SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning

Benjamin Ellis[1*] **Jonathan Cook**[1] **Skander Moalla**[1] **Mikayel Samvelyan**[2 3]
Mingfei Sun[1] **Anuj Mahajan**[1] **Jakob N. Foerster**[1] **Shimon Whiteson**[1]

[1]University of Oxford [2]University College London [3]Meta AI

1 Links

- SMACv2: https://github.com/oxwhirl/smacv2. MIT License.
- MAPPO implementation: https://github.com/benellis3/mappo.
- QMIX implementation: https://github.com/benellis3/pymarl2. Apache-2 License.

2 Further Background

2.1 StarCraft

StarCraft II is a real-time strategy game featuring 3 different races, Protoss, Terran and Zerg, with different properties and associated strategies. The objective is to build an army powerful enough to destroy the enemy's base. When battling two armies, players must ensure army units are acting optimally. This is called *micromanagement*. An important micromanagement strategy is *focus firing*, which is ordering all allied units to jointly target the enemies one by one to ensure that damage taken is minimised.

Another important strategy is *kiting*, where units flee from the enemy and then pick them off one by one as they chase.

2.2 QMIX

QMIX can be thought of as an extension of DQN[7] to the Dec-POMDP setting. The joint optimal action is found by forcing the joint Q to adhere to the individual global max (IGM) principle[11], which states that the joint action can be found by maximising individual agents' Q_i functions:

$$\arg\max_{\boldsymbol{a}} Q(s, \boldsymbol{\tau}, \boldsymbol{a}) = \begin{cases} \arg\max_a Q_1(\tau_1, a_1) \\ \arg\max_a Q_2(\tau_2, a_2) \\ \dots \\ \arg\max_a Q_n(\tau_n, a_n) \end{cases}$$

This central Q is trained to regress to a target $r + \gamma\hat{Q}(s, \boldsymbol{\tau}, \boldsymbol{a})$ where $\hat{Q}$ is a target network that is updated slowly. The central Q estimate is computed by a mixing network, whose weights are conditioned on the state, which takes as input the utility function Q_i of the agents. The weights of the mixing network are restricted to be positive, which enforces the IGM principle[11] by ensuring the central Q is monotonic in each Q_i.

*Correspondence to benellis@robots.ox.ac.uk.

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

2.3 Independent and Multi-agent PPO

Proximal Policy Optimisation (PPO) is a method initially developed for single-agent reinforcement learning which aims to address performance collapse in policy gradient methods. It does this by heuristically bounding the ratio of action probabilities between the old and new policies. To this end, it optimises the below objective function.

$$\mathbb{E}_{s\sim d^{\pi},a\sim\pi}\left[\min\left(\frac{\tilde{\pi}(a|s)}{\pi(a|s)}A^{\pi}(s,a),\text{clip}\left(\frac{\tilde{\pi}(a|s)}{\pi(a|s)},1-\epsilon,1+\epsilon\right)A^{\pi}(s,a)\right)\right]$$

where $\text{clip}(t,a,b)$ is a function that outputs a if $t<a$, b if $t>b$ and t otherwise.

Extending this to the multi-agent setting can easily done in two ways. The first is to use independent learning, where each agent treats the others as part of the environment and learns a critic using its observation and action history. The second is to use a centralised critic conditioned on the central state. This is called multi-agent PPO. Both independent PPO (IPPO) and multi-agent PPO (MAPPO) have demonstrated strong performance on SMAC[16, 2]. Note that we do not apply the observation and state changes suggested by Yu et al. [16] anywhere in this paper. This is because these changes were not implemented as a wrapper on top of SMAC, but instead by modifying SMAC directly, leading to the environment code becoming unmanageably complicated.

3 Experimental Details

In this section we describe extra details of the experiments run as part of the paper.

3.1 Stochasticity

This section describes details of the experiments run in Section 5.1 in the main paper. Both the closed-loop and open-loop algorithms were based on the MAPPO implementation from Sun et al. [13]. The code used for these experiments can be found here. Both the open-loop and closed-loop algorithms use the same neural network architecture as in [13]. Both were also provided with access to the available actions mask of the environment and conditioned the critic on the state. We used the hyperparameters from Sun et al. [13], with the exception of the actor's learning rate, which we set to 0.0005 and decayed linearly throughout training. This is because learning rate decay has been found to be important for bounding PPO's policy updates [12]. Full hyperparameters can be found in Table 5. Full results are shown in Figure 1.

3.2 SMAC Feature Inferrability & Relevance

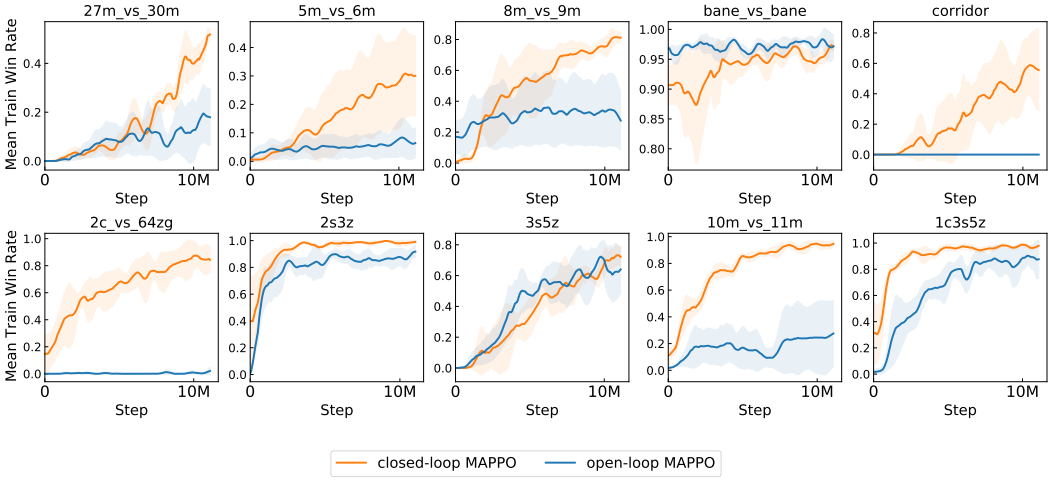
This section describes details of the experiments run in Section 5.2 and Section 7.3 in the main paper. The code used for these experiments can be found here.

For each scenario, we had 3 QMIX policies trained using the implementation and hyperparameters from Hu et al. [3], each with a different network and SMAC initialization. Training results for the policies for SMAC are shown in Figure 2. Each policy constitutes a seed and is used to collect a dataset of episodes for the feature-relevance experiment on the scenario. We call these policies *expert policies*. QMIX hyperparameters used are given in Table 4. A dataset consists of two folds: 8192 episodes used for training and 4096 episodes used for evaluation.

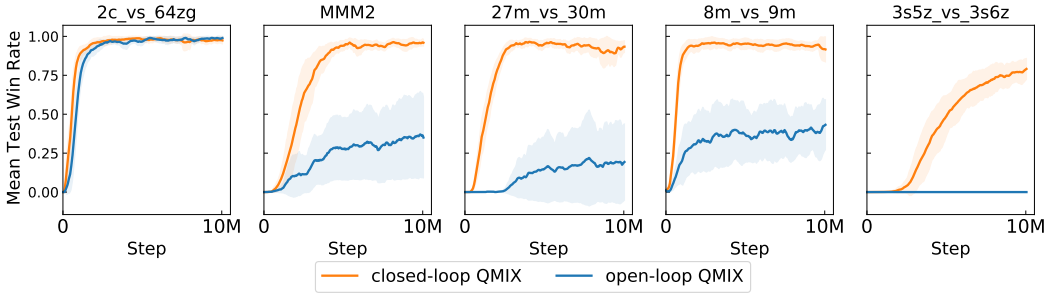
For a given mask, the experiment then consists of training a new QMIX network, termed *regression network*, whose input is trajectories with observations and states masked according to the mask, and whose task is to predict the Q-values output by the expert policy on the unmasked trajectories.

The *nothing* mask does not apply any effect on the observations and states in a trajectory. Otherwise, a mask, say 'health (ally)', masks the health feature of every ally in the observation of each agent (except its own health) and masks the health feature of all the units controlled by QMIX in the state. Table 1 shows the feature sets zeroed out for different masks.

The regression network has the same architecture as the expert network. We use the mean squared error (MSE) as a loss function and optimise the network via Adam with a batch size of 512 episodes and learning rate equal to 0.005. (The other Adam hyper-parameters are the default PyTorch values.)



(a) MAPPO open-loop and closed-loop results



(b) QMIX open-loop and closed-loop results

Figure 1: Plot of selected SMAC scenarios treated as an open-loop planning problem by limiting the observation to the current timestep and agent ID. Plots show the mean win rate and standard deviation across 3 training seeds for MAPPO and QMIX.

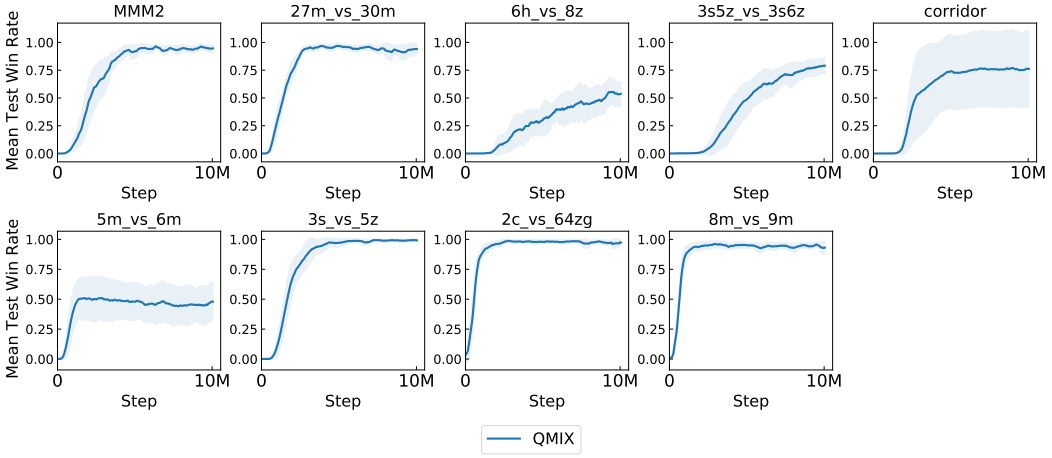


Figure 2: QMIX training results on SMAC

Table 1: Masked features for each setting of the feature quality experiment. ✓means a feature is masked and ✗means a feature is not masked.

Mask	Ally						Enemy				
	Health	Shield	x	y	Distance	Actions	Health	Shield	x	y	Distance
everything	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
nothing	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
health (ally)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
shield (ally)	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
distance (ally)	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗
health and shield (ally)	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
actions only	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
all except actions	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
ally all	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
health (enemy)	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
shield (enemy)	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
distance (enemy)	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
enemy all	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓

Training is performed with early stopping according to the validation fold with an evaluation every 5 epochs and a patience of 10 tries (i.e. training is stopped when the MSE on the validation fold does not decrease in 50 consecutive epochs and performance at the best epoch is retained). Models for all masks and scenarios hit early stopping and trained for about 200 epochs on average, except a few ones which trained to a limit of 500 epochs.

All hyper-parameters were tuned on a few different scenarios from both SMAC and SMACv2 to minimise validation MSE using datasets and expert policies not used for the reported results. The sizes of the training and validation folds are respectively 1.6 times and 0.8 times the size of the QMIX replay buffer (5000 episodes) and have been chosen to be large enough to minimize validation MSE while keeping experiments practical.

Figure 3 shows the full results of the feature quality experiments for SMAC and Figure 4 shows the full results for SMACv2. Table 3 shows summary results for the *everything* and *nothing* masks for SMACv2 and Table 2 shows the same data for SMAC.

Experiments for this section were conducted on 80-core CPU machines with NVIDIA GeForce RTX 2080 Ti or Tesla V100-SXM2-16GB GPUs. A single (scenario, seed, mask) combination took around 1 hour to train. In total, the experiments in this section took about 1500 GPU hours.

3.3 SMACv2 Runs

Here we describe the hyperparameter and training procedure used in section 7.1. As in previous experiments, we used the implementation by Hu et al. [3] for QMIX and by Sun et al. [13] for MAPPO. These implementations have both achieved very strong results on SMAC. We therefore tuned the hyperparameters by taking deviations from these for a few key parameters. For MAPPO, we tuned the actor's learning rate and the clipping range. Additionally, we added linear learning rate decay to the MAPPO implementation because this has been shown to be important to bound the policy update[12]. For QMIX, we tuned the ϵ-annealing time and λ in the eligibility trace $Q(\lambda)$. All other hyperparameters, including neural network architecture, were unchanged from the implementations mentioned previously.

The MAPPO code for these experiments can be found here, and the QMIX code here. The QMIX code is distributed under the Apache license, and the MAPPO code under the MIT license. The only differences in these branches and the implementations used for previous experiments on SMAC are important to changing the environment from SMAC to SMACv2.

The open-loop policy is identical to MAPPO, except the policy receives as input only the timestamp and agent ID, as in Section 5.1.

We tuned hyperparameters by running each set of hyperparameters on all scenarios and then choosing the best results. The grids of hyperparameters for QMIX and MAPPO can be found in Table 7 and 8. For IPPO, we used mostly the same hyperparameters as MAPPO. For QPLEX we used the hyperparameters from [15]. All algorithms were trained for 10M environment steps with evaluations

Table 2: Mean Q values in the different feature quality experiments on SMAC scenarios

Map	Mask	$\bar{Q}$	ϵ_{rmse}	$\frac{\epsilon_{\text{rmse}}}{\bar{Q}}$	ϵ_{abs}	$\frac{\epsilon_{\text{rmse}}^{\text{mask}} - \epsilon_{\text{rmse}}^{\text{nothing}}}{\bar{Q}}$
corridor	everything	7.0 ± 0.08	0.55 ± 0.08	0.078 ± 0.002	0.39 ± 0.05	0.042 ± 0.001
	ally_all_except_actions		0.34 ± 0.09	0.047 ± 0.005	0.21 ± 0.04	0.01 ± 0.008
	ally_health_and_shield		0.29 ± 0.07	0.041 ± 0.003	0.19 ± 0.03	0.005 ± 0.006
	ally_health		0.29 ± 0.05	0.0413 ± 0.0007	0.19 ± 0.02	0.005 ± 0.004
	ally_distance		0.27 ± 0.03	0.038 ± 0.002	0.174 ± 0.009	0.001 ± 0.001
	ally_shield		0.26 ± 0.02	0.037 ± 0.003	0.166 ± 0.005	0.0004 ± 0.0003
	ally_all		0.4 ± 0.2	0.06 ± 0.01	0.28 ± 0.09	0.02 ± 0.02
	enemy_health_and_shield		0.29 ± 0.05	0.0414 ± 0.0005	0.2 ± 0.03	0.005 ± 0.004
	enemy_health		0.3 ± 0.04	0.043 ± 0.003	0.21 ± 0.02	0.006 ± 0.001
	enemy_distance		0.32 ± 0.04	0.046 ± 0.002	0.203 ± 0.009	0.009 ± 0.001
	enemy_shield		0.25 ± 0.03	0.036 ± 0.003	0.161 ± 0.005	-0.0001 ± 0.0008
	enemy_all		0.42 ± 0.09	0.059 ± 0.003	0.29 ± 0.06	0.022 ± 0.006
	nothing		0.25 ± 0.02	0.037 ± 0.003	0.164 ± 0.006	
3s5z_vs_3s6z	everything	6.97 ± 0.01	0.61 ± 0.01	0.087 ± 0.001	0.418 ± 0.006	0.046 ± 0.003
	ally_all_except_actions		0.34 ± 0.01	0.049 ± 0.002	0.193 ± 0.01	0.0083 ± 0.0009
	ally_health_and_shield		0.326 ± 0.01	0.047 ± 0.002	0.182 ± 0.009	0.006 ± 0.001
	ally_health		0.333 ± 0.006	0.0479 ± 0.0008	0.188 ± 0.001	0.007 ± 0.001
	ally_distance		0.29 ± 0.02	0.041 ± 0.003	0.16 ± 0.01	-0.0 ± 0.003
	ally_shield		0.299 ± 0.007	0.043 ± 0.001	0.167 ± 0.004	0.0018 ± 0.0004
	ally_all		0.53 ± 0.02	0.076 ± 0.003	0.35 ± 0.02	0.035 ± 0.004
	enemy_health_and_shield		0.38 ± 0.02	0.055 ± 0.003	0.25 ± 0.02	0.014 ± 0.002
	enemy_health		0.36 ± 0.01	0.051 ± 0.002	0.22 ± 0.01	0.01 ± 0.001
	enemy_distance		0.32 ± 0.01	0.046 ± 0.002	0.18 ± 0.008	0.005 ± 0.001
	enemy_shield		0.302 ± 0.004	0.0434 ± 0.0007	0.188 ± 0.006	0.0023 ± 0.0008
	enemy_all		0.5 ± 0.04	0.071 ± 0.007	0.33 ± 0.04	0.03 ± 0.006
	nothing		0.286 ± 0.008	0.041 ± 0.001	0.161 ± 0.007	
5m_vs_6m	everything	8.1 ± 0.08	1.71 ± 0.08	0.21 ± 0.02	1.24 ± 0.09	0.073 ± 0.007
	ally_all_except_actions		1.17 ± 0.04	0.15 ± 0.02	0.76 ± 0.06	0.007 ± 0.004
	ally_health_and_shield		1.13 ± 0.05	0.14 ± 0.02	0.71 ± 0.05	0.002 ± 0.002
	ally_health		1.13 ± 0.05	0.14 ± 0.02	0.72 ± 0.05	0.001 ± 0.006
	ally_distance		1.15 ± 0.05	0.15 ± 0.02	0.73 ± 0.05	0.004 ± 0.004
	ally_shield		1.12 ± 0.07	0.14 ± 0.02	0.71 ± 0.05	0.001 ± 0.002
	ally_all		1.24 ± 0.06	0.16 ± 0.02	0.83 ± 0.09	0.014 ± 0.007
	enemy_health_and_shield		1.28 ± 0.03	0.16 ± 0.02	0.81 ± 0.03	0.021 ± 0.001
	enemy_health		1.27 ± 0.05	0.16 ± 0.02	0.82 ± 0.06	0.019 ± 0.004
	enemy_distance		1.22 ± 0.02	0.15 ± 0.02	0.82 ± 0.02	0.013 ± 0.007
	enemy_shield		1.11 ± 0.06	0.14 ± 0.02	0.7 ± 0.06	-0.0 ± 0.003
	enemy_all		1.44 ± 0.06	0.18 ± 0.02	0.99 ± 0.07	0.04 ± 0.009
	nothing		1.11 ± 0.05	0.14 ± 0.02	0.71 ± 0.04	
8m_vs_9m	everything	12.3 ± 0.07	0.75 ± 0.07	0.061 ± 0.006	0.46 ± 0.06	0.025 ± 0.004
	ally_all_except_actions		0.6 ± 0.04	0.049 ± 0.004	0.32 ± 0.04	0.013 ± 0.003
	ally_health_and_shield		0.53 ± 0.05	0.043 ± 0.005	0.27 ± 0.04	0.007 ± 0.002
	ally_health		0.56 ± 0.03	0.045 ± 0.002	0.29 ± 0.02	0.01 ± 0.004
	ally_distance		0.49 ± 0.06	0.04 ± 0.005	0.27 ± 0.05	0.0039 ± 0.0006
	ally_shield		0.47 ± 0.02	0.038 ± 0.002	0.27 ± 0.02	0.002 ± 0.003
	ally_all		0.73 ± 0.07	0.06 ± 0.007	0.43 ± 0.06	0.024 ± 0.004
	enemy_health_and_shield		0.55 ± 0.04	0.045 ± 0.004	0.31 ± 0.03	0.009 ± 0.002
	enemy_health		0.55 ± 0.05	0.045 ± 0.004	0.32 ± 0.03	0.009 ± 0.001
	enemy_distance		0.46 ± 0.07	0.038 ± 0.006	0.27 ± 0.04	0.002 ± 0.002
	enemy_shield		0.42 ± 0.05	0.034 ± 0.005	0.23 ± 0.04	-0.001 ± 0.002
	enemy_all		0.64 ± 0.06	0.052 ± 0.006	0.38 ± 0.05	0.016 ± 0.003
	nothing		0.44 ± 0.07	0.036 ± 0.006	0.24 ± 0.04	
27m_vs_30m	everything	9.6 ± 0.04	0.49 ± 0.04	0.051 ± 0.006	0.33 ± 0.03	0.016 ± 0.002
	ally_all_except_actions		0.47 ± 0.05	0.049 ± 0.007	0.28 ± 0.04	0.014 ± 0.003
	ally_health_and_shield		0.44 ± 0.07	0.046 ± 0.009	0.26 ± 0.05	0.01 ± 0.003
	ally_health		0.45 ± 0.06	0.047 ± 0.007	0.26 ± 0.04	0.011 ± 0.002
	ally_distance		0.35 ± 0.06	0.037 ± 0.008	0.21 ± 0.04	0.001 ± 0.001
	ally_shield		0.35 ± 0.02	0.036 ± 0.004	0.2 ± 0.02	0.001 ± 0.003
	ally_all		0.5 ± 0.06	0.052 ± 0.008	0.32 ± 0.05	0.016 ± 0.003
	enemy_health_and_shield		0.46 ± 0.04	0.048 ± 0.006	0.31 ± 0.03	0.012 ± 0.004
	enemy_health		0.5 ± 0.05	0.052 ± 0.007	0.34 ± 0.04	0.0166 ± 0.0003
	enemy_distance		0.41 ± 0.03	0.042 ± 0.005	0.25 ± 0.03	0.007 ± 0.003
	enemy_shield		0.35 ± 0.06	0.036 ± 0.007	0.21 ± 0.03	0.0009 ± 0.0008
	enemy_all		0.53 ± 0.02	0.055 ± 0.004	0.37 ± 0.02	0.02 ± 0.002
	nothing		0.34 ± 0.05	0.036 ± 0.006	0.21 ± 0.04	
2c_vs_64zg	everything	7.72 ± 0.02	0.56 ± 0.02	0.072 ± 0.002	0.42 ± 0.01	0.054 ± 0.003
	ally_all_except_actions		0.15 ± 0.04	0.02 ± 0.005	0.08 ± 0.02	0.0018 ± 0.001
	ally_health_and_shield		0.15 ± 0.04	0.02 ± 0.005	0.08 ± 0.02	0.002 ± 0.002
	ally_health		0.15 ± 0.03	0.02 ± 0.004	0.08 ± 0.02	0.0018 ± 0.0005
	ally_distance		0.13 ± 0.03	0.017 ± 0.003	0.07 ± 0.01	-0.0007 ± 0.0004
	ally_shield		0.14 ± 0.02	0.018 ± 0.002	0.07 ± 0.01	0.0 ± 0.001
	ally_all		0.15 ± 0.04	0.019 ± 0.005	0.08 ± 0.02	0.002 ± 0.001
	enemy_health_and_shield		0.43 ± 0.006	0.056 ± 0.002	0.317 ± 0.004	0.038 ± 0.003
	enemy_health		0.426 ± 0.006	0.0553 ± 0.001	0.314 ± 0.006	0.037 ± 0.003
	enemy_distance		0.18 ± 0.02	0.024 ± 0.002	0.125 ± 0.009	0.006 ± 0.001
	enemy_shield		0.14 ± 0.03	0.018 ± 0.003	0.07 ± 0.01	0.0 ± 0.0008
	enemy_all		0.54 ± 0.02	0.07 ± 0.002	0.41 ± 0.01	0.052 ± 0.003
	nothing		0.14 ± 0.03	0.018 ± 0.004	0.07 ± 0.02	
6h_vs_8z	everything	7.5 ± 0.09	0.87 ± 0.09	0.12 ± 0.02	0.65 ± 0.09	0.07 ± 0.01
	ally_all_except_actions		0.43 ± 0.02	0.059 ± 0.008	0.31 ± 0.03	0.007 ± 0.002
	ally_health_and_shield		0.41 ± 0.03	0.056 ± 0.009	0.29 ± 0.03	0.0034 ± 0.0007
	ally_health		0.41 ± 0.03	0.055 ± 0.008	0.28 ± 0.03	0.003 ± 0.002
	ally_distance		0.41 ± 0.03	0.056 ± 0.009	0.29 ± 0.03	0.004 ± 0.001
	ally_shield		0.38 ± 0.03	0.051 ± 0.008	0.26 ± 0.03	-0.001 ± 0.001
	ally_all		0.49 ± 0.01	0.067 ± 0.007	0.36 ± 0.02	0.014 ± 0.002
	enemy_health_and_shield		0.45 ± 0.02	0.06 ± 0.008	0.32 ± 0.02	0.008 ± 0.001
	enemy_health		0.46 ± 0.03	0.063 ± 0.01	0.33 ± 0.03	0.011 ± 0.001
	enemy_distance		0.47 ± 0.03	0.064 ± 0.009	0.33 ± 0.03	0.012 ± 0.0004
	enemy_shield		0.39 ± 0.02	0.053 ± 0.008	0.27 ± 0.03	0.001 ± 0.002
	enemy_all		0.55 ± 0.02	0.075 ± 0.008	0.4 ± 0.02	0.0224 ± 0.001
	nothing		0.38 ± 0.03	0.052 ± 0.009	0.27 ± 0.03	
3s_vs_5z	everything	8.1 ± 0.04	0.3 ± 0.04	0.037 ± 0.007	0.2 ± 0.03	0.012 ± 0.003
	ally_all_except_actions		0.19 ± 0.02	0.024 ± 0.003	0.1 ± 0.02	-0.0014 ± 0.0008
	ally_health_and_shield		0.17 ± 0.02	0.021 ± 0.002	0.081 ± 0.01	-0.004 ± 0.001
	ally_health		0.17 ± 0.03	0.021 ± 0.004	0.07 ± 0.01	-0.004 ± 0.002
	ally_distance		0.2 ± 0.02	0.024 ± 0.002	0.09 ± 0.02	-0.0 ± 0.004
	ally_shield		0.18 ± 0.02	0.022 ± 0.002	0.083 ± 0.007	-0.003 ± 0.003
	ally_all		0.3 ± 0.08	0.04 ± 0.01	0.17 ± 0.05	0.012 ± 0.007
	enemy_health_and_shield		0.26 ± 0.03	0.032 ± 0.004	0.17 ± 0.03	0.007 ± 0.001
	enemy_health		0.24 ± 0.03	0.03 ± 0.005	0.15 ± 0.03	0.005 ± 0.002
	enemy_distance		0.19 ± 0.02	0.024 ± 0.003	0.09 ± 0.02	-0.0015 ± 0.0006
	enemy_shield		0.24 ± 0.02	0.03 ± 0.004	0.13 ± 0.01	0.0049 ± 0.0002
	enemy_all		0.26 ± 0.03	0.033 ± 0.004	0.17 ± 0.02	0.008 ± 0.001
	nothing		0.2 ± 0.02	0.025 ± 0.003	0.08 ± 0.01	
MMM2	everything	9.4 ± 0.1	0.7 ± 0.1	0.08 ± 0.02	0.47 ± 0.09	0.039 ± 0.008
	ally_all_except_actions		0.41 ± 0.07	0.04 ± 0.01	0.23 ± 0.05	0.007 ± 0.003
	ally_health_and_shield		0.38 ± 0.03	0.041 ± 0.005	0.22 ± 0.03	0.003 ± 0.002
	ally_health		0.39 ± 0.04	0.042 ± 0.006	0.21 ± 0.04	0.004 ± 0.002
	ally_distance		0.34 ± 0.05	0.036 ± 0.007	0.19 ± 0.04	-0.0015 ± 0.0007
	ally_shield		0.35 ± 0.06	0.038 ± 0.008	0.2 ± 0.04	-0.0001 ± 0.0009
	ally_all		0.6 ± 0.1	0.07 ± 0.02	0.36 ± 0.09	0.028 ± 0.008
	enemy_health_and_shield		0.43 ± 0.06	0.046 ± 0.008	0.29 ± 0.04	0.008 ± 0.001
	enemy_health		0.44 ± 0.05	0.047 ± 0.007	0.29 ± 0.04	0.0094 ± 0.0005
	enemy_distance		0.33 ± 0.05	0.036 ± 0.006	0.19 ± 0.04	-0.0022 ± 0.0009
	enemy_shield		0.33 ± 0.04	0.035 ± 0.006	0.18 ± 0.04	-0.002 ± 0.002
	enemy_all		0.59 ± 0.08	0.06 ± 0.01	0.39 ± 0.05	0.026 ± 0.004
	nothing		0.35 ± 0.05	0.038 ± 0.007	0.2 ± 0.04	

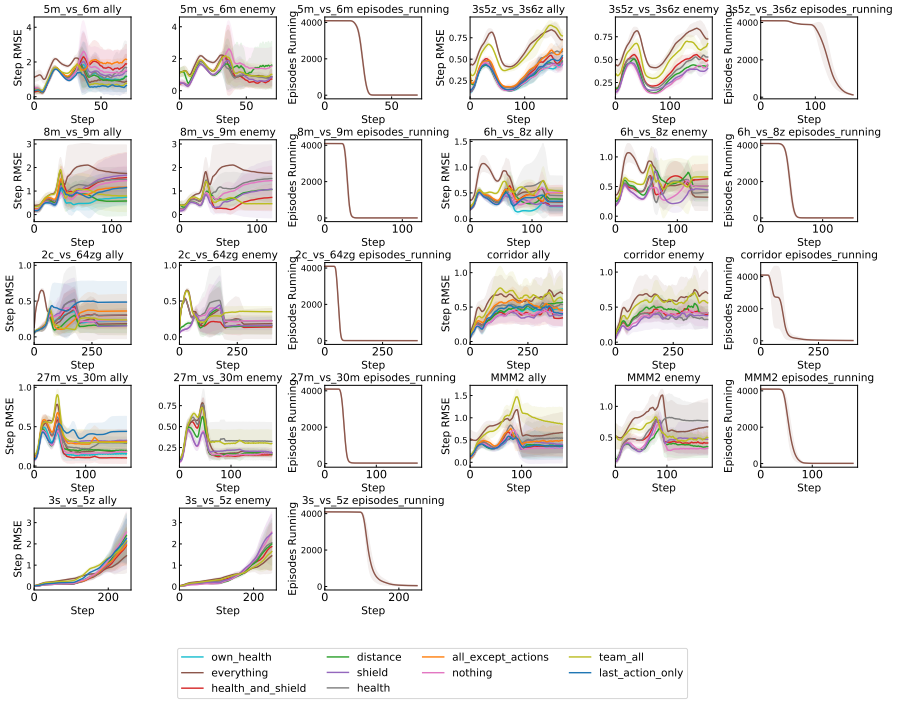


Figure 3: Results of the feature quality experiments for all tested SMAC scenarios not in the main paper.

Table 3: SMACv2 feature quality experiment results

Map	Mask	$\bar{Q}$	ϵ_{rmse}	$\frac{\epsilon_{\text{rmse}}}{\bar{Q}}$	ϵ_{abs}	$\frac{\epsilon_{\text{rmse}}^{\text{mask}} - \epsilon_{\text{rmse}}^{\text{nothing}}}{\bar{Q}}$
5_gen_protoss	everything	8.4 ± 0.05	1.5 ± 0.05	0.179 ± 0.006	1.18 ± 0.04	0.119 ± 0.004
	ally_all_except_actions		0.69 ± 0.02	0.082 ± 0.004	0.52 ± 0.01	0.022 ± 0.002
	ally_health_and_shield		0.62 ± 0.02	0.074 ± 0.004	0.45 ± 0.02	0.014 ± 0.002
	ally_health		0.54 ± 0.03	0.064 ± 0.005	0.39 ± 0.02	0.004 ± 0.001
	ally_distance		0.52 ± 0.04	0.062 ± 0.006	0.38 ± 0.03	0.0019 ± 0.0005
	ally_shield		0.52 ± 0.04	0.062 ± 0.006	0.38 ± 0.03	0.0018 ± 0.0008
	ally_all		0.85 ± 0.009	0.102 ± 0.004	0.656 ± 0.01	0.042 ± 0.002
	enemy_health_and_shield		0.79 ± 0.03	0.094 ± 0.004	0.6 ± 0.03	0.034 ± 0.003
	enemy_health		0.69 ± 0.05	0.082 ± 0.007	0.51 ± 0.04	0.022 ± 0.003
	enemy_distance		0.56 ± 0.03	0.067 ± 0.006	0.41 ± 0.03	0.0074 ± 0.0002
	enemy_shield		0.57 ± 0.03	0.068 ± 0.005	0.42 ± 0.02	0.008 ± 0.002
	enemy_all		0.96 ± 0.02	0.116 ± 0.004	0.74 ± 0.02	0.055 ± 0.002
	nothing		0.5 ± 0.03	0.06 ± 0.006	0.37 ± 0.03	
5_gen_terran	everything	8.2 ± 0.1	2.0 ± 0.1	0.243 ± 0.009	1.6 ± 0.1	0.148 ± 0.008
	ally_all_except_actions		1.0 ± 0.05	0.1221 ± 0.001	0.75 ± 0.04	0.028 ± 0.002
	ally_health_and_shield		0.82 ± 0.04	0.1 ± 0.001	0.6 ± 0.03	0.0052 ± 0.0009
	ally_health		0.83 ± 0.04	0.1007 ± 0.0003	0.6 ± 0.04	0.0061 ± 0.0007
	ally_distance		0.78 ± 0.04	$0.0945 \pm 9e-05$	0.56 ± 0.03	0.0002 ± 0.0007
	ally_shield		0.75 ± 0.02	0.092 ± 0.002	0.55 ± 0.02	-0.003 ± 0.001
	ally_all		1.18 ± 0.06	0.145 ± 0.002	0.92 ± 0.06	0.05 ± 0.003
	enemy_health_and_shield		0.98 ± 0.06	0.12 ± 0.003	0.72 ± 0.05	0.025 ± 0.003
	enemy_health		0.97 ± 0.07	0.118 ± 0.003	0.71 ± 0.05	0.023 ± 0.003
	enemy_distance		0.82 ± 0.04	0.1002 ± 0.0002	0.59 ± 0.03	0.0055 ± 0.0006
	enemy_shield		0.75 ± 0.05	0.091 ± 0.005	0.54 ± 0.04	-0.004 ± 0.004
	enemy_all		1.2 ± 0.1	0.151 ± 0.007	0.95 ± 0.09	0.057 ± 0.008
	nothing		0.78 ± 0.03	0.0945 ± 0.0007	0.56 ± 0.03	
5_gen_zerg	everything	7.2 ± 0.05	2.41 ± 0.05	0.33 ± 0.02	1.86 ± 0.04	0.176 ± 0.009
	ally_all_except_actions		1.36 ± 0.02	0.188 ± 0.008	0.99 ± 0.02	0.031 ± 0.002
	ally_health_and_shield		1.15 ± 0.03	0.16 ± 0.005	0.83 ± 0.03	0.002 ± 0.001
	ally_health		1.17 ± 0.03	0.161 ± 0.005	0.83 ± 0.02	0.0037 ± 0.0008
	ally_distance		1.18 ± 0.02	0.164 ± 0.007	0.85 ± 0.02	0.006 ± 0.001
	ally_shield		1.13 ± 0.02	0.157 ± 0.006	0.81 ± 0.02	-0.0005 ± 0.0003
	ally_all		1.46 ± 0.02	0.202 ± 0.01	1.09 ± 0.02	0.044 ± 0.002
	enemy_health_and_shield		1.31 ± 0.03	0.18 ± 0.008	0.95 ± 0.03	0.024 ± 0.002
	enemy_health		1.3 ± 0.02	0.179 ± 0.007	0.94 ± 0.02	0.022 ± 0.001
	enemy_distance		1.24 ± 0.01	0.173 ± 0.008	0.88 ± 0.02	0.014 ± 0.002
	enemy_shield		1.14 ± 0.02	0.159 ± 0.006	0.82 ± 0.02	0.0005 ± 0.0008
	enemy_all		1.67 ± 0.01	0.23 ± 0.01	1.24 ± 0.02	0.074 ± 0.008
	nothing		1.14 ± 0.02	0.157 ± 0.007	0.81 ± 0.02	

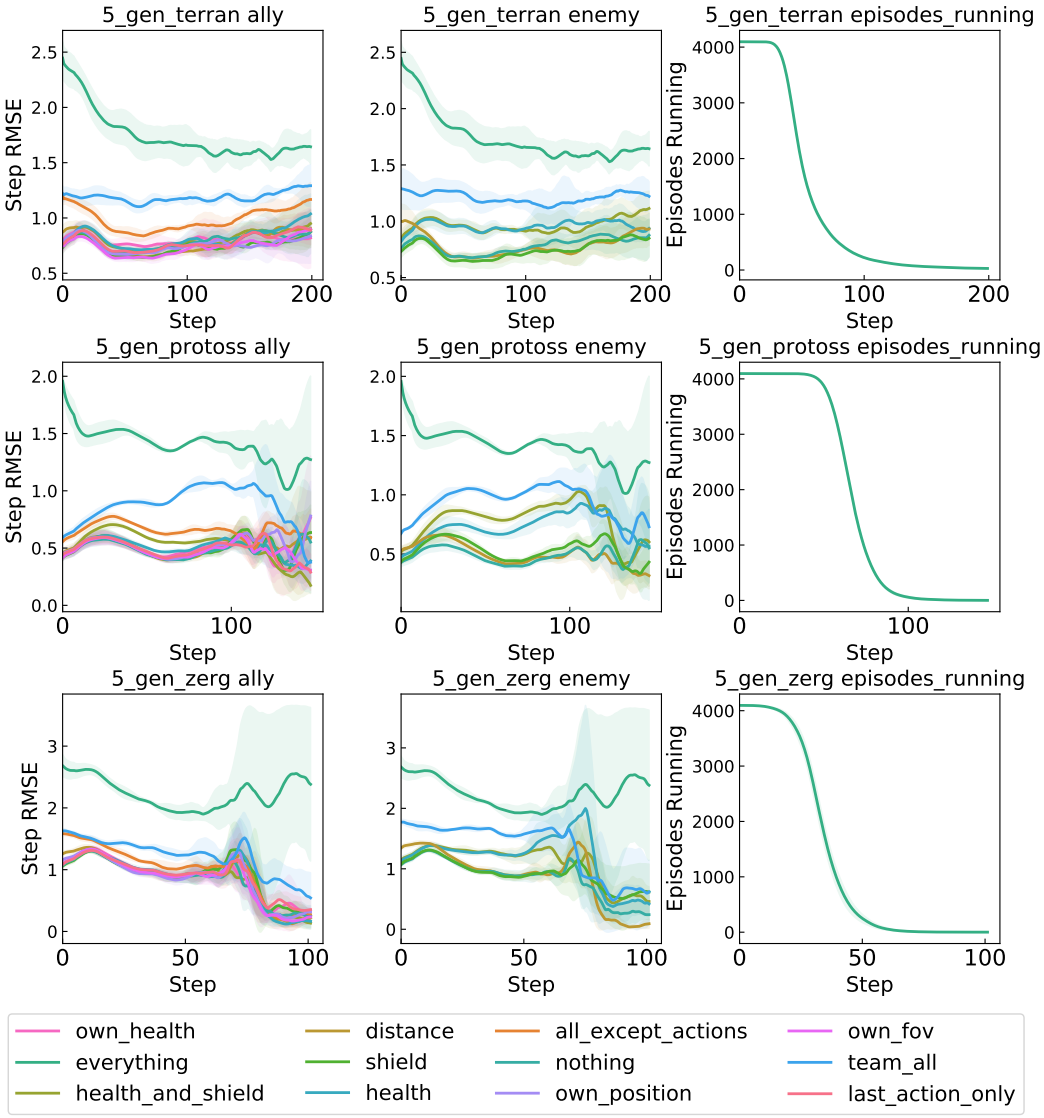


Figure 4: Feature Quality Experiments for the 5 and 10 unit SMACv2 scenarios.

of 20 episodes every 2k steps. The hyperparameters used for QMIX are given in Table 4, for MAPPO and IPPO in Table 5, and for QPLEX in Table 6.

3.4 EPO Runs

This section provides the implementation details for the Extended Partial Observability challenge in SMACv2 and outlines the corresponding training procedure.

For the EPO baselines in Section 7.2, we used the same hyperparameters as Section 7.1. Implementations of MAPPO and QMIX were also consistent with those in Section 7.1. Each training run and corresponding evaluations were conducted across 3 random seeds. Experiments for this section were conducted on 80-core CPU machines with NVIDIA GeForce RTX690 2080 Ti or Tesla V100-SXM2-16GB GPUs. Both MAPPO and QMIX experiments took between 36-48 hours to complete, each running on 8 GPUs, for 3 p values, across 3 seeds and 3 races.

Table 4: QMIX hyperparameters used for experiments. Parameters with (SMAC) or (SMACv2) after them denote that parameter setting was only used for SMAC or SMACv2 experiments respectively. These are the values in the corresponding configuration file in PyMarl[9, 3]. Mac is the code responsible for marshalling inputs to the neural networks, learner is the code used for learning and runner determines whether experience is collected in serial or parallel.

Parameter Name	Value
Action Selector	epsilon greedy
ϵ Start	1.0
ϵ Finish	0.05
ϵ Anneal Time	100000
Runner	parallel
Batch Size Run	4
Buffer Size	5000
Batch Size	128
Optimizer	Adam
t_{max}	10050000
Target Update Interval	200
Mac	n_mac
Agent	n_rnn
Agent Output Type	q
Learner	nq_learner
Mixer	qmix
Mixing Embed Dimension	32
Hypernet Embed Dimension	64
Learning Rate	0.001
λ (SMAC)	0.6
λ (SMACv2)	0.4

4 Environment Additional Details

4.1 SMACv2 Additional Details

In this section we describe the generation of teams in SMACv2. For each race, 3 different types of units are used. Each race has a special unit that should not be generated too often. For Protoss this is the colossus. This is a very powerful unit. If a team has many colossi, the battle will devolve into a war about who can use their colossi most effectively. Terran has the medivac unit. This is a healing unit that cannot attack the enemy and so is only spawned sparingly. Zerg has the baneling unit. This is a suicidal unit which deals area-of-effect damage to enemy units by rolling into them and exploding. In scenarios with many banelings, the agents learn to spread out and hide in the corners with the hope that the enemy banelings explode and the allies win by default. All of these special units are spawned with a probability of 10%. The other units used spawn with a probability of 45%. This is summarised in Table 9.

There are two changes to the observation space from SMAC. First, each agent observes their own field-of-view direction. Secondly, each agent observes their own position in the map as x- and y-coordinates. This is normalised by dividing by the map width and height respectively. The only change to the state from SMAC was to add the field-of-view direction of each agent to the state.

Additionally, we made one small change to the reward function in SMACv2. This was to fix a bug where the enemies healing can give allied units reward. There are more details available about this problem in the associated Github issue. SMACv2 also has an identical API to the original SMAC, allowing for very simple transition between the two frameworks.

The code for SMACv2 can be found in the Github repo, where there is a README detailing how to run the benchmark with random agents.

Table 5: MAPPO and IPPO hyperparameters used for the experiments on SMAC and SMACv2. Parameters with (SMAC) or (SMACv2) after them denote that parameter setting was only used for SMAC or SMACv2 experiments respectively.

Hyperparameter	Value
Action Selector	multinomial
Mask Before Softmax	True
Runner	parallel
Buffer Size	8
Batch Size Run	8
Batch Size	8
Target Update Interval	200
Learning Rate Actor (SMAC)	0.001
Learning Rate Actor (SMACv2)	0.0005
Learning Rate Critic	0.001
τ	0.995
λ	0.99
Agent Output Type	pi_logits
Learner	trust_region_learner
Critic (MAPPO)	centralV_rnn_critic
Critic (IPPO)	decentral_rnn_critic
Detach Every	20
Replace Every	None
Mini Epochs Actor	10
Mini Epochs Critic	10
Entropy Loss Coeff	0.0
Advantage Calc Method	GAE
Bootstrap Timeouts	False
Surrogate Clipped	True
Clip Range	0.1
Is Advantage Normalized	True
Observation Normalized	True
Use Popart	False
Normalize Value	False
Obs Agent Id	True
Obs Last Action	False

4.2 EPO Additional Details

In EPO, the first ally agent to observe an enemy is guaranteed to observe the usual SMAC features associated with that enemy. Upon an enemy being observed for the first time, by any agent on the ally team, a random binary draw occurs for all other agents (i.e., those that had not yet observed this particular enemy). Random tie breaking ensures only one agent is guaranteed to see the enemy should two or more observe it for the first time on the same timestep. The draw is weighted by a tunable environmnet parameter p corresponding to the probability of success. If the draw is successful for a particular agent, any future instances of the agent observing the enemy will occur as normal, without masking. If the draw is unsuccessful, that agent will not be able to observe the enemy on future timesteps, irrespective of whether or not it is within sight range. If the first agent to have observed the enemy dies, the next time the enemy falls within an ally agent's sight range that agent is guaranteed to see the enemy and the random draw occurs again for all other agents.

5 Analysis of Changes in SMACv2

To investigate the impact of the changes introduced in SMACv2, we perform three ablation studies. We focus only on the 5 unit scenarios, and only train MAPPO on the ablations because it is significantly faster to train than QMIX. We use the same hyperparameters and setup as in Section 7.1. We ablate different team compositions by using a single unit type for each of the non-special units in each

Table 6: QPLEX hyperparameters used for experiments on SMACv2.

Hyperparameter	Value
Action Selector	epsilon_greedy
Epsilon Start	1.0
Epsilon Finish	0.05
Epsilon Anneal Time	50000
Runner	parallel
Batch Size Run	4
Buffer Size	5000
Batch Size	128
Optimizer	adam
Target Update Interval	200
Agent Output Type	q
Learner	dmaq_qatten_learner
Double Q	True
Mixer	dmaq_qatten
Mixing Embed Dim	32
Hypernet Embed	64
Adv Hypernet Layers	1
Adv Hypernet Embed	64
Td Lambda	0.8
Num Kernel	4
Is Minus One	True
Is Adv Attention	True
Is Stop Gradient	True
N Head	4
Attend Reg Coef	0.001
State Bias	True
Mask Dead	False
Weighted Head	False
Nonlinear	False
Burn In Period	100
Name	qplex_qatten_sc2

Table 7: QMIX hyperparameter tuning grid

Hyperparameter	Values Tried
ϵ annealing time	$[100 \times 10^3, 500 \times 10^3]$
λ	[0.6, 0.8, 0.4]

Table 8: MAPPO hyperparameter tuning grid

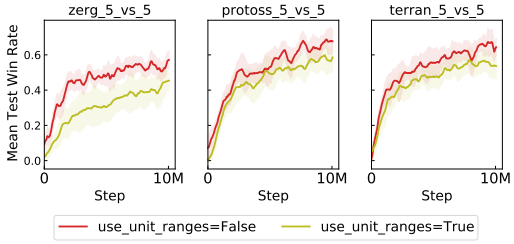
Hyperparameter	Value
Actor Learning Rate	$[0.0007, 0.0004, 0.0001]$
Clip Range	$[0.15, 0.05, 0.1]$

Table 9: Unit types used per-race in the SMACv2 scenarios.

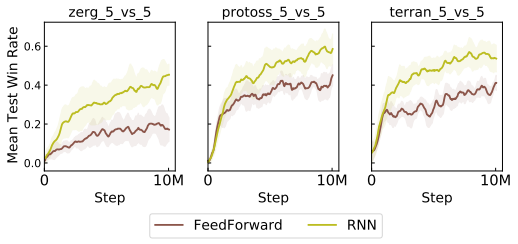
Race	Unit Types	Probability of Generation
Terran	Marine	0.45
	Marauder	0.45
	Medivac	0.1
Zerg	Zergling	0.45
	Baneling	0.1
	Hydralisk	0.45
Protoss	Stalker	0.45
	Zealot	0.45
	Colossus	0.1

(a) Position ablations

(b) Unit type ablations



(c) Field-of-view ablations



(d) Feed-forward vs RNN ablation

Figure 5: Ablation of SMACv2 unit type, field-of-view and starting position features. Figure 5d shows the performance of a feed-forward policy compared to an RNN baseline. The unit weights are relative to a fixed order of units. For Zerg this is zergling, hydralisk, baneling, for Terran it is marine, marauder, medivac and for Protoss it is stalkers, zealots and colossi. Plots show mean and standard deviation across three seeds.

race. We investigate random start positions by only using the *surround* or *reflect* scenarios and by removing the random start positions entirely. We also ablate the effect of using the true unit ranges. Additionally, we investigate partial observability by comparing feed-forward and recurrent networks.

The results of these experiments are shown in Figure 5. The position ablations show that stochastic starting positions and hence stochasticity itself contributes greatly to the challenge of SMACv2. Without random start positions, MAPPO achieves win rates of close to 100%. Both *surround* and *reflect* scenarios have a similar win rate, suggesting similar difficulty. There does not appear to be additional difficulty from combining the two. This is logical as the scenarios can be disambiguated using start positions at the beginning of the episode.

Our experiments of unit type ablations, shown in Figure 5b, indicate that unit variety significantly impacts the difficulty of the tasks. All three races show large differences between the easiest unit distribution and the baseline, suggesting that a diverse range of units contributes to SMACv2's difficulty. In both the Zerg and Protoss scenarios, the melee-only scenarios are easier than the ablation with ranged units. From observing episodes, the enemy AI tends to aggressively pursue allied units,

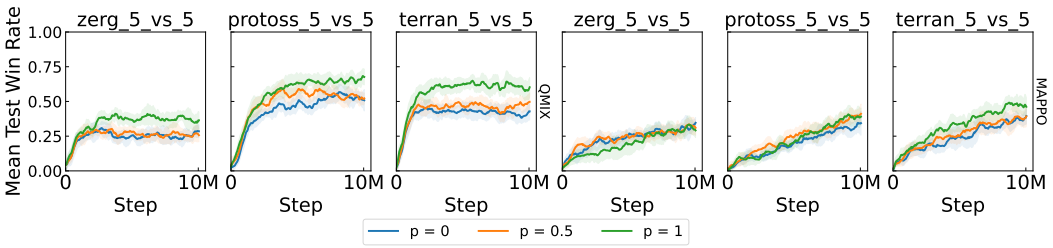


Figure 6: Comparison of the mean test win rate when $p = 0, 0.5, 1$ for QMIX (left) and MAPPO (right) on SMACv2. Here, the available action mask is still present.

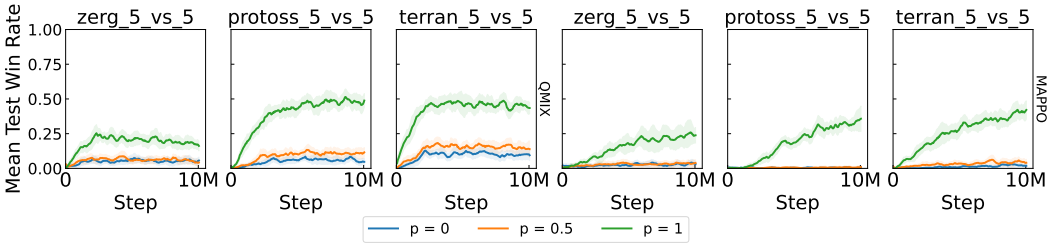


Figure 7: Comparison of mean test win rate when $p = 0, 0.5, 1$ with 5 agents against 5 enemy units, for QMIX (left) and MAPPO (right). Here, the available actions mask has been removed.

which allows the allies to lure it out of position. This is easier to exploit with melee units than ranged ones. Terran scenarios have no melee units, but the marine scenarios are slightly easier. Overall these results show that generalising to different unit types is a significant part of the challenge in SMACv2.

Our field-of-view ablations, shown in Figure 5c, compare the fixed unit sight and attack ranges to the SMACv2 versions. There is an increase in difficulty from varying the unit ranges, but this effect is small. This suggests that the difficulty of SMACv2 is better explained by the diversity of start positions and unit types than the change to the true sight and attack ranges.

Finally, to evaluate the effect that the changes had on the partial observability of SMACv2, we compare a feedforward network with the performance of the RNN baseline, shown in Figure 5d. There is a significant decrease in performance from using the feedforward network. However, the feedforward network can still learn reasonable policies. This suggests that although being able to resolve partial observability is useful in SMACv2, the primary difficulty consists in generalising to a range of micro-management scenarios. This is in contrast to the extended partial observability challenge, where resolving partial observability through communication is the primary difficulty. The zerg scenario has a larger relative performance difference. Partial observability may be more of a problem here because the splash damage done to allies by banelings creates an incentive for allies to spread out more.

Overall, the results in Figures 5a and 5b show that current MARL methods struggle with the increased stochasticity due to map randomisation. To solve this problem, agents must be able to use their observations to make inferences about relevant state or joint observation features in the Dec-POMDP in more general settings. Current inference capabilities could be improved by research on more effective sequence models, such as specialised transformers [14, 6, 1]. The work in this area either applies transformer models to traditional RL methods [5], or focuses on paradigms similar to upside-down RL [10, 1]. While the latter work shows promising generalisation [8], it is mostly in the offline setting. Hu et al. [4] demonstrate a promising sequence model that can be applied to any MARL algorithm, but only demonstrate their work on marine units, and rely on the Dec-POMDP having a certain structure. Expanding such work to handle more diverse scenarios is an interesting avenue of future work.

The code for the ablations can be found here. This is distributed under an MIT license. The changes on this branch enable easy running of the ablations and add associated environment configurations for this purpose.

Table 10: SMACv2 Scenarios with their number of allies, number of enemies, and unit types.

Scenario Name	Number of Allies	Number of Enemies	Unit Types
protoss_5_vs_5	5	5	Stalker, Zealot, Colossus
protoss_10_vs_10	10	10	Stalker, Zealot, Colossus
protoss_20_vs_20	20	20	Stalker, Zealot, Colossus
protoss_10_vs_11	10	11	Stalker, Zealot, Colossus
protoss_20_vs_23	20	23	Stalker, Zealot, Colossus
terran_5_vs_5	5	5	Marine, Marauder, Medivac
terran_10_vs_10	10	10	Marine, Marauder, Medivac
terran_20_vs_20	20	20	Marine, Marauder, Medivac
terran_10_vs_11	10	11	Marine, Marauder, Medivac
terran_20_vs_23	20	23	Marine, Marauder, Medivac
zerg_5_vs_5	5	5	Zergling, Hydralisk, Baneling
zerg_10_vs_10	10	10	Zergling, Hydralisk, Baneling
zerg_20_vs_20	20	20	Zergling, Hydralisk, Baneling
zerg_10_vs_11	10	11	Zergling, Hydralisk, Baneling
zerg_20_vs_23	20	23	Zergling, Hydralisk, Baneling

6 Limitations and Broader Impact

The main limitation of SMACv2 is that it is confined to scenarios within the game of StarCraft II. This is an environment which, while complex, cannot represent the dynamics of all multi-agent tasks. Evaluation of MARL algorithms therefore should not be limited to one benchmark, but should target variety with a range of tasks.

Whilst the scenarios in SMACv2 involve battles between two armies of units, only one side can be controlled by RL agents. It is technically possible to control two armies using two StarCraft II clients that communicate via LAN, which would allow to train two groups of decentralised agents against each other, e.g., via self-play. We leave the implementation of this functionality as future work.

SMACv2 aims to contribute to the development of MARL algorithms. As with any machine learning field, it is possible that improving the capabilities of these algorithms could lead to unethical uses. However, there are also many potential benefits to better cooperative AI, such as applications in automated driving among others. We believe that the potential benefits of developing more capable and cooperative AI outweigh the potential risks.

References

[1] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[2] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[3] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and SW Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*, 2021.

[4] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent rl via policy decoupling with transformers. In *International Conference on Learning Representations*, 2020.

[5] Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Boehmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on*

Machine Learning, volume 139 of *Proceedings of Machine Learning Research*, pages 4596–4606. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/iqbal21a.html.

[6] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[8] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[9] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2186–2188, 2019.

[10] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019.

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

[12] Mingfei Sun, Vitaly Kurin, Guoqing Liu, Sam Devlin, Tao Qin, Katja Hofmann, and Shimon Whiteson. You may not need ratio clipping in ppo. *arXiv preprint arXiv:2202.00079*, 2022.

[13] Mingfei Sun, Sam Devlin, Jacob Beck, Katja Hofmann, and Shimon Whiteson. Trust region bounds for decentralized ppo under non-stationarity. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, page 5–13, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2020.

[16] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.