Deep Multi-Agent Reinforcement Learning for Decentralised Continuous Cooperative Control

Christian A. Schroeder de Witt[*] [†] **Bei Peng**[*†] **Pierre-Alexandre Kamienny**[†]

Philip H. S. Torr[†] **Wendelin Böhmer**[†] **Shimon Whiteson**[†]

Abstract

Centralised training with decentralised execution (CTDE) is an important learning paradigm in multi-agent reinforcement learning (MARL). To make progress in CTDE, we introduce Multi-Agent MuJoCo (MAMuJoCo), a novel benchmark suite that, unlike StarCraft Multi-Agent Challenge (SMAC), the predominant benchmark environment, applies to continuous robotic control tasks. To demonstrate the utility of MAMuJoCo, we present a range of benchmark results on this new suite, including comparing the state-of-the-art actor-critic method MADDPG against two novel variants of existing methods. These new methods outperform MADDPG on a number of MAMuJoCo tasks. In addition, we show that, in these continuous cooperative MAMuJoCo tasks, value factorisation plays a greater role in performance than the underlying algorithmic choices. This motivates the necessity of extending the study of value factorisations from Q-learning to actor-critic algorithms.

1 Introduction

Reinforcement learning (RL) has shown promise in learning optimal control policies for a variety of single-agent robot control problems, ranging from idealised multi-joint simulations (Gu et al. 2016; Haarnoja et al. 2018) and complex grasping control problems (Kalashnikov et al. 2018; Andrychowicz et al. 2020), to modular robotics (Varshavskaya, Kaelbling, and Rus 2008; Kojcev et al. 2018). However, many real-world robot control tasks can be naturally framed as multiple decentralised collaborating agents. Cooperative robotic control tasks arise in autonomous aerial construction (Augugliaro et al. 2013, 2014), industrial manufacturing (Caccavale et al. 2008), and agricultural robotics (Shamshiri et al. 2018) and have so far received comparatively little attention from the deep RL community.

In cooperative robotic tasks, unlike in established multi-agent benchmarks, such as the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al. 2019), robotic actuators are usually continuous, so learning algorithms must scale to large continuous joint action spaces. In addition, while explicit communication between robotic actuators has been considered (Wang et al. 2018), practical applications frequently require fully *decentralised policies* for safety reasons, as communication cannot be guaranteed under all circumstances (Takadama et al. 2003). In such settings, agents need to coordinate based on alternative means, including implicit communication or, if available, common knowledge (Schroeder de Witt et al. 2019).

Fortunately, decentralised policies can often be trained in a centralised fashion in a simulator or laboratory. The framework of *centralised training with decentralised execution* (CTDE; Oliehoek, Spaan, and Nikos Vlassis 2008; Kraemer and Banerjee 2016) allows policy training to exploit extra information that is not available during execution in order to accelerate learning in such settings. Recently, significant progress has been made in developing MARL algorithms under CTDE for cooperative tasks with discrete action spaces (Foerster et al. 2018; Sunehag et al. 2018; Rashid et al. 2018; Son et al. 2019; Rashid et al. 2020). However, work on decentralised continuous control has received comparatively little attention and has been largely limited to deep deterministic policy gradient approaches (Lowe et al. 2017; Iqbal and Sha 2019; Li et al. 2019). We believe the lack of diverse continuous benchmarks is one factor limiting progress in continuous MARL.

Hence, in this paper, we take the first steps toward building decentralised continuous robotic benchmark tasks, to stimulate more progress in continuous MARL. We present *Multi-Agent MuJoCo* (MAMuJoCo), a new, comprehensive benchmark suite for CTDE that allows the study of decentralised coordination. Based on the popular single-agent MuJoCo benchmark suite from OpenAI Gym (Brockman et al. 2016), MAMuJoCo features a wide variety of novel robotic control tasks in which multiple agents within a single robot have to solve a task cooperatively (see Figure 1). The number of agents and level of partial observability in each task can be finely configured.

MAMuJoCo also includes scenarios with a larger and more flexible number of agents, which takes inspiration from modular robotics (Yim, Zhang, and Duff 2002; Kurokawa et al. 2008).

Specifically, in two scenarios, ManyAgent Swimmer (Figure 1A) and ManyAgent Ant (Figure 1K), one can configure an arbitrarily large number of agents, each controlling a consecutive segment of arbitrary length.

[*]Equal contribution. Correspondence to Christian Schroeder de Witt <cs@robots.ox.ac.uk>

[†]University of Oxford, UK

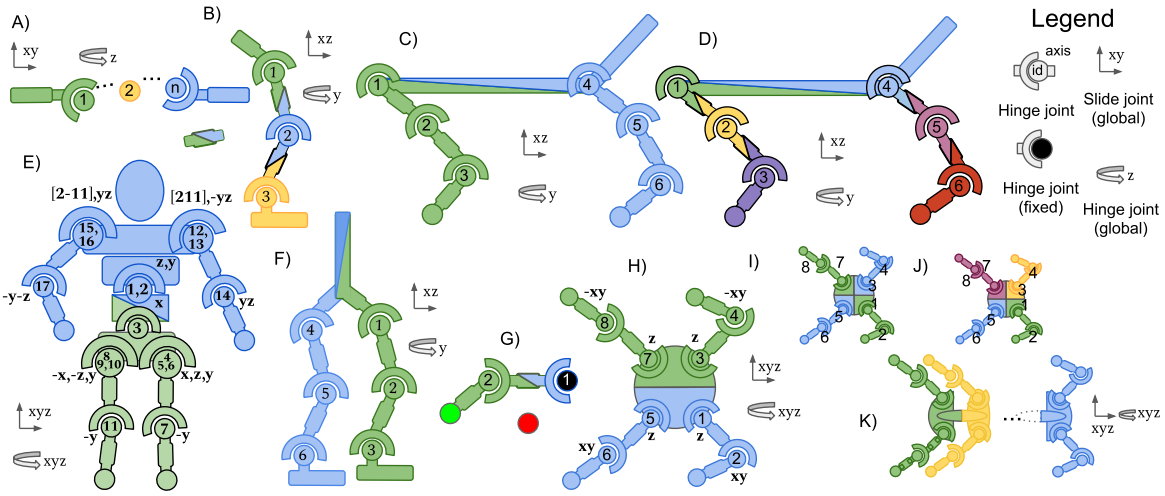


Figure 1: **Agent partitionings for MAMuJoCo environments:** A) Manyagent Swimmer, B) 3-Agent Hopper [3x1], C) 2-Agent HalfCheetah [2x3], D) 6-Agent HalfCheetah [6x1], E) 2-Agent Humanoid and 2-Agent HumanoidStandup (each [1x9,1x8]), F) 2-Agent Walker [2X3], G) 2-Agent Reacher [2x1], H) 2-Agent Ant [2x4], I) 2-Agent Ant Diag [2x4], J) 4-Agent Ant [4x2], and K) Manyagent Ant. Colours indicate agent partitionings. Each joint corresponds to a single controllable motor. Split partitions indicate shared body segments. Square brackets indicate [(number of agents) x (joints per agent)]. Joint IDs are in order of definition in the corresponding OpenAI Gym XML asset files (Brockman et al. 2016). Global joints indicate degrees of freedom of the center of mass of the composite robotic agent.

Furthermore, MAMuJoCo allows the definition of custom implicit communication channels (deformable tendons) and explicitly exposes common knowledge between agents. All of this makes MAMuJoCo uniquely suited to the study of decentralised coordination.

To demonstrate the utility of MAMuJoCo, we present a range of benchmark results on this new suite. These results compare the state-of-the-art actor-critic method MADDPG (Lowe et al. 2017) to two novel variants of existing methods. The first, COMIX, adapts the highly successful value-based method QMIX (Rashid et al. 2018) to cope with continuous actions. The second, FacMADDPG, introduces the factorisation that is key to QMIX into MADDPG's critic. Studying the relative performance between COMIX, MADDPG, and FacMADDPG in MAMuJoCo can help us answer an interesting research question: *What is the key to performance in such settings, the use of value-based methods instead of policy gradients, or the choice of factorisation of the joint Q-value function?* Previous work on this question (Bescuca 2019) is inconclusive due to the confounder that the policy gradient methods studied (COMA and Central-V (Foerster et al. 2018)) are on-policy, while the respective Q-learning method, QMIX, is off-policy with experience replay (Lin 1992a). Here however, COMIX, MADDPG, and FacMADDPG are all off-policy.

Our empirical results show that COMIX and FacMADDPG yield similar performance and that both significantly outperform MADDPG on a number of MAMuJoCo tasks. As COMIX and FacMADDPG use the same value factorisation as in QMIX, this suggests that value factorisation plays a greater role in performance in most continuous cooperative MAMuJoCo tasks than the underlying algorithmic choices. As the critic is not used for greedy action selection in actor-critic methods, this motivates the search for less restrictive (and potentially better) critic factorisations for actor-critic methods than that of QMIX.

2 Background

Dec-POMDPs. We consider a *fully cooperative multi-agent task* in which a team of cooperative agents choose sequential actions in a stochastic, partially observable environment. It can be modeled as a *decentralised partially observable Markov decision process* (Dec-POMDP Oliehoek, Amato et al. 2016), defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{U}, P, r, \mathcal{Z}, O, \rho, \gamma \rangle$. Here $\mathcal{N} := \{1, \dots, N\}$ denotes the set of N agents and $s \in \mathcal{S}$ describes the discrete or continuous state of the environment. The initial state $s_0 \sim \rho$ is drawn from distribution ρ, and at each time step t, all agents $a \in \mathcal{N}$ choose simultaneously discrete or continuous actions $u_t^a \in \mathcal{U}$, yielding the joint action $\boldsymbol{u}_t := \{u_t^a\}_{a=1}^N \in \mathcal{U}^N$. After executing the joint action $\boldsymbol{u}_t$ in state s_t, the next state $s_{t+1} \sim P(s_t, \boldsymbol{u}_t)$ is drawn from transition kernel P and the collaborative reward $r_t = r(s_t, \boldsymbol{u}_t)$ is returned to the team.

In a Dec-POMDP, the true state of the environment cannot be directly observed by the agents. Each agent $a \in \mathcal{N}$ draws an individual observation $z_t^a \in \mathcal{Z}, \boldsymbol{z}_t := \{z_t^a\}_{a=1}^N$, from the observation kernel $O(s_t, a)$. The history of an agent's observations and actions is denoted by $\tau_t^a \in \mathcal{T}_t := (\mathcal{Z} \times \mathcal{U})^t \times \mathcal{Z}$, and the set of all agents' histories is $\boldsymbol{\tau}_t := \{\tau_t^a\}_{a=1}^N$. Agent a chooses its actions with a decentralised policy $u_t^a \sim \pi(\cdot|\tau_t^a)$ based only on its individual history. The collaborating team of agents aims to learn a *joint pol-*

icy $\pi(\boldsymbol{u}|\boldsymbol{\tau}_t) := \prod_{a=1}^{N} \pi^a(u^a|\tau_t^a)$ that maximises their expected discounted return, $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in [0, 1)$ is a discount factor. This joint policy induces a joint action-value function Q^π that estimates the expected discounted return when the agents take joint action $\boldsymbol{u}_t$ with histories $\boldsymbol{\tau}_t$ in state s_t and then follow some joint policy π through $Q^\pi(s_t, \boldsymbol{\tau}_t, \boldsymbol{u}_t) := \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right]$.

CTDE. Even if safety, security, or other feasibility constraints require agent policies to be fully decentralised during execution, such policies are usually trained either in a laboratory or in simulation. In both cases, the paradigm of *centralised training with decentralised execution* (CTDE Kraemer and Banerjee 2016) allows the learning process to make use of extra state information. CTDE allows the learning algorithm to access all local action-observation histories $\boldsymbol{\tau}$ and global state s, as well as share gradients and parameters, but each agent's executed policy can condition only on its own action-observation history τ^a.

VDN and QMIX. Value decomposition networks (VDN Sunehag et al. 2018) and QMIX (Rashid et al. 2018) are two representative examples of value function factorisation (Koller and Parr 1999) that aim to efficiently learn a centralised but factored action-value function. They both work in cooperative MARL tasks with discrete actions, using CTDE. To ensure consistency between the centralised and decentralised polices, both VDN and QMIX factor the joint action-value function Q_{tot} into a sum of individual action-value functions Q_a,[1] one for each agent a, that condition only on individual action-observation histories:

$$Q_{tot}(s, \boldsymbol{\tau}, \boldsymbol{u}; \boldsymbol{\theta}, \phi) := f_\phi\big(s, \{Q_a(\tau^a, u^a; \theta^a)\}_{a=1}^N\big), \quad (1)$$

with f_ϕ chosen so as to ensure that Q_{tot} to be monotonic in each Q_a: $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in \mathcal{N}$, which is sufficient to guarantee that decentralised policies are consistent with their centralised counterpart. For QMIX, f_ϕ is given by a monotonic mixing network. Monotonicity can be guaranteed by non-negative mixing weights. These weights are generated by separate *hypernetworks* (Ha, Dai, and Le 2016), parameterised by ϕ, which condition on the full state s. This allows Q_{tot} to depend on the full state information in non-monotonic ways. VDN is a special case of QMIX, where $f = \sum_{a=1}^{N} Q_a(\tau^a, u^a; \theta^a)$ is a simple unparameterised sum over utilities, with no dependence on s. In both VDN and QMIX, the loss function is analogous to the standard Deep Q-Network loss (DQN Mnih et al. 2015) (see Appendix A.1), where Q is replaced by Q_{tot}. During execution, each agent selects actions greedily with respect to its own Q_a.

3 Multi-Agent MuJoCo

MAMuJoCo is a novel open source [2] benchmark for continuous cooperative multi-agent robotic control. Starting from the popular fully observable single-agent robotic MuJoCo (Todorov, Erez, and Tassa 2012) control suite included with OpenAI Gym (Brockman et al. 2016), we create a wide variety of novel scenarios in which multiple agents within a single robot have to solve a task cooperatively.

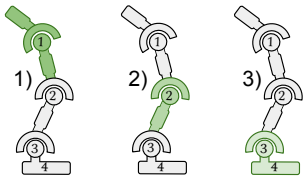


Figure 2: **Observations by distance for 3-Agent Hopper (as seen from agent 1).** Each corresponds to joints and body parts observable at 1) zero, 2) one unit, and 3) two unit graph distances from agent 1.

Multiple agents are introduced within a single robot as partial observability arises through latency, bandwidth, and noisy sensors in a single robot. Even if communication is free and instant when it works, we want policies that keep working even when communication channels within the robot malfunction. Without access to the exact full state, local decision rules become more important and introducing autonomous agents at individual decision points (e.g., each physical component of the robot) is reasonable and beneficial. This also makes it more robust to single-point failures (e.g., broken sensors) and more adaptive and flexible as new independent decision points (thus agents) may be added easily. Similar physical decompositions can be found in early behavior-based robotic research (Brooks 1986; Arkin 1989).

This design also offers important benefits. It facilitates comparisons to existing literature on both the fully observable single-agent domain (OpenAI 2020), as well as settings with low-bandwidth communication (Wang et al. 2018). More importantly, it allows for the study of novel MARL algorithms for decentralised coordination in isolation (scenarios with multiple robots may add confounding factors such as spatial exploration), which is currently a gap in the research literature.

MAMuJoCo also includes scenarios with a larger and more flexible number of agents, which takes inspiration from modular robotics (Yim, Zhang, and Duff 2002; Kurokawa et al. 2008). Compared to traditional robots, modular robots are more versatile, configurable, and scalable as it is easier to replace or add modules to change the degrees of freedom. We therefore develop two scenarios named ManyAgent Swimmer (see Figure 1A) and ManyAgent Ant (see Figure 1K), in which one can configure an arbitrarily large number of agents (within the memory limits), each controlling a consecutive segment of arbitrary length. This design is similar to many practical modular snake robots (Wright et al. 2012; Nakagaki et al. 2016), which mimic snake-like motion for diverse tasks such as navigating rough terrains, moving underwater, and urban search and rescue.

Single-robot multi-agent tasks in MAMuJoCo arise by first representing a given single robotic agent as a *body graph*, where vertices (joints) are connected by adjacent

[1] Strictly speaking, each Q_a is a *utility function* since by itself it does not estimate an expected return. We refer to Q_a as action-value function for simplicity.

[2] https://github.com/schroederdewitt/multiagent_mujoco

edges (body segments), as shown in Figure 1. We then partition the body graph into disjoint sub-graphs, one for each agent, each of which contains one or more joints that can be controlled. Note that in ManyAgent Swimmer (see Figure 1A) and ManyAgent Ant (see Figure 1K), the number of agents are not limited by the given single robotic agent.

Each agent's action space in MAMuJoCo is given by the joint action space over all motors controllable by that agent. For example, the agent corresponding to the green partition in 2-Agent HalfCheetah (Figure 1C) consists of three joints (joint ids 1, 2, and 3) and four adjacent body segments.

Each joint has an action space $[-1, 1]$, so the action space for each agent is a 3-dimensional vector with each entry in $[-1, 1]$.

For each agent a, observations are constructed in a two-stage process. First, we infer which body segments and joints are observable by agent a. Each agent can always observe all joints within its own sub-graph. A configurable parameter $k \geq 0$ determines the maximum graph distance to the agent's subgraph at which joints are observable (see Figure 2 for an example). Body segments directly attached to observable joints are themselves observable. The agent observation is then given by a fixed order concatenation of the representation vector of each observable graph element. Depending on the environment and configuration, representation vectors may include attributes such as position, velocity, and external body forces. In addition to joint and body-segment specific observation categories, agents can also be configured to observe the position and/or velocity attributes of the robot's central torso.

Restricting both the observation distance k, as well as limiting the set of observable element categories imposes partial observability. However, task goals remain unchanged from the single-agent variants (see Table 1 in the Appendix), except that the goals must be reached collaboratively by multiple agents: we simply repurpose the original single-agent reward signal as a team reward signal.

4 Algorithms

We now proceed to empirical analysis using MAMuJoCo. In this section, we briefly summarise the algorithms implemented in our new benchmark suite [3]. We find that, apart from DDPG (Lillicrap et al. 2015) and MADDPG, not many existing methods work off the shelf with continuous actions.

Fortunately, we show in this section how existing methods can be extended to continuous action spaces to make compelling contenders for this new benchmark.

IQL-CEM. A simple and natural approach to solving Dec-POMDPs is *independent Q-learning* (IQL) (Tan 1993), in which each agent a learns an individual action-value function Q_a independently, using DQN (see Appendix A.1). Each agent then selects actions greedily with respect to its own Q_a. In discrete action spaces, this operation can be performed efficiently through enumeration (unless the action space is extremely large).

In continuous action spaces, however, performing greedy action selection via enumeration is impossible.

We therefore utilise existing continuous Q-learning approaches in single-agent settings to extend IQL to continuous action spaces. Specifically, we implement IQL-CEM, which uses the *cross-entropy method* (CEM De Boer et al. 2005) to perform approximate greedy action selection. CEM is a sampling-based derivative-free heuristic search method that has been successfully used to find approximate maxima of nonconvex Q-networks in a number of single-agent robotic control tasks (Kalashnikov et al. 2018). We also benchmark IQL-NAF, which adds quadratic function constraints on each Q_a based on *normalized advantage functions* (NAF Gu et al. 2016; Amos, Xu, and Kolter 2017) to make maximisation easy.

In IQL-CEM, CEM is used by each agent to find an action that approximately optimises its per-agent Q-value. Specifically, CEM iteratively draws a batch of N random samples from a candidate distribution $\mathcal{D}_k$, e.g., a Gaussian, at each iteration k. The best $M < N$ samples (with the highest Q-values) are then used to fit a new Gaussian distribution $\mathcal{D}_{k+1}$, and this process repeats K times. We use a CEM hyperparameter configuration similar to Qt-Opt (Kalashnikov et al. 2018), where $N = 64$, $M = 6$, and $K = 2$.[4] Gaussian distributions are initialised with mean $\mu = 0$ and standard deviation $\sigma = 1$. Algorithm 1 in the appendix outlines the full CEM process for IQL-CEM.

MADDPG. *Multi-agent deep deterministic policy gradient* (MADDPG Lowe et al. 2017) is an extension of DDPG (Lillicrap et al. 2015) to multi-agent settings. It is an actor-critic method that works in both cooperative and competitive MARL tasks with discrete or continuous action spaces. MADDPG was originally designed for the general case of *partially observable stochastic games* (Kuhn 1953), in which it learns a separate actor and centralised critic for each agent such that agents can learn arbitrary reward functions (including conflicting rewards in competitive settings). Here we implement a version specific to Dec-POMDPs and consider continuous actions. We assume each agent a has a deterministic policy μ^a, parameterised by θ^a, with $\boldsymbol{\mu}(\boldsymbol{\tau}; \boldsymbol{\theta}) := \{\mu^a(\tau^a; \theta^a)\}_{a=1}^N$. For Dec-POMDPs, MADDPG learns a shared centralised critic $Q_a^{\boldsymbol{\mu}}(s, \boldsymbol{u}; \phi)$ for all agents that conditions on the full state s and the joint actions $\boldsymbol{u}$ of all agents. The policy gradient for θ^a is:

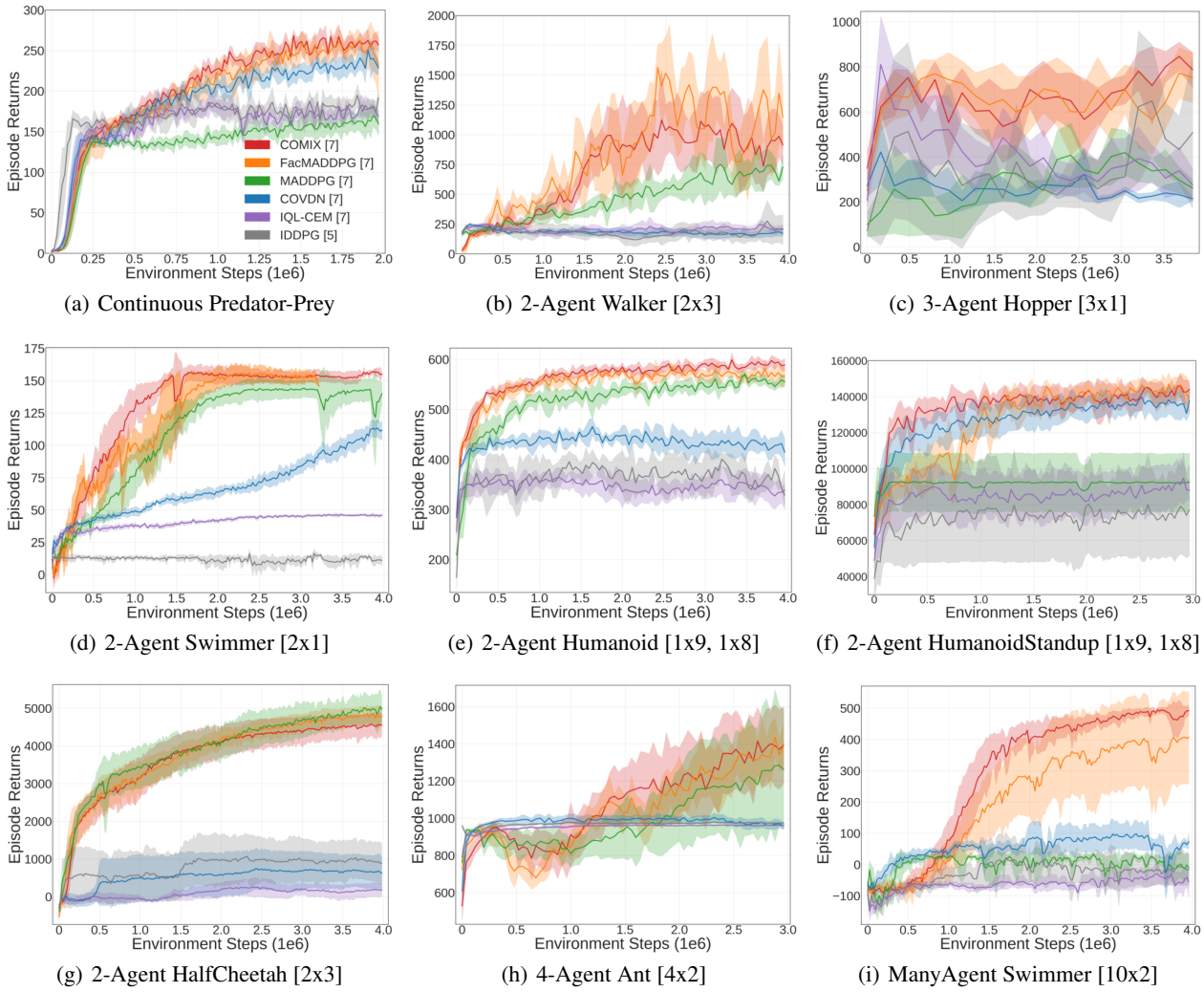
$$\nabla_{\theta^a} \mathcal{L}^{\mu}_{[\theta^a]} := -\mathbb{E}_{\mathcal{D}}\Big[\nabla_{\theta^a} \mu^a(\tau_t^a; \theta^a) \nabla_{u^a} Q_a^{\boldsymbol{\mu}}(s_t, \hat{\boldsymbol{u}}_t^a; \phi)\big|_{u^a = \mu^a(\tau_t^a)}\Big], \quad (2)$$

where $\hat{\boldsymbol{u}}_t^a := \{u_t^1, \ldots, u_t^{a-1}, u^a, u_t^{a+1}, \ldots, u_t^N\}$ and $s_t, \boldsymbol{u}_t, \boldsymbol{\tau}_t$ are sampled from a replay buffer $\mathcal{D}$. Each $Q_a^{\boldsymbol{\mu}}$ is trained by minimising an off-policy TD error loss (see Appendix B). Additionally, we benchmark independent DDPG (IDDPG), in which each agent learns a deterministic policy independently, using DDPG (Lillicrap et al. 2015).

COVDN and COMIX. We introduce COVDN and COMIX, which are novel variants of the popular value-based methods VDN and QMIX respectively, to cope with

[3] All code is available at http://github.com/oxwhirl/comix.

[4] We empirically find 2 iterations to suffice.



(a) Continuous Predator-Prey (b) 2-Agent Walker [2x3] (c) 3-Agent Hopper [3x1]

(d) 2-Agent Swimmer [2x1] (e) 2-Agent Humanoid [1x9, 1x8] (f) 2-Agent HumanoidStandup [1x9, 1x8]

(g) 2-Agent HalfCheetah [2x3] (h) 4-Agent Ant [4x2] (i) ManyAgent Swimmer [10x2]

Figure 3: Mean episode return for COMIX, FacMADDPG, MADDPG, COVDN, IQL-CEM, and IDDPG on Continuous Predator-Prey and different MAMuJoCo tasks. The mean across at least 5 seeds is plotted and the 95% confidence interval is shown shaded.

continuous actions. In COVDN and COMIX, we factor the joint action-value function Q_{tot} assuming additive and monotonic mixing, respectively. Both methods perform greedy selection of actions u^a with respect to utility functions Q_a for each agent a using CEM.

The centralised but factored Q_{tot} allows us to use CEM to sample actions for each agent independently and to use the individual utility function Q_a to guide the selection of maximal actions. Algorithm 1 in the appendix outlines the full CEM process used in both COVDN and COMIX. Algorithm 2 in the appendix outlines the full process for COMIX. In addition, we benchmark COVDN-NAF and COMIX-NAF, which add quadratic function constraints on each Q_a based on NAF, instead of using CEM to perform maximal action selection.

FacMADDPG. Learning a centralised critic conditioning on a large full state space (or joint agent observation space) can be difficult (Iqbal and Sha 2019). At the same time, it seems plausible that the factorisations employed by VDN and QMIX should not be restricted to value-based methods, but can be extended to actor-critic methods.

We therefore introduce FacMADDPG, a novel variant of MADDPG with an agent-specific factorisation that facilitates the learning of a centralised critic in Dec-POMDPs. In FacMADDPG, all agents share a single centralised critic that is

factored as

$$Q^{\boldsymbol{\mu}}(s, \boldsymbol{u}; \boldsymbol{\theta}, \phi) = g_\phi\big(s, \{Q_a(\tau^a, u^a; \theta^a)\}_{a=1}^N\big), \qquad (3)$$

where g_ϕ is a function represented by a monotonic mixing network. Although the monotonicity constrained on g_ϕ is no longer required as the critic is not used for greedy

action selection, FacMADDPG does impose monotonicity on g_ϕ in order to keep the factorisation comparable to the one employed by COMIX. We also find that FacMADDPG significantly outperforms an ablation without monotonicity constraints (see Section 6.2 and Appendix H). We believe exploring more forms of less restrictive (and potentially better) nonmonotonic factorisations for the centralised critic is an interesting avenue for future work.

5 Experimental Setup

Partially Observable Continuous Predator-Prey.

To validate the performance of the range of MARL algorithms discussed previously on an existing benchmark, we consider the mixed *simple tag* environment introduced by Leibo et al. (2017), which is a variant of the classic predator-prey game where we replaced the prey's movement with a hard-coded avoidance heuristic[5] (see Appendix E for further details on the environment).

Multi-Agent MuJoCo. All MAMuJoCo environments are configured according to its default configuration (other than 4-Agent Ant[6]), where each agent can observe only positions (not velocities) of its own body parts and at graph distances greater than zero. In ManyAgent Swimmer, we configure the number of agents to be 10, each controlling a consecutive segment of length 2. We set maximum observation distances to $k = 3$ for 2-Agent HalfCheetah, $k = 2$ for 3-Agent Hopper, $k = 1$ for 2-Agent Walker, and $k = 0$ for 2-Agent Swimmer, 2-Agent Humanoid, 2-Agent HumanoidStandup, 4-Agent Ant, and ManyAgent Swimer [10x2]. Default team reward is used (see Table 1 in the Appendix).

Evaluation Procedure. We evaluate each method's performance using the following procedure: for each run of a method, we pause training every fixed number of timesteps (2000 timesteps for Continuous Predator-Prey and 4000 timesteps for MAMuJoCo) and run 10 independent episodes with each agent performing greedy decentralised action selection. The mean value of these 10 episode returns are then used to evaluate the performance of the learned policies. See Appendix F for further experimental details.

6 Results & Discussion

We now present benchmark results for the algorithms presented in Section 4.

6.1 Results

To better calibrate the performance of these algorithms on existing environments, we first benchmark them on Continuous Predator-Prey (see Figure 3(a)).

Surprisingly, COMIX, FacMADDPG, COVDN, IQL-CEM, and IDDPG all significantly outperform MADDPG on this task, both in terms of absolute performance and learning speed. While COMIX, FacMADDPG, and COVDN perform similarly, IQL-CEM and IDDPG do not reach the same limit performance.

We then benchmark this range of MARL algorithms on a diverse variety of MAMuJoCo tasks. COMIX significantly outperforms MADDPG on 2-Agent Walker, 3-Agent Hopper, 2-Agent Swimmer, 2-Agent Humanoid, 2-Agent HumanoidStandup, and ManyAgent Swimmer (Figure 3(b)-3(f) and Figure 3(i)), both in terms of absolute performance and learning speed. On 2-Agent HalfCheetah (Figure 3(g)) and 4-Agent Ant (Figure 3(h)), COMIX performs similarly to MADDPG. COVDN, IQL-CEM, and IDDPG all perform drastically worse than COMIX across all eight MAMuJoCo tasks[7], demonstrating the necessity of the non-linear mixing of agent utilities and conditioning on the state information in order to achieve competitive performance in such tasks. Finally, COMIX performs significantly better and is noticeably more stable than COMIX-NAF and COVDN-NAF (see Appendix G), demonstrating the superior performance of CEM over NAF in these tasks.

6.2 Discussion

Despite the ability to represent a richer form of coordination with its functionally unconstrained critic (Son et al. 2019), in our experiments MADDPG is not able to outperform COMIX, which uses a monotonically constrained mixing network. We hypothesise that this is because MADDPG's critic directly conditions on the full state (or joint observations) and actions of all agents. COMIX, by contrast, represents the optimal joint action-value function using a monotonic mixing function of per-agent utilities. Early in training, MADDPG's critic estimator may thus be more prone to picking up non-trivial suboptimal coordination patterns than COMIX. Such local minima might be hard to subsequently escape. By contrast, the monotonicity constraint on COMIX's mixing network may smooth the optimisation landscape, allowing COMIX to avoid suboptimal local minima more efficiently than MADDPG. In other words, COMIX's network architecture imposes a suitable prior that captures the forms of additive-monotonic coordination required to solve these tasks.

We find that FacMADDPG performs similarly to COMIX on both Continuous Predator-Prey (see Figure 3(a)) and all eight MAMuJoCo tasks (see Figure 3(b)-3(i)).

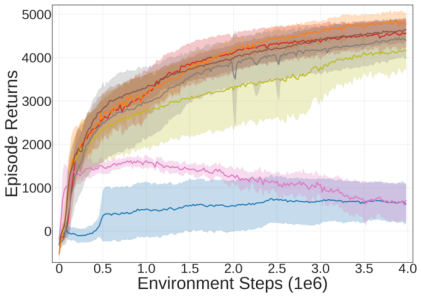
As COMIX and FacMADDPG use the same value factorisation as in QMIX and are both off-policy, this suggests that, in these continuous cooperative MAMuJoCo tasks, factorisation of the joint Q-value function plays a greater role in performance than the underlying algorithmic choices.

To further investigate the role of value function factorisation in such settings, we explore factoring the centralised critic in MADDPG in different ways: 1) *FacMADDPG-VDN:* we factor the centralised critic into a sum of individual action-value functions Q_a as in VDN, 2) *FacMADDPG-VDN-S:* we factor the centralised critic into a sum of Q_a and a state-dependent bias, and 3) *FacMADDPG-nonmonotonic:* we factor the centralised critic into a non-linear function of per-agent Q-values and a state-dependent bias. The joint Q_{tot} is not constrained to be monotonic in per-agent Q_a.

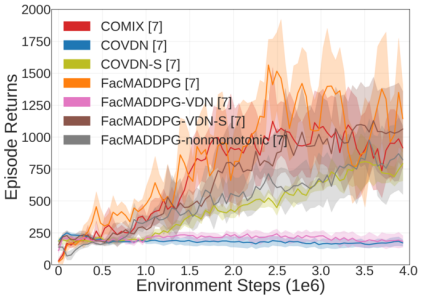
[5] https://github.com/schroederdewitt/multiagent-particle-envs/

[6] In 4-Agent Ant, each agent can observe the positions and velocities of its own body parts.

[7] The only exception is that, in 2-Agent HumanoidStandup (Figure 3(f)), COVDN performs slightly worse than COMIX.



(a) 2-Agent HalfCheetah [2x3]



(b) 2-Agent Walker [2x3]

Figure 4: Mean episode return on (a) 2-Agent HalfCheetah and (b) 2-Agent Walker. The mean across 7 seeds is plotted and the 95% confidence interval is shown shaded.

Figure 4 shows that COVDN and FacMADDPG-VDN yield similarly poor performance on both 2-Agent HalfCheetah and 2-Agent Walker. More interestingly, while FacMADDPG performs similarly to COMIX in both tasks, FacMADDPG-VDN-S performs significantly better than COVDN-S. The similar performance between FacMADDPG and FacMADDPG-VDN-S indicates that, in these tasks, factoring the centralised critic using QMIX and VDN-S results in equally good joint action-value functions. However, QMIX can learn more accurate per-agent utilities to better extract decentralised policies compared to VDN-S. Furthermore, FacMADDPG performs significantly better than FacMADDPG-nonmotonic in 2-Agent Walker and is noticeably more stable in 2-Agent HalfCheetah. This suggests that, while value factorisations are not constrained by decentralisability requirements in an actor-critic method, the excess coordinative expressivity in an unconstrained critic could lead to an increase in learning difficulty .

We believe exploring more forms of less restrictive (and potentially better) nonmonotonic factorisations for the centralised critic is an interesting avenue for future work (for a first attempt, see also Appendix H.1).

7 Related Work

While several MARL benchmarks with continuous action spaces have been released, few are simultaneously diverse, fully cooperative, decentralisable, and admit partial observability. The Multi-Agent Particle suite (Lowe et al. 2017) features a few decentralisable tasks in a fully observable planar point mass toy environment. Presumably due to its focus on real-world robotic control, RoboCup Soccer Simulation (Kitano et al. 1997; Stone and Sutton 2001; Riedmiller et al. 2009) does not currently feature an easily configurable software interface for MARL, nor suitable AI-controlled benchmark opponents. Liu et al. (2019) introduce MuJoCo Soccer Environment, a multi-agent soccer environment with continuous simulated physics that cannot be used in a purely cooperative setting and does not admit partial observability.

The most similar existing environments, though not as diverse as MAMuJoCo, are the decomposed MuJoCo environments Centipede and Snakes (Wang et al. 2018). The latter is similar to MAMuJoCo's 2-Agent Swimmer. Ackermann et al. (2019) evaluate on one environment similar to a configuration of 2-Agent Ant, but, similarly to Gupta, Egorov, and Kochenderfer (2017), do not consider tasks across different numbers of agents and MuJoCo scenarios.

A number of multi-agent variants of deep deterministic policy gradients (Lillicrap et al. 2015; Lowe et al. 2017) have been proposed for MARL in continuous action spaces: MADDPG-M (Kilinc and Montana 2018) uses communication channels in order to overcome observation noise in partially observable settings. By contrast, we consider fully decentralised settings without communication. R-MADDPG (Wang, Everett, and How 2019) equips MADDPG with recurrent policies and critics in a partially observable setting with communication. As we are primarily interested in the relative performance between policy gradients and value-based approaches, we employ feed-forward networks to avoid the complexities of recurrent network training.

NerveNet (Wang et al. 2018) achieves policy transfer across robotic agents with different numbers of repeated units. Unlike COMIX, NerveNet is not fully decentralisable as it requires explicit communication channels. Iqbal and Sha (2019) introduce MAAC, a variant of MADDPG for stochastic games, in which the centralised critics employ an attention mechanism on top of agent-specific observation embeddings. Unlike FacMADDPG, MAAC explicitly addresses settings where agents receive both individual and team rewards. Besides VDN and QMIX, QTRAN (Son et al. 2019) allows for arbitrary utility function mixings by introducing auxiliary losses that align utility function maxima with maxima of the joint Q-function. Despite being more expressive, QTRAN does not scale well to complex environments, such as SMAC (Bohmer, Kurin, and Whiteson 2020; Rashid et al. 2020) and may not generalise well to continuous actions due to the pointwise nature of its auxiliary losses.

Continuous Q-learning has so far been studied almost exclusively in the fully observable single-agent setting. Two distinct approaches to making greedy action selection tractable have emerged: Both normalized advantage functions (NAF Gu et al. 2016) and partially input-convex neural networks (PICNN Amos, Xu, and Kolter 2017) constrain the functional form of the action-value function approximator so as to guarantee an easily identifiable global maximum. As neither method strictly dominates the other in single-agent MuJoCo (Amos, Xu, and Kolter 2017), we choose NAF for its simplicity. On the other hand, heuristic search approaches, such as CEM forfeit global guarantees but al-

low for unconstrained Q-function approximators. CEM has been used successfully in single-agent robotic simulations (Kalashnikov et al. 2018). As for COMIX, we find that ablations using NAF perform poorly (see Appendix G).

8 Conclusion

In order to stimulate research in continuous MARL, in this paper, we introduced a novel benchmark suite MAMuJoCo. MAMuJoCo consists of a diverse set of multi-agent tasks with continuous action spaces and is easily extensible. We studied the utility of our benchmark by evaluating both existing algorithms, and novel continuous variants of existing algorithms. We found that our methods outperform state-of-the-art MADDPG on a number of MAMuJoCo tasks. In addition, we showed that, in these continuous cooperative MAMuJoCo tasks, value factorisation plays a greater role in performance than the underlying algorithmic choices. This motivates the necessity for extending the study of value factorisations from Q-learning to actor-critic algorithms.

References

Ackermann, J.; Gabler, V.; Osa, T.; and Sugiyama, M. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465* .

Amos, B.; Xu, L.; and Kolter, J. Z. 2017. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 146–155. JMLR. org.

Andrychowicz, O. M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. 2020. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research* 39(1): 3–20.

Arkin, R. C. 1989. Motor schema—based mobile robot navigation. *The International journal of robotics research* 8(4): 92–112.

Augugliaro, F.; Lupashin, S.; Hamer, M.; Male, C.; Hehn, M.; Mueller, M. W.; Willmann, J. S.; Gramazio, F.; Kohler, M.; and D'Andrea, R. 2014. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine* 34(4): 46–64.

Augugliaro, F.; Mirjan, A.; Gramazio, F.; Kohler, M.; and D'Andrea, R. 2013. Building tensile structures with flying machines. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3487–3492. IEEE.

Bescuca, M. 2019. Factorised critics in deep multi-agent reinforcement learning. In *Master Thesis, University of Oxford*.

Bohmer, W.; Kurin, V.; and Whiteson, S. 2020. Deep coordination graphs. In *International Conference on Machine Learning*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* .

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation* 2(1): 14–23.

Caccavale, F.; Uchiyama, M.; Siciliano, B.; and Khatib, O. 2008. Springer Handbook of Robotics. In *Cooperative Manipulators*, 701–718. Springer.

Ciosek, K.; and Whiteson, S. 2018. Expected policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

De Boer, P.-T.; Kroese, D. P.; Mannor, S.; and Rubinstein, R. Y. 2005. A tutorial on the cross-entropy method. *Annals of operations research* 134(1): 19–67.

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*.

Gu, S.; Lillicrap, T.; Sutskever, I.; and Levine, S. 2016. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2829–2838.

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* .

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 1856–1865.

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

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. 2018. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293* .

Kilinc, O.; and Montana, G. 2018. Multi-agent deep reinforcement learning with extremely noisy observations. *arXiv preprint arXiv:1812.00922* .

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E.; and Matsubara, H. 1997. RoboCup: A challenge problem for AI. *AI magazine* 18(1): 73–73.

Kojcev, R.; Etxezarreta, N.; Hernández, A.; and Mayoral, V. 2018. Hierarchical learning for modular robots. *arXiv preprint arXiv:1802.04132* .

Koller, D.; and Parr, R. 1999. Computing factored value functions for policies in structured MDPs. In *Proceedings of IJCAI*, 1332–1339.

Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190: 82–94.

Kuhn, H. 1953. Extensive games and the problem of information. *Annals of Mathematics Studies* 28.

Kurokawa, H.; Tomita, K.; Kamimura, A.; Kokaji, S.; Hasuo, T.; and Murata, S. 2008. Distributed self-reconfiguration of M-TRAN III modular robotic system. *The International Journal of Robotics Research* 27(3-4): 373–386.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037* .

Li, S.; Wu, Y.; Cui, X.; Dong, H.; Fang, F.; and Russell, S. 2019. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4213–4220.

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* .

Lin, L.-J. 1992a. Reinforcement learning for robots using neural networks. In *Dissertation, Carnegie Mellon University*.

Lin, L.-J. 1992b. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8(3-4): 293–321.

Liu, S.; Lever, G.; Merel, J.; Tunyasuvunakool, S.; Heess, N.; and Graepel, T. 2019. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151* .

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.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Nakagaki, K.; Dementyev, A.; Follmer, S.; Paradiso, J. A.; and Ishii, H. 2016. Chainform: A linear integrated modular hardware system for shape changing interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 87–96.

Oliehoek, F. A.; Amato, C.; et al. 2016. *A concise introduction to decentralized POMDPs*, volume 1. Springer.

Oliehoek, F. A.; Spaan, M. T. J.; and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *JAIR* 32: 289–353.

OpenAI. 2020. openai/baselines. Original-date: 2017-05-24T01:58:13Z.

Plappert, M.; Houthooft, R.; Dhariwal, P.; Sidor, S.; Chen, R. Y.; Chen, X.; Asfour, T.; Abbeel, P.; and Andrychowicz, M. 2018. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905* .

Rashid, T.; Farquhar, G.; Peng, B.; and Whiteson, S. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation. *arXiv preprint arXiv:2006.10800* .

Rashid, T.; Samvelyan, M.; Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 4292–4301.

Riedmiller, M.; Gabel, T.; Hafner, R.; and Lange, S. 2009. Reinforcement learning for robot soccer. *Autonomous Robots* 27(1): 55–73.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft multi-agent challenge. *CoRR* abs/1902.04043.

Schroeder de Witt, C.; Foerster, J.; Farquhar, G.; Torr, P.; Boehmer, W.; and Whiteson, S. 2019. Multi-agent common knowledge reinforcement learning. In *Advances in Neural Information Processing Systems 32*, 9924–9935. Curran Associates, Inc.

Shamshiri, R.; Weltzien, C.; Hameed, I.; Yule, I.; Grift, T.; Balasundram, S.; Pitonakova, L.; Ahmad, D.; and Chowdhary, G. 2018. Research and development in agricultural robotics: A perspective of digital farming. *International Journal of Agricultural and Biological Engineering* 11: 1–14.

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. In *International Conference on Machine Learning*, 5887–5896.

Stone, P.; and Sutton, R. S. 2001. Scaling reinforcement learning toward RoboCup soccer. In *Icml*, volume 1, 537–544. Citeseer.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, 2085–2087.

Takadama, K.; Matsumoto, S.; Nakasuka, S.; and Shimohara, K. 2003. A reinforcement learning approach to fail-safe design for multiple space robots—cooperation mechanism without communication and negotiation schemes. *Advanced Robotics* 17(1): 21–39.

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

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Varshavskaya, P.; Kaelbling, L. P.; and Rus, D. 2008. Automated design of adaptive controllers for modular robots using reinforcement learning. *The International Journal of Robotics Research* 27(3-4): 505–526.

Wang, R. E.; Everett, M. D.; and How, J. P. 2019. R-MADDPG for partially observable environments and limited communication. In *Proceedings of the Reinforcement Learning for Real Life workshop (at ICML)*.

Wang, T.; Liao, R.; Ba, J.; and Fidler, S. 2018. NerveNet: learning structured policy with graph neural networks. In *6th International Conference on Learning Representations, ICLR*.

Wright, C.; Buchan, A.; Brown, B.; Geist, J.; Schwerin, M.; Rollinson, D.; Tesch, M.; and Choset, H. 2012. Design and architecture of the unified modular snake robot. In *2012 IEEE International Conference on Robotics and Automation*, 4347–4354. IEEE.

Yim, M.; Zhang, Y.; and Duff, D. 2002. Modular robots. *IEEE Spectrum* 39(2): 30–34.

A Background

A.1 Deep Q-learning

Deep Q-Network (DQN Mnih et al. 2015) uses a deep neural network to estimate the action-value function, $Q(s, \boldsymbol{\tau}, \boldsymbol{u}; \theta) \approx \max_\pi Q^\pi(s, \boldsymbol{\tau}, \boldsymbol{u})$, where θ are the parameters of the network. For the sake of simplicity, we assume here feed-forward networks, which condition on the last observations $\boldsymbol{z}_t^a$, rather than the entire agent histories $\boldsymbol{\tau}_t$. The network parameters θ are trained by gradient descent on the mean squared regression loss:

$$\mathcal{L}^{\text{DQN}}_{[\theta]} := \mathbb{E}_{\mathcal{D}}\Big[\big(y_t^{\text{DQN}} - Q(s_t, \boldsymbol{z}_t, \boldsymbol{u}_t; \theta)\big)^2\Big], \tag{4}$$

where $y_t^{\text{DQN}} := r_t + \gamma \max_{\boldsymbol{u}'} Q(s_{t+1}, \boldsymbol{z}_{t+1}, \boldsymbol{u}'; \theta^-)$ and the expectation is estimated with transitions $(s_t, \boldsymbol{z}_t, \boldsymbol{u}_t, r_t, s_{t+1}, \boldsymbol{z}_{t+1}) \sim \mathcal{D}$ sampled from an *experience replay buffer* $\mathcal{D}$ (Lin 1992b). The use of replay buffer reduces correlations in the observation sequence. To further stabilise learning, θ^- denotes parameters of a *target network* that are only periodically copied from the most recent θ.

B MADDPG

Multi-agent deep deterministic policy gradient (MADDPG Lowe et al. 2017) is an actor-critic method that works in both cooperative and competitive MARL tasks with discrete or continuous action spaces. MADDPG was originally designed for the general case of *partially observable stochastic games* (Kuhn 1953), in which it learns a separate actor and centralised critic for each agent such that agents can learn arbitrary reward functions (including conflicting rewards in competitive settings). Here we implement a version specific to Dec-POMDPs and consider continuous actions. We assume each agent a has a deterministic policy μ^a, parameterised by θ^a, with $\boldsymbol{\mu}(\boldsymbol{\tau}; \boldsymbol{\theta}) := \{\mu^a(\tau^a; \theta^a)\}_{a=1}^N$. For Dec-POMDPs, MADDPG learns a centralised critic $Q_a^{\boldsymbol{\mu}}(s, \boldsymbol{u}; \phi)$ for each agent a with shared weights ϕ

that conditions on the full state s and the joint actions $\boldsymbol{u}$ of all agents. The policy gradient for θ^a is:

$$\nabla_{\theta^a}\mathcal{L}^{\mu}_{[\theta^a]} := -\mathbb{E}_{\mathcal{D}}\Big[\nabla_{\theta^a}\mu^a(\tau_t^a;\theta^a)\nabla_{u^a}Q_a^{\boldsymbol{\mu}}(s_t, \hat{\boldsymbol{u}}_t^a; \phi)\big|_{u^a=\mu^a(\tau_t^a)}\Big],$$

where $\hat{\boldsymbol{u}}_t^a := \{u_t^1, \ldots, u_t^{a-1}, u^a, u_t^{a+1}, \ldots, u_t^N\}$ and $s_t, \boldsymbol{u}_t, \boldsymbol{\tau}_t$ are sampled from a replay buffer $\mathcal{D}$. The shared centralised critic $Q_a^{\boldsymbol{\mu}}$ is trained by minimising the following loss:

$$\mathcal{L}^{\text{DPG}}_{[\phi]} := \mathbb{E}_{\mathcal{D}}\Big[\Big(y_t^a - Q_a^{\boldsymbol{\mu}}(s_t, \boldsymbol{u}_t; \phi)\Big)^2\Big], \tag{5}$$

where $y_t^a := r_t + \gamma\, Q_a^{\boldsymbol{\mu}}\big(s_{t+1}, \boldsymbol{\mu}(\boldsymbol{\tau}_{t+1}; \boldsymbol{\theta}'); \phi'\big)$ and transitions are sampled from a replay buffer $\mathcal{D}$ (Lin 1992a) and $\boldsymbol{\theta}'$ and ϕ' are target-network parameters.

C CEM

Cross-entropy method (CEM De Boer et al. 2005) is a sampling-based derivative-free heuristic search method that has been successfully used to find approximate maxima of nonconvex Q-networks in a number of single-agent robotic control tasks (Kalashnikov et al. 2018). Algorithm 1 outlines the full CEM process used in IQL-CEM, COVDN, and COMIX. Note that, for each agent, we sample actions from independent one-dimensional, rather than multivariate, Gaussians to find an action that approximately optimises its individual Q-function. To verify the utility of using CEM, we performed an ablation for COMIX using random search (over a space of $[-1, 1]$) and found it to be significantly worse than using CEM (with 2 iterations) in the continuous predator-prey task (see section F).

D COMIX

Q-learning has shown considerable success in multi-agent settings with discrete action spaces (Rashid et al. 2018). However, performing greedy action selection in Q-learning requires evaluating $\arg\max_{\mathbf{u}} Q_{tot}(s, \boldsymbol{\tau}, \boldsymbol{u})$, where Q_{tot} is the joint state-action value function. In discrete action spaces, this operation can be performed efficiently through enumeration (unless the action space is extremely large). In continuous action spaces, however, enumeration is impossible. Hence, existing continuous Q-learning approaches in single-agent settings either impose constraints on the form of Q-value to make maximisation easy (Gu et al. 2016; Amos, Xu, and Kolter 2017), at the expense of estimation bias, or perform only approximate greedy action selection (Kalashnikov et al. 2018). Neither approach scales easily to the large joint action spaces inherent to multi-agent settings, as 1) the joint action space grows exponentially in the number of agents, and 2) training state-action values required for greedy action selection becomes impractical when there are many agents.

This highlights the importance of learning a centralised but factored Q_{tot}. To factor large joint action spaces efficiently in a decentralisable fashion, COMIX models a joint state-action value function $Q_{MIX} = f\big(s, Q_1(\tau^1, u^1; \theta^1), \ldots, Q_N(\tau^N, u^N; \theta^N)\big)$, where Q_a are per-agent utility functions used for greedy action selection. Similarly to QMIX (Rashid et al. 2018), COMIX imposes

Algorithm 1 For each agent a, we perform n_c CEM iterations. Hyper-parameters $d_i \in \mathbb{N}$ control how many actions are sampled at the ith iteration.

function CEM $(Q_1, \ldots, Q_N, \tau_1, \ldots, \tau_N)$
  for $a := 1,\ a \leq N$ **do**
    $\boldsymbol{\mu}_a \leftarrow \mathbf{0} \in \mathbb{R}^{|\mathcal{A}_a|}$
    $\boldsymbol{\sigma}_a \leftarrow \mathbf{1} \in \mathbb{R}^{|\mathcal{A}_a|}$
    for $i := 1,\ i \leq n_c$ **do**
      for $j := 1,\ j \leq d_i$ **do**
        $\mathbf{v'}_{aj} \sim \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\sigma}_a)$
        $\mathbf{v}_{aj} \leftarrow \tanh(\mathbf{v'}_{aj})$
        $q_{aj} \leftarrow Q_a(\tau_a, \mathbf{v}_{aj})$
        $j \leftarrow j + 1$
      end for
      if $i < n_c$ **then**
        $U \leftarrow \{\mathbf{v'}_{al} \mid q_{al} \in \text{top}k_i(q_{a1}, \ldots, q_{ad_i}), \forall l \in \{1 \ldots N\}\}$
        $\boldsymbol{\mu}_a \leftarrow \text{sample\_mean}(U)$
        $\boldsymbol{\sigma}_a \leftarrow \text{sample\_std}(U)$
      else
        $m \leftarrow \arg\max_j q_{aj}$
        $\mathbf{u}_a \leftarrow \mathbf{v}_{am}$
      end if
      $i \leftarrow i + 1$
    end for
    $a \leftarrow a + 1$
  end for
  return $\langle \mathbf{u}_1, \ldots, \mathbf{u}_n \rangle$
end function

a monotonicity constraint on f to keep joint action selection compatible with action selection from individual utility functions.

COMIX is a simple variant of QMIX that scales to continuous action spaces. COMIX performs greedy selection of actions u^a with respect to utility functions $Q_a(\tau^a, u^a; \theta^a)$ for each agent a using the *cross-entropy method* (CEM De Boer et al. 2005), a sampling-based derivative-free heuristic search method that has been successfully used to find approximate maxima of nonconvex Q-networks in a number of single-agent robotic control tasks (Kalashnikov et al. 2018). The centralised but factored Q_{MIX} allows us to use CEM to sample actions for each agent independently and to use the individual utility function Q_a to guide the selection of maximal actions. Algorithm 2 outlines the full process for COMIX.

E Partially Observable Continuous Predator-Prey

We consider the mixed *simple tag* environment (Figure 5) introduced by Leibo et al. (2017), which is a variant of the classic predator-prey game. Three slower cooperating circular agents (red), each with continuous movement action spaces $u^a \in \mathbb{R}^2$, must catch a faster circular prey (green) on a randomly generated two-dimensional toroidal plane with two large landmarks blocking the way.

To obtain a purely cooperative environment, we replace the prey's policy by a hard-coded heuristic, that, at any time

Algorithm 2 Algorithmic description of COMIX. The function CEM is defined in Algorithm 1.

function COMIX
  Initialise ReplayBuffer, $\theta, \theta^-, \phi, \phi^-$
  for each training episode e **do**
    $s_0, \mathbf{z}_0 \leftarrow \text{EnvInit}()$
    for $t := 0$ until $t = T$ step 1 **do**
      $\mathbf{u}_t \leftarrow \text{CEM}(Q_1, \ldots, Q_N, \tau_t^1, \ldots, \tau_t^N)$
      $\langle s_{t+1}, \mathbf{z}_{t+1}, r_t \rangle \leftarrow \text{EnvStep}(\mathbf{u}_t)$
      ReplayBuffer $\leftarrow \langle s_t, \mathbf{u}_t, \mathbf{z}_t, r_t, s_{t+1}, \boldsymbol{z}_{t+1} \rangle$
    end for
    $\{\langle s_i, \boldsymbol{u}_i, \boldsymbol{z}_i, r_i, s'_i, \boldsymbol{z}'_i \rangle\}_{i=1}^b \sim$ ReplayBuffer
    $y_i \leftarrow r_i + \gamma \max_{\mathbf{u}'_i} Q_{\text{tot}}(s'_i, \boldsymbol{z}'_i, \boldsymbol{u}'_i; \boldsymbol{\theta}^-, \phi^-),\ \forall i$
    $\mathcal{L} \leftarrow \sum_{i=1}^{b} \Big(y_i - Q_{\text{tot}}(s_i, \boldsymbol{z}_i, \boldsymbol{u}_i; \boldsymbol{\theta}, \phi)\Big)^2$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}$
    $\phi \leftarrow \phi - \alpha \nabla_{\phi} \mathcal{L}$
  end for
end function

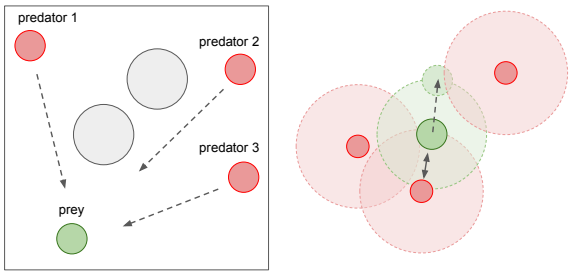


Figure 5: Continuous Predator-Prey. **Left:** Top-down view of toroidal plane, with predators (red), prey (green) and obstacles (grey). **Right:** Illustration of the prey's avoidance heuristic. Observation radii of both agents and prey are indicated.

step, moves the prey to the sampled position with the largest distance to the closest predator. If one of the cooperative agents collides with the prey, a team reward of $+10$ is emitted; otherwise, no reward is given. In the original simple tag environment, each agent can observe the relative positions of the other two agents, the relative position and velocity of the prey, and the relative positions of the landmarks. This means each agent's private observation provides an almost complete representation of the true state of the environment.

To introduce partial observability to the environment, we add an agent *view radius*, which restricts the agents from receiving information about other entities (including all landmarks, the other two agents, and the prey) that are out of range. Specifically, we set the view radius such that the agents can only observe other agents roughly 60% of the time. We open-source the full set of multi-agent particle environments with added partial observability (code available for download here: https://github.com/schroederdewitt/multiagent-particle-envs/).

F Experimental Details

In all experiments, we use a replay buffer of size 10^6, the target networks are updated via soft target updates with $\tau = 0.001$, and we scale the gradient norms during training to be at most 0.5. The mixing network used in COMIX, COMIX-NAF, and FacMADDPG consists of a single hidden layer of 64 units, utilising an ELU non-linearity. The hypernetworks producing the first layer weights and final layer weights and bias of the mixing network all consist of a single hidden layer of 64 units with a ReLU non-linearity. The output of the hypernetwork is passed through an absolute function and then resized to produce weights of appropriate size. For all value-based methods, to speed up the learning, we share the parameters of the agent networks across all agents. Similarly, in both MADDPG and FacMADDPG, a single actor and critic network is shared among all agents as we consider the Dec-POMDP settings here.

F.1 Continuous Predator-Prey

In all value-based methods, the architecture of all agent networks is a MLP with 2 hidden layers of 64 units and ReLU non-linearities, except for COVDN-NAF and COMIX-NAF where we replace ReLU units with tanh units as it leads to better performance. In IDDPG, MADDPG, and FacMADDPG, the architecture of the shared agent network and critic network is also a MLP with 2 hidden layers of 64 units and ReLU non-linearities, while the final output layer of the actor is a tanh layer, to bound the actions. In all value-based methods, the agent receives its local observation and individual action as input. In IDDPG, the actor receives the local observation as input and the critic receives the local observation and individual action as input. In MADDPG and FacMADDPG, the actor receives the local observation as input, while the critic receives the global state and the joint action of all agents as input. The global state consists of the joint observations of all agents.

During training and testing, we restrict each episode to have a length of 25 time steps. Training lasts for 2 million timesteps. To encourage exploration, we use uncorrelated, mean-zero Gaussian noise with $\sigma = 0.1$ during training (for all 2 million timesteps). We set $\gamma = 0.85$ for all experiments. The replay buffer contains the most recent 10^6 transitions. We train on a batch size of 1024 after every timestep. For the soft target network updates we use $\tau = 0.001$. All neural networks (actor and critic) are trained using Adam (Kingma and Ba 2014) optimiser with a learning rate of 0.01. To evaluate the learning performance, the training is paused after every 2000 timesteps during which 10 independent test episodes are run with agents performing action selection greedily in a decentralised fashion.

F.2 Multi-Agent MuJoCo

In all value-based methods, the architecture of all agent networks is a MLP with 2 hidden layers with 400 and 300 units respectively, similar to the setting used in OpenAI Spinning Up.[8] All neural networks use ReLU non-linearities for all hidden layers, except for COVDN-NAF and COMIX-NAF where we find tanh units lead to better performance. In both MADDPG and FacMADDPG, the architecture of the shared agent network and critic network is also a MLP with 2 hidden layers with 400 and 300 units respectively, while the final output layer of the actor network is a tanh layer, to bound the actions. In all value-based methods, the agent receives its local observation and individual action as input. In IDDPG, the actor receives the local observation as input and the critic receives the local observation and individual action as input. In MADDPG and FacMADDPG, the actor receives the local observation as input, and the critic receives the global state and the joint action of all agents as input. The global state consists of the full state information returned by the original OpenAI Gym (Brockman et al. 2016).

During training and testing, we restrict each episode to have a length of 1000 time steps. Training lasts for 4 million timesteps. To encourage exploration, we use uncorrelated, mean-zero Gaussian noise with $\sigma = 0.1$ during training (for all 4 million timesteps). We also use the same trick as in OpenAI Spinning Up to improve exploration at the start of training. For a fixed number of steps at the beginning (we set it to be 10000), the agent takes actions which are sampled from a uniform random distribution over valid actions. After that, it returns to normal Gaussian exploration. We set $\gamma = 0.99$ for all experiments. The replay buffer contains the most recent 10^6 transitions. We train on a batch size of 100 after every timestep. For the soft target network updates we use $\tau = 0.001$. All neural networks (actor and critic) are trained using Adam optimiser with a learning rate of 0.001. To evaluate the learning performance, the training is paused after every 4000 timesteps during which 10 independent test episodes are run with agents performing action selection greedily in a decentralised fashion.

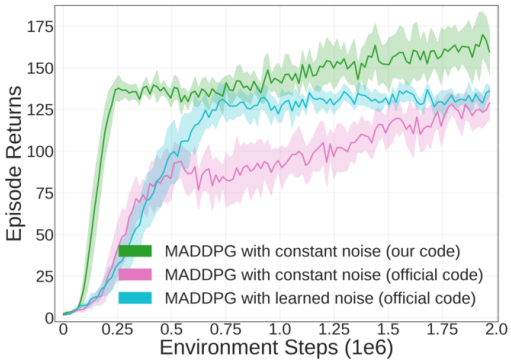
F.3 Exploration

The choice of exploration strategy plays a substantial role in the performance of deep deterministic policy gradient algorithms (Ciosek and Whiteson 2018). To keep exploration strategies comparable across MADDPG and Q-learning based COMIX, we restrict ourselves to noising in action spaces rather than parameter space (Plappert et al. 2018). MADDPG's official codebase [9] uses additive Gaussian noise with a standard deviation that is itself given by an additional policy output that is learnt end-to-end within the policy gradient loss. As Q-learning does not allow explicitly predict a policy output, we cannot apply a comparable strategy for COMIX. However, using MADDPG's official codebase, we find empirically that constant i.i.d. noising in the action spaces exhibits similar performance at lower variance than learnt noise on 2-Agent HalfCheetah (see Figure 6(b)). Even on Continuous Predator-Prey, a significantly less complex environment on which MADDPG's official codebase was tuned on, learnt exploration does not result in better limit performance than i.i.d. Gaussian noise (see Figure 6(a)).

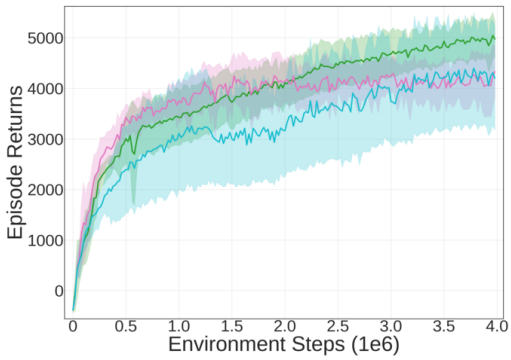
In addition, in our experiments we implement a version of MADDPG specific to Dec-POMDPs, in which the agent net-

[8] https://spinningup.openai.com/en/latest/.

[9] https://github.com/openai/maddpg.git



(a) Continuous Predator-Prey



(b) 2-Agent HalfCheetah [2x3]

Figure 6: Mean episode return on (a) Continuous Predator-Prey and (b) 2-Agent HalfCheetah comparing MADDPG with constant i.i.d. Gaussian noise and MADDPG with learned Gaussian noise. The mean across 7 seeds is plotted and the 95% confidence interval is shown shaded.

work and centralised critic is shared across all agents. Note that in MADDPG's official codebase, it learns a separate actor and centralised critic for each agent such that agents can learn arbitrary reward functions (including conflicting rewards in competitive settings). Figure 6 shows that, with constant i.i.d. Gaussian noise, MADDPG with shared agent and critic network (green) performs significantly better than MADDPG with separate actor and critic for each agent (pink) in Continuous Predator-Prey. In 2-Agent HalfCheetah, these two implementations yield similar performance.

G More Results

G.1 IQL-NAF, COVDN-NAF and COMIX-NAF

We benchmark *IQL-NAF*, *COVDN-NAF*, and *COMIX-NAF*, which add quadratic function constraints on each individual Q_a based on Normalized Advantage Functions (NAF Gu et al. 2016). Specifically, in COVDN-NAF, we represent Q_{tot} assuming additive mixing as in VDN, and add quadratic function constraints on Q_a based on NAF. In COMIX-NAF, we factor Q_{tot} assuming mixing monotonicity as in QMIX, and add quadratic function constraints on each Q_a based on NAF.

Figure 7 shows that, compared to COVDN-NAF and COMIX-NAF, COMIX is noticeably more stable on Continuous Predator-Prey (shown in Figure 7a). COVDN-NAF has sharp drops in performance at the late stage of training, while COMIX-NAF converges significantly more slowly than COMIX and is much more varied across seeds. On both 2-Agent HalfCheetah and 2-Agent Walker (shown in Figure 7b and 7c), COMIX significantly outperforms all other algorithms considered, both in terms of absolute performance and learning speed. This demonstrates that greedy action selection based on CEM heuristic search is more stable and performant than simple exact methods in practice.

H Critic Mixing Network Constraints in FacMADDPG

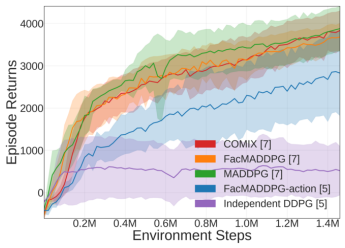
As in an actor-critic setting, the critic is not used for greedy action selection, FacMADDPG does not strictly require a monotonocity constraint on its critic mixing network. However, we find empirically that introducing the monotonicity requirement significantly increases performance in some tasks. As shown in Figure 8, FacMADDPG without the monotonicity constraint performs significantly worse than FacMADDPG in Continuous Predator-Prey and 2-Agent Walker. In 2-Agent HalfCheetah, FacMADDPG-nonmonotonic yields similar performance to FacMADDPG, but is much more varied across seeds. This supports the hypothesis that introducing monotonicity constraints strikes a reasonable trade-off between having independent critics with limited coordinative ability and the case where excess coordinative expressivity in the unconstrained critic leads to an increase in learning difficulty.

H.1 FacMADDPG-action

To take a first step toward identifying less restrictive critic factorisations in actor-critic settings, we empirically evaluate *FacMADDPG-action*, a variant of FacMADDPG whose mixing network is not monotonic in the agents' actions. FacMADDPG-action's joint critic is given by

$$Q^{\boldsymbol{\mu}}(s, \boldsymbol{u}; \boldsymbol{\theta}, \phi) = g_\phi\big(s, \{Q_a(\tau^a, u^a; \theta^a)\}_{a=1}^N\big), \quad (6)$$

where $\tilde{g}_\phi$ is no longer monotonic due to its direct dependence on the joint action $\mathbf{u}$. We find that FacMADDPG-action's empirical performance is not convincing (see below), adding further evidence that monotonic mixing may provide a particularly powerful inductive bias on cooperative learning tasks.



Results for 2-Agent HalfCheetah, including FacMADDPG-action.

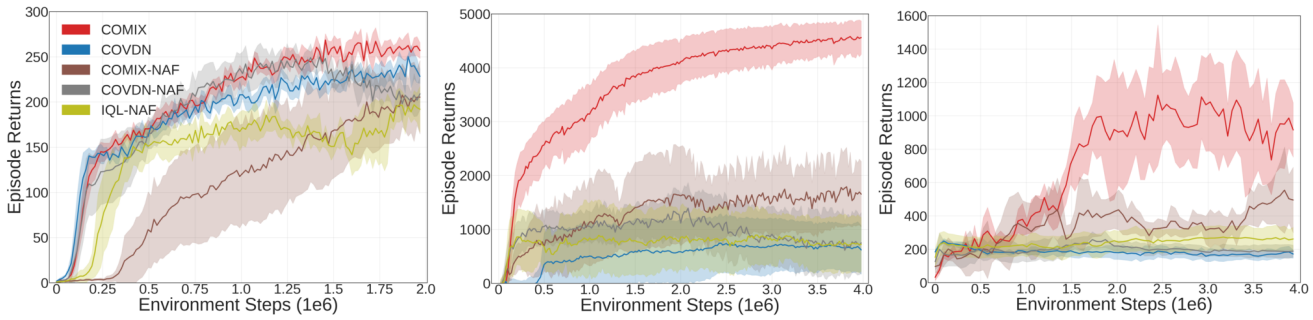


Figure 7: Mean episode return on (a) Continuous Predator-Prey, (b) 2-Agent HalfCheetah [2x3], and (c) 2-Agent Walker [2x3] comparing COMIX and ablations. The mean across 7 seeds is plotted and the 95% confidence interval is shown shaded.

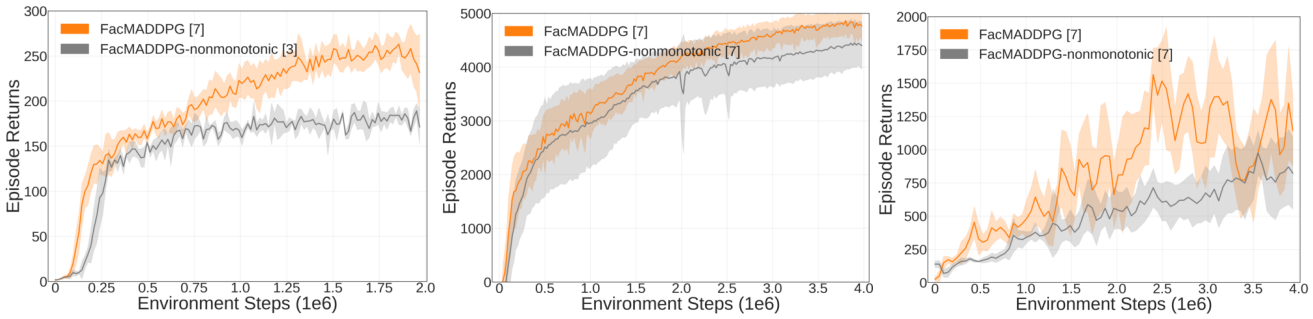


Figure 8: Mean episode return on (a) Continuous Predator-Prey, (b) 2-Agent HalfCheetah [2x3], and (c) 2-Agent Walker [2x3] comparing FacMADDPG and FacMADDPG without monotonicity constraints on the mixing network of the critic.

Task	Goal	Special observations	Reward function
2-Agent Swimmer	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 0.0001\alpha$
2-Agent Reacher	Fingertip (green) needs to reach target at random location (red).	Target is only visible to green agent.	$-\lVert\text{distance from fingertip to target}\rVert_2^2 + \alpha$
2-Agent Ant	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 5 \cdot 10^{-4} \lVert\text{external contact forces}\rVert_2^2 + 0.5\alpha + 1$
2-Agent Ant Diag	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 5 \cdot 10^{-4} \lVert\text{external contact forces}\rVert_2^2 + 0.5\alpha + 1$
2-Agent HalfCheetah	Maximise +ve x-speed.	-	$\frac{\Delta x}{\Delta t} + 0.1\alpha$
2-Agent Humanoid	Maximise +ve x-speed.	-	$0.25\frac{\Delta x}{\Delta t} + \min(10, 5 \cdot 10^{-6} \lVert\text{external contact forces}\rVert_2^2)$
2-Agent HumanoidStandup	Maximise +ve x-speed.	-	$\frac{y}{\Delta t} + \min(10, 5 \cdot 10^{-6} \lVert\text{external contact forces}\rVert_2^2)$
3-Agent Hopper	Maximise +ve x-speed.	-	$\frac{\Delta x}{\Delta t} + 0.001\alpha + 1.0$
4-Agent Ant	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 5 \cdot 10^{-4} \lVert\text{external contact forces}\rVert_2^2 + 0.5\alpha + 1$
6-Agent HalfCheetah	Maximise +ve x-speed.	-	$0.25\frac{\Delta x}{\Delta t} + \min(10, 5 \cdot 10^{-6} \lVert\text{external contact forces}\rVert_2^2)$
ManyAgent Swimmer	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 0.0001\alpha$
ManyAgent Ant	Maximise +ve x-speed.	All agents can observe velocities of the central torso.	$\frac{\Delta x}{\Delta t} + 5 \cdot 10^{-4} \lVert\text{external contact forces}\rVert_2^2 + 0.5\alpha + 1$

Table 1: Overview of tasks contained in MAMuJoCo. We define α as an action regularisation term $-\lVert\mathbf{u}\rVert_2^2$.