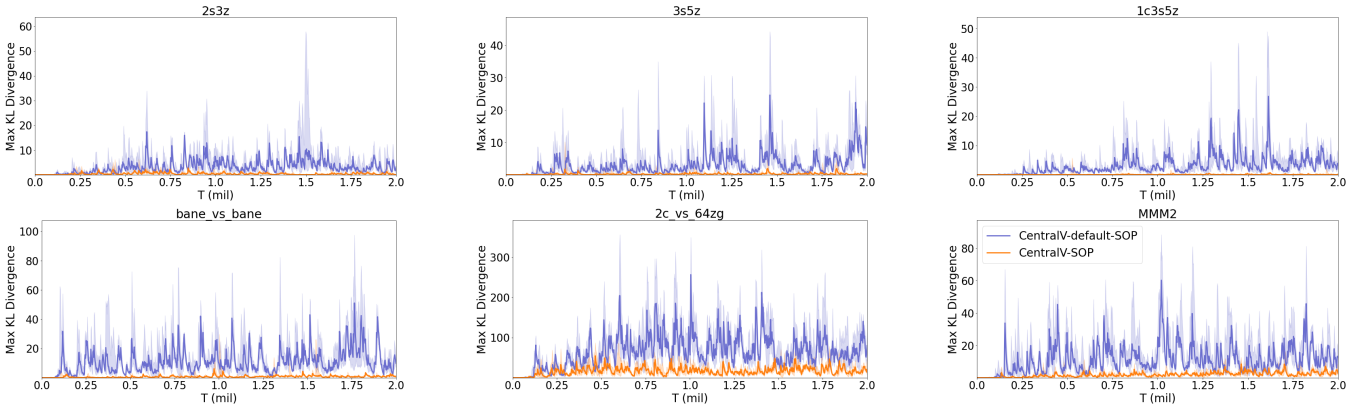

Figure 8: Comparing KL divergence of CentralV and CentralV-default with SOP training. Batch size 8.

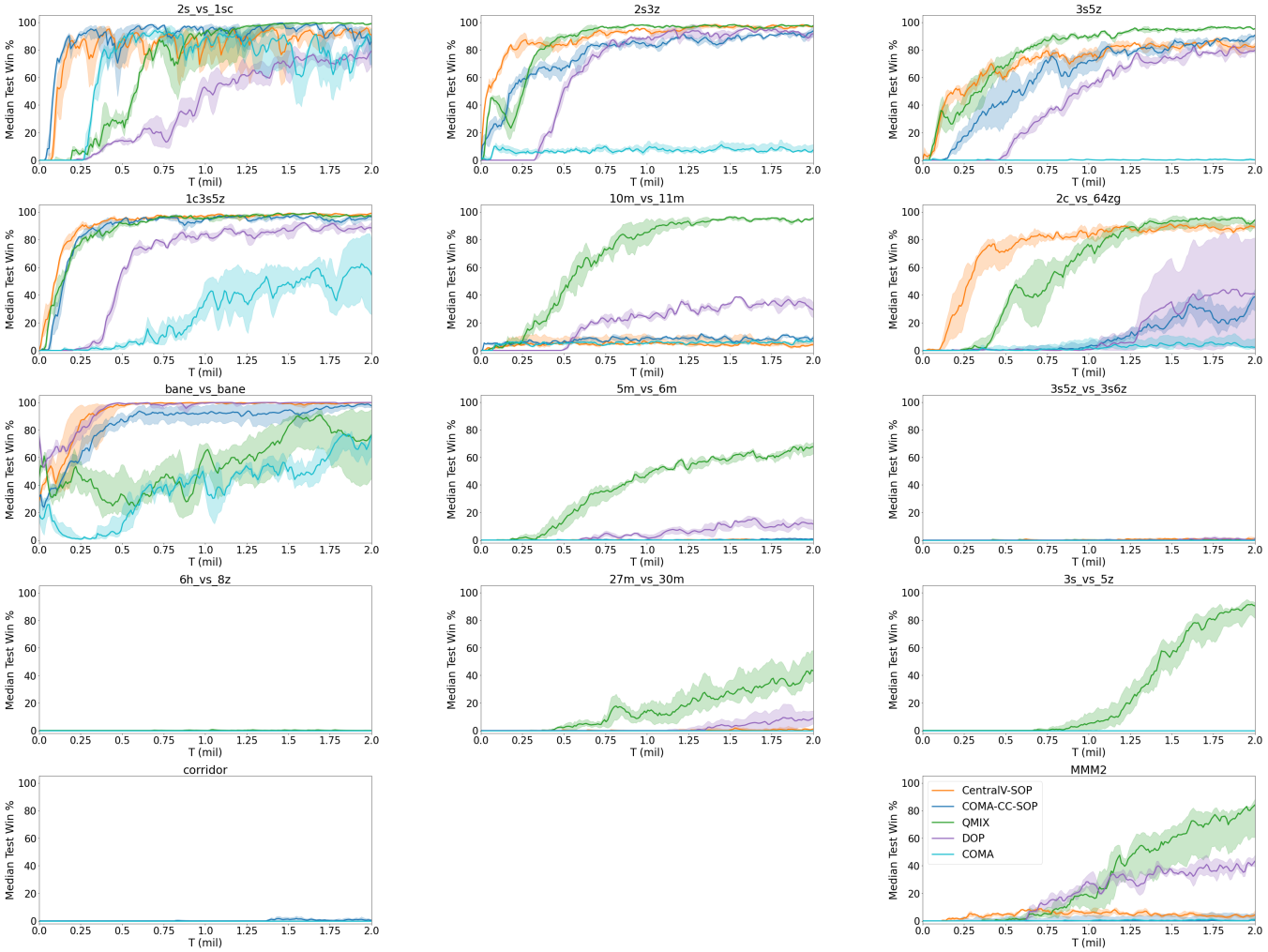

Figure 9: Comparing MAPG methods with QMIX on the entire SMAC challenge. Batch size 32.