Decentralized Multi-Agent Actor-Critic with Generative Inference

Kevin Corder[1], **Manuel M. Vindiola**[2], **Keith Decker**[1]
[1]Dept. of Computer and Information Sciences, University of Delaware, Newark, DE 19716
[2]U.S. Army Research Laboratory, Aberdeen, MD 21005
kcorder@udel.edu, manuel.m.vindiola.civ@mail.mil, decker@udel.edu

Abstract

Developing coordinated behaviors within a multi-agent system is challenging for learning systems due to the non-stationarity introduced as agents co-adapt their policies over time. Recent multi-agent actor-critic methods address this challenge by combining centralized training with decentralized execution. These works constrain an agent's learning to the centralized phase such that agents may only execute pre-learned policies during the decentralized phase. This results in poor performance when disrupted agent communications introduce delayed, noisy, or missing information required for performing an action. In this work, we propose a new system that can gracefully handle partially-observable information due to communication disruptions during decentralized execution. Our approach augments the multi-agent actor-critic method's centralized training phase with generative modeling. This allows agents to infer other agents' observations when provided with the locally available context. Our method is evaluated on three tasks that require each agent to combine local observations with remote observations communicated by other agents. We evaluate our approach by introducing both partial observability and changing environment dynamics in a decentralized execution and training regime. We show that continued training in the decentralized phase performs as well or better than existing actor-critic methods.

Introduction

Reinforcement learning (RL) with function approximation has been used to solve difficult sequential decision making problems in high dimensional state and action spaces, such as game playing (Mnih, Silver, and Riedmiller 2013) and robotics (Haarnoja et al. 2018). Many decision making problems are best modeled as a multi-agent system in which agents learn concurrently with other agents. Two naive approaches that use single-agent RL methods in multi-agent problems are independent learning (IL) and joint action learning (JAL), but these approaches perform poorly. IL agents treat all others as part of the stochastic environment, ignoring relevant information about them. JAL agents condition on the full joint action and observation spaces for all agents, but these joint spaces grow exponentially with the number of agents and are therefore not scalable.

Most multi-agent reinforcement learning (MARL) methods rely on decentralized policies where each agent's own policy is dependent on local observations and actions. Decentralized scenarios usually have partial observations and limited communication. In some problems, the learning must also be decentralized and rely on agent communication (Zhang et al. 2018). In other cases, a simulator may be available so that agents may learn with extra state information while assuming free, instantaneous communication. After centralized learning, the agents execute decentralized policies using only local information. This training procedure is often feasible for software systems or physical domains with dedicated simulators, such as robotics or autonomous vehicles. Many recent works have adopted this Centralized Training, Decentralized Execution (CTDE) framework; we review several such methods in the following section.

Agent communication is the main approach to learning in decentralized systems when agents have partial observations. In CTDE methods, the communication during centralized training is implicit: all observations are shared freely and instantly. However, communication networks may be lossy, delayed, or lacking coverage.

We propose to address the limited communication learning gap of CTDE methods in decentralized execution with generative modeling. Specifically, we use a modified context conditional generative adversarial network (CC-GAN) (Denton, Gross, and Fergus 2016) to infer missing joint observations given partial observations. The task of filling in partial observations by generative inference is similar to the image inpainting problem for a missing patch of pixels: with an arbitrary number of missing observations, we would like to infer the most likely observation of the other agents.

We will extend the popular MADDPG method (Lowe et al. 2017) as it appears most amenable to full decentralization. MADDPG agents require both (1) the policies of other agents and (2) other agents' observations as input to the policies and critics. As other agents' approximate policies can be learned, the agents only need to learn a model for the other agents' observations. The generative model will learn this joint distribution of agent observations by training on random combinations of missing agent communications during centralized training. During the decentralized portion, the agents may sample from this model to continue learning under partial observability.

When the decentralized environment is dissimilar to the centralized environment, both agent policies and the generative model require fine-tuning to the new dynamics. In our system, this is worsened by the co-adaptation of the generative inference and inferred observations seen by agents. We provide experimental results to show how reward differs when the decentralized environment is partially observable vs. both partially observable with altered dynamics, as well as our method's performance with several options for decentralized updates.

Our contributions are as follows. We review the recent trend of CTDE MARL literature and identify that they are ill-suited to learn in the decentralized execution phase without explicit communication. We show how a context conditional generative model can address this problem for a popular CTDE method and provide a modified GAN that can learn the joint observation distribution. We experimentally evaluate our approach on three continuous multi-agent tasks. To the best of our knowledge, this is the first work to use generative models to overcome multi-agent partial observability or address decentralized learning in CTDE methods.

Related work

Centralized Training with Decentralized Execution

Many recent multi-agent actor-critic methods utilize centralized training with decentralized execution. This training procedure lessens the non-stationarity of co-adapting agents by providing additional information in centralized training. The majority of these methods: (1) solve only cooperative tasks by using a shared reward for all agents, (2) use a centralized critic function that conditions on all agents' observations and actions, and (3) use policy or critic networks that include recurrent components, such as an LSTM (Schmidhuber and Hochreiter 1997). Recurrent networks have been shown to be effective at learning policies in partially observable environments (Hausknecht and Stone 2015).

Gupta et al(Gupta, Egorov, and Kochenderfer 2017) solve cooperative, partially observable tasks with recurrent policies and curriculum learning. They compare several versions of the CTDE methods, including using Q-learning vs. actor-critic, centralized vs. decentralized policies with parameter sharing, and feed-forward vs. recurrent policies. (Foerster et al. 2018) use a centralized critic for all agents with decentralized recurrent policies for cooperative tasks. In addition, they use counterfactual baselines: difference rewards comparing agents' actions to a default action.

Instead of learning a single centralized critic for all agents, (Lowe et al. 2017) introduced multi-agent DDPG (MADDPG) which has a centralized critic for each agent and may be used in cooperative or competitive tasks. While recurrent networks may be used with MADDPG, only feed-forward networks were tested. (Rashid et al. 2018) also uses centralized critics for each agent, but includes a centralized mixing network to combine each agent's critic function. They also use recurrent polices and may be used in cooperative tasks. (Foerster et al. 2016) learns communication protocols over a limited-bandwidth communication channel. They propose two approaches that use recurrent policies in cooperative settings via parameter sharing or sending gradients over the communication channel.

We chose to extend MADDPG because it appears the most amenable to decentralization: each agent i has its own critic function Q_i (with no mixing network), policy π_i, approximate policies of other agents μ_i^j, and reward function to allow for both cooperative and competitive tasks. In addition, the policies are deterministic which allows for continuous state and action spaces.

Decentralized Learning

Traditional decentralized MARL approaches rely on persistent reliable communication so that agents may share local observations and jointly choose actions in an uncertain environment. When states are represented in a factored form, agents may solve a distributed constraint optimization problem over the network to choose a good joint action for all agents (Zhang and Lesser 2013). In pure MARL approaches, agents choose actions while sharing information over the communication channel. In some systems agents share local rewards but the state is fully observable (Zhang et al. 2018), and others use a communication-based consensus protocol to agree on a global state from local observations before choosing joint actions (Zhang, Ma, and Li 2018).

In contrast, our approach aims to allow learning despite disruptions in communication. When communication is unavailable, we infer the missing data given the available local observations of neighboring agents.

(Marinescu et al. 2014) propose an alternative communication-free method for decentralized learning that minimizes non-stationarity with a prediction and pattern change detection module. This module is updated when an approximate distributed constraint optimization solution determines that the environment transitions have sufficiently diverged from the predictions. However, this approach relies on solving hard optimization problems and a prediction model conditioned on past trajectories.

Our approach is also related to model-based learning (Sutton and Barto 2018) because the context conditional generative model essentially models the current state, but instead conditions only on the current joint partial observation.

Background

Reinforcement Learning

Formally, each task in decentralized MARL is represented by a discrete-time partially observable Markov Game (Littman 1994), a multi-agent extension of the Markov decision process (Sutton and Barto 2018). A Markov Game is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{R}, \mathcal{T}, n, \gamma \rangle$ where a set of n agents choose actions based on local observations to maximize their own expected cumulative reward. At each time step t, the environment has a true state $s \in \mathcal{S}$ and each agent i simultaneously chooses an action a_i from their individual set of available actions $\mathcal{A}_i \in \mathcal{A}$. The environment stochastically transitions to a new state s' given by the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, and each agent then receives a reward r_i according to its own reward function $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The discount factor γ is used for calculating expected return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ for time horizon T.

In the partially observable setting typical in decentralized MARL, each agent receives a private observation $o_i \in \mathcal{O}_i$ correlated with the state s. Agents choose actions using a stochastic policy $\pi_i : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, where π_i is a learned function with parameters θ_i.

The three main approaches to RL are action-value methods, policy gradient methods, and the actor-critic hybrid approach (Sutton and Barto 2018). Q-learning estimates the action-value function $Q^{\pi}(s, a)$: the future discounted reward when taking action a from state s while following policy π. Deep Q-Networks (DQN) used Q-learning with neural network function approximation to play Atari games from pixels to superhuman performance (Mnih, Silver, and Riedmiller 2013). DQN also introduced two stability improvements: target Q functions that are updated less frequently, and an experience replay buffer that stores environment transitions (s, a, r, s') for decorrelated batch updates.

Instead of learning a value function, policy gradient methods learn a parameterized policy directly (Sutton and Barto 2018). This approach is often more efficient but tends to have high variance. To reduce variance, actor-critic algorithms combine an action-value function Q along with the parameterized policy π to guide policy updates.

Policy gradient methods normally learn a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. Deterministic policy gradient (DPG) methods learn a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that returns a single action (Silver et al. 2014). Deep DPG (DDPG) is an off-policy model-free actor-critic algorithm that combines the DQN value function with a deterministic policy (Lillicrap et al. 2016). DDPG is a popular single-agent RL method that works with continuous state and action spaces. Like DQN, DDPG uses experience replay and target networks for both value and policy networks. Random noise is added to the policy's output for better exploration. From here on, all policies π are assumed deterministic.

Generative Adversarial Networks

Generative models learn a data distribution and can generate new samples similar to the learned distribution. The most popular class of models is the generative adversarial network (GAN) (Goodfellow, Pouget-Abadie, and Mirza 2014). A GAN is composed of two neural networks with opposing goals: a generator network $\mathcal{G}$ that receives noise as input and produces samples similar to the data distribution, and a discriminator network $\mathcal{D}$ that tries to determine real data points from those sampled from $\mathcal{G}$. While GANs have largely been applied to image generation, they should be able to learn any joint data distribution.

Wasserstein GANs (WGANs) are a variant of GANs that have been shown to have more reliable convergence and less mode collapse (Arjovsky, Chintala, and Bottou 2017). WGAN uses a critic rather than a discriminator (outputs are not probabilities), trains using a simple loss metric that approximates the Wasserstein distance when the network enforces a 1-Lipshitz constraint, and allows pre-training the critic to optimality. To avoid confusing the WGAN critic with the critic Q, we will continue to refer to it as the discriminator $\mathcal{D}$ as this distinction makes no difference in our work. For sample batch $\mathbf{x}$ and noise batch $z \sim \mathcal{N}(0, 1)$ with batch size b, the WGAN discriminator and generator have the following losses:

$$\nabla_{\theta_{\mathcal{D}}} \frac{1}{b} \sum_{i=1}^{b} \left[\mathcal{D}\left(\mathbf{x}^{(i)}\right) - \mathcal{D}\left(\mathcal{G}\left(z^{(i)}\right)\right) \right] \tag{1}$$

$$\nabla_{\theta_{\mathcal{G}}} \frac{1}{b} \sum_{i=1}^{b} \mathcal{D}\left(\mathcal{G}\left(z^{(i)}\right)\right) \tag{2}$$

Our work uses the context-conditional generative model, where the model takes a partial input and must generate a complete data sample. The closest computer vision task to our problem is image inpainting, where a patch of pixels from an image is removed and the model must fill the missing patch based on its learned model of the pixels' joint distribution. The CC-GAN objective function is given by

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{X}} [\log \mathcal{D}(\mathbf{x})] + \mathop{\mathbb{E}}_{\substack{\mathbf{x} \sim \mathcal{X}, \\ \mathbf{m} \sim \mathcal{M}}} [\log (1 - \mathcal{D}(\mathbf{x_I}))] \tag{3}$$

where $\mathbf{m}$ denotes a binary mask used to drop a patch from image $\mathbf{x}$, and $\mathbf{x_I} = (1 - \mathbf{m}) \odot \mathbf{x_G} + \mathbf{m} \odot \mathbf{x}$ is the combined inpainted image, where $\odot$ is element-wise multiplication and $\mathbf{x_G}$ is generated image by $\mathcal{G}$.

Other generative models for image inpainting include autoregressive models and context encoders, but they are not suitable for our approach. Autoregressive models, such as PixelRNN (van Oord, Kalchbrenner, and Kavukcuoglu 2016), require a pre-specified ordering over the pixels and thus will not work for arbitrary missing data. Context encoders use a variational autoencoder coupled with adversarial loss (Pathak et al. 2016), but results tend to be less accurate compared to CC-GANs.

Decentralized Fine-Tuning

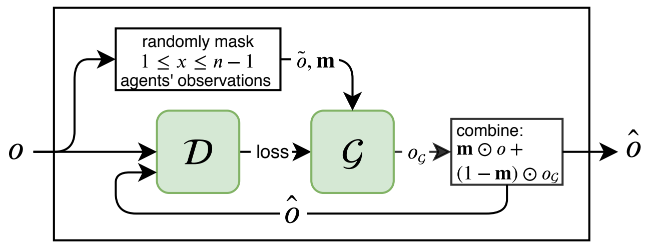


Figure 1: CC-WGAN update diagram. Describes an agent-based communication training procedure similar to random masking found in image inpainting tasks.

Inferring Observations with CC-WGAN

We approach the problem of inferring missing information from partial observations as a generative sampling problem, similar to the task of image inpainting. We use a modified

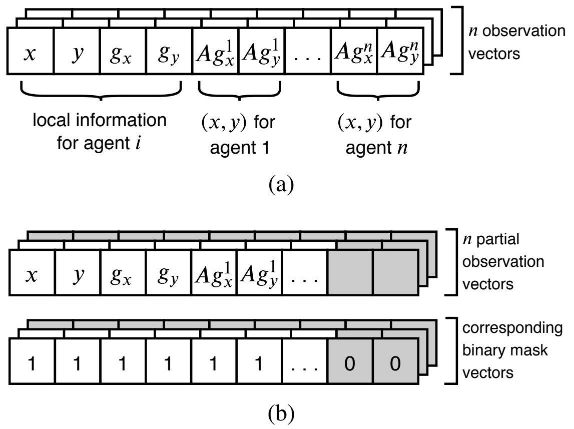


Figure 2: Random masking input and output used in training the CC-WGAN for a 2-D agent-distance partial observability criterion, where each agent's observation contains (x, y) positions for itself, other agents (Ag_x^i, Ag_y^i), and the goal (g_x, g_y). Fig. 2(a) depicts n agents' observation vectors fed to the masking function. Fig. 2(b) shows the masking function output: a set of n partial observations and n binary masks for agent i. Masked values in observations are filled with random normal values $z \sim \mathcal{N}(0, 1)$.

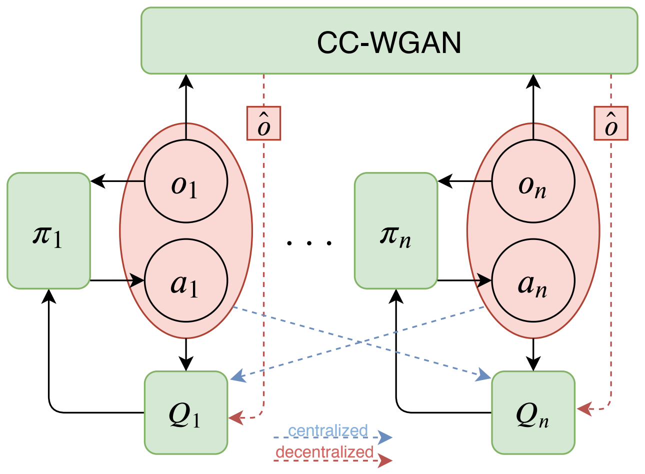


Figure 3: MADDPG with CC-WGAN learning diagram. During centralized training, the actor-critc learning updates are identical to MADDPG and CC-WGAN collects joint observations. During decentralized execution, agents sample from CC-WGAN to fill partial observations. Note that this diagram excludes the approximate policies μ; only the observations are shared during centralized training.

CC-GAN as our generative model (Denton, Gross, and Fergus 2016). Specifically, we train a WGAN with gradient penalty constraints (Arjovsky, Chintala, and Bottou 2017; Gulrajani et al. 2017) with the CC-GAN random data masking training procedure. We refer to our modified model as the context-conditional WGAN (CC-WGAN). In our experiments, the CC-WGAN was more reliable than regular CC-GAN for low-dimensional data.

Unlike the standard image generation task for GANs, we have no training data. We store joint observations $o = \langle o_1, \dots, o_n \rangle$ in a replay buffer $\mathcal{B}_\mathcal{G}$ just as MADDPG to stabilize learning when batch training.

Fig. 1 shows the update procedure for training the CC-WGAN. When the model updates, it randomly samples joint observations $o \sim \mathcal{B}_\mathcal{G}$ from the joint observation replay buffer $\mathcal{B}_\mathcal{G}$ to randomly mask $1 \leq x \leq n-1$ agent observations. During centralized training, all joint observations are available. If the CC-WGAN is updated in the decentralized phase, inferred observations are mixed into the updates. This is a form of semi-supervised learning because the model updates on its own predictions (Goodfellow, Bengio, and Courville 2016).

For each joint observations o, we randomly mask combinations of missing agents from o with a binary mask $\mathbf{m}$. Masked elements in o are replaced with random normal noise $z \sim \mathcal{N}(0, 1)$ to get the partial observation $\tilde{o}$. The masking procedure requires some knowledge of the conditions when observations will become partial, e.g., inter-agent distance greater than communications allow. Fig. 2 shows the masking procedure for distance-based partial observability based on (x, y) coordinates. In the diagram, $\mathcal{G}$ takes joint partial observation $\tilde{o}$ and binary mask vector $\mathbf{m}$, and produces the generated output $o_\mathcal{G} = \mathcal{G}(\tilde{o}, \mathbf{m})$. We then replace the masked portion in o with that portion of the generated output $o_\mathcal{G}$ to get a combined observation $\hat{o} = \mathbf{m} \odot o + (1 - \mathbf{m}) \odot o_\mathcal{G}$.

Where CC-GAN passes only the inpainted patch to the discriminator, this assumes fixed size image patches. Since any number of agents may be missing from the joint observation $\hat{o}$, we instead pass the combined observation to the discriminator. $\mathcal{D}$ is trained to distinguish batches of size m of real joint observations o and inferred observations $\hat{o}$ by minimizing the empirical Wasserstein distance:

$$\frac{1}{b} \sum_{i=1}^{b} \left[\mathcal{D}\left(o^{(i)}\right) - \mathcal{D}\left(\hat{o}^{(i)}\right) \right] \tag{4}$$

where $\hat{o} = \mathbf{m} \odot o + (1 - \mathbf{m}) \odot \mathcal{G}(\tilde{o}, \mathbf{m})$. Similarly, $\mathcal{G}$ is updated by maximizing:

$$\frac{1}{b} \sum_{i=1}^{b} \mathcal{D}\left(\hat{o}^{(i)}\right) \tag{5}$$

MADDPG with Inferred Observations

As stated before, we augment MADDPG (Lowe et al. 2017) with the CC-WGAN because it appears the most flexible CTDE method for decentralization. Each MADDPG agent i learns a deterministic policy π_i, a centralized critic Q_i, and a set of approximate policies μ_i^j for each other agent j.

Fig. 3 shows the MADDPG method updates along with the CC-WGAN for both centralized and decentralized phases. During centralized training, the critics Q_i and policies π_i are updated exactly as MADDPG. In addition, the CC-WGAN is collecting joint observations in its replay buffer and updating as described above.

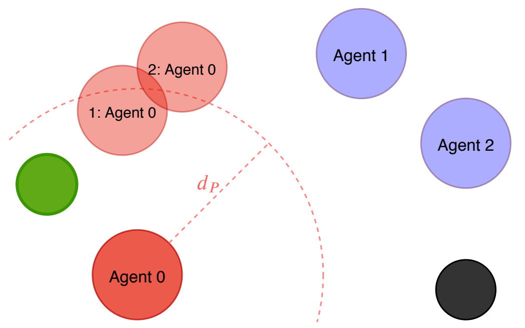


Figure 4: Physical deception scenario with agent inference at the beginning of an episode. Cooperating Agents 1 and 2 are near each other and can therefore observe each others' real positions. They are both too far away from adversary Agent 0 (with partially observable distance d_P) so they each sample from the generative model to infer Agent 0's position.

In the decentralized phase, local observations may be missing information about other agents. At each time step each agent i receives a partial observation $\tilde{o_i}$ which consists of the agent's local information and possibly information about other agents. When updating the centralized critics, if an agent i has information about agent j in its local partial observation $\tilde{o_i}$, then it can also see agent j's partial observation $\tilde{o_j}$. This is because we assume agents within range are "communicating" all local information. Just as in training, the joint partial observation $\tilde{o}$ is passed to the generator to get o_G and combined via binary mask $\mathbf{m}$ with $\tilde{o}$ to get the inferred observation $\hat{o}$.

Following the derivation in (Lowe et al. 2017), the deterministic policy loss with inferred observations is:

$$\nabla_{\theta_i} J\left(\pi_i\right) = \underset{\hat{o},a\sim\mathcal{B}}{\mathbb{E}} \left[\nabla_{\theta_i} \pi_i \left(a_i | \hat{o_i}\right) \nabla_{a_i} Q_i^{\pi}\left(\hat{o}, a\right)\right], \tag{6}$$

where $a = a_1, \ldots, a_n$ are taken from approximate policies such that $\mu_i^j(\hat{o_j}) = a_j$. The approximate policies are updated with:

$$\mathcal{L}(\theta_i^j) = -\mathbb{E}_{\hat{o_j}, a_j}\left[\log \pi_i^j\left(a_j | \hat{o_j}\right) + \lambda H(\pi_i^j)\right] \tag{7}$$

where $H(\pi_i^j)$ is the entropy of the policy distribution and λ is a small weight (0.001 in experiments). The centralized critics Q_i are updated with:

$$\mathcal{L}\left(\theta_i\right) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}\left[\left(Q_i^{\mu}\left(\mathbf{x}, a_1, \ldots, a_N\right) - y\right)^2\right], \tag{8}$$

$$y = r_i + \gamma Q_i^{\mu}(\hat{o}', \mu_i'^1\left(\hat{o_1}\right), \ldots, \mu_i'^N\left(\hat{o_N}\right)) \tag{9}$$

where μ_i' is a target policy and $\hat{o_i}'$ is an inferred next observation following observation o.

Experiments

Environments and Setup

We evaluate our method under three continuous scenarios, two competitive and one cooperative, of the Multi-agent Particles Environment (MPE)[1]. MPE was introduced in the original MADDPG paper (Lowe et al. 2017). In order to directly compare to their results, we use the physical deception and predatory-prey competitive scenarios also used by (Lowe et al. 2017). We additionally test on a cooperative navigation scenario where agents share a reward function to show both competitive and cooperative settings. The communication-based scenarios tested by Lowe et al. have no clear way to make partially observable and still be solvable so we did not include them. In contrast, the physical scenarios we evaluate on are easy to make partially observable by using a partially observable distance d_P: when agents' are farther from each other than d_P, they cannot observe each others' positions, velocity, etc. (see Fig. 4). We use $d_P = 1$ in all experiments, where the width of the 2D square environment is 2.

When using agent-distance partial observability, learning the coordination of predator-prey appears hardest, followed by physical deception, and lastly cooperative navigation. In predator-prey, three agents must coordinate to catch one faster agent, so there is no stable strategy with limited view. In physical deception, two agents should learn to deceive an adversary agent by covering two landmarks to hide which is the correct goal. If the adversary is out of range, this strategy should not change. In cooperative navigation, each agent must move to and remain near different landmarks. With limited range observations, an agent can still tell if another agent is covering the same landmark within distance d_P and move to a different one.

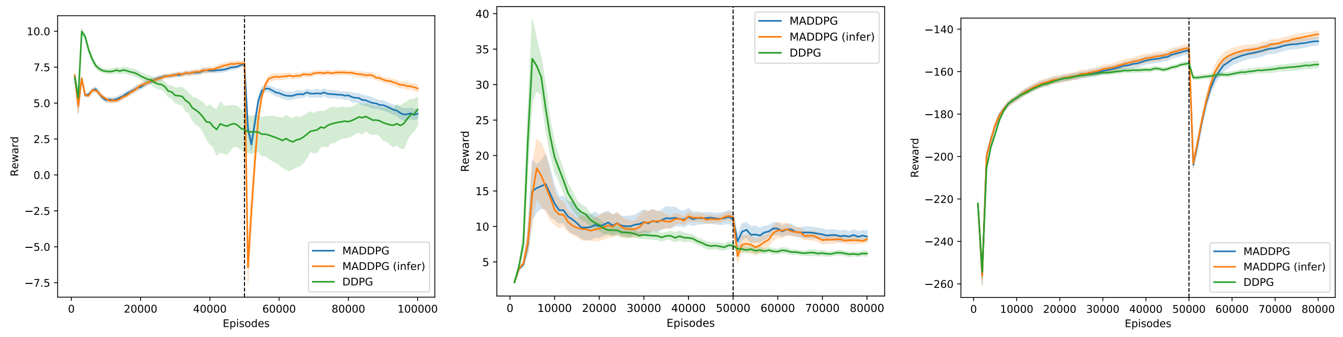
In addition to making the decentralized phase partially observable, we approximate real-world deployment by changing the transition function by modifying simulation dynamics. Dynamics are changed by adding scaled random normal noise and translation to both actions and observations. Combined with partial observability, the added noise makes learning more difficult for both the policies and the CC-WGAN and requires fine-tuning to the new distribution.

Each episode has 200 steps with no early termination. All models are updated every 100 steps, and are represented with a three-layer, feed-forward neural network with 64 hidden units. The models use the Adam optimizer and each non-output layer uses a ReLU activation function. Each plot shows the mean and standard deviation shading of agent rewards over 30 independent trials for each algorithm. Each algorithm within a single plot receives the same set of 30 random seeds for accurate comparisons with random exploration. In all plots, a vertical dashed line marks the episode in which the environment becomes decentralized.

Results

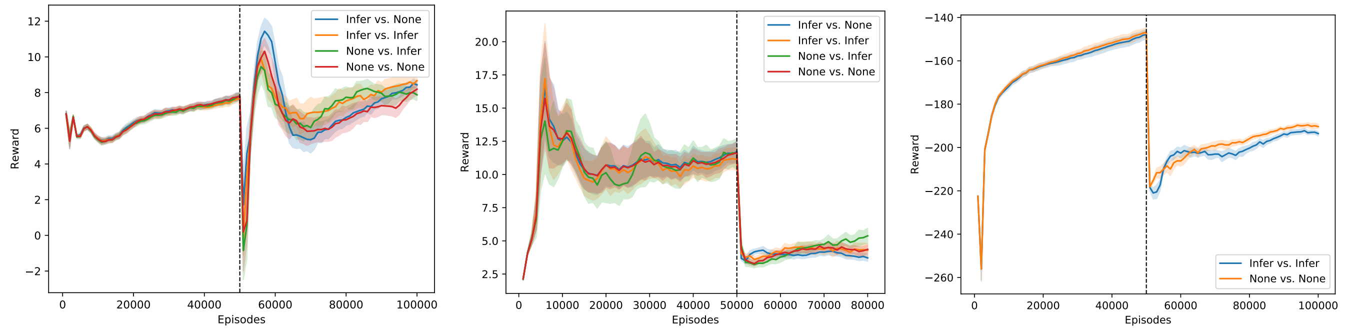
The following plots compare our approach of augmenting MADDPG with CC-WGAN inference against regular MADDPG and DDPG. We chose DDPG because it performed the best among all IL methods in (Lowe et al. 2017) and we use the same environment. Agents learn approximate policies for all other agents in MADDPG with and without generative inference. MADDPG and our version are identi-

[1]MPE code: http://github.com/openai/multiagent-particle-envs



(a) Physical Deception (b) Predator-Prey (c) Cooperative Navigation

Figure 5: Reward for cooperating agents with distance-based partial observations in decentralized phase.



(a) Physical Deception (b) Predator-Prey (c) Cooperative Navigation

Figure 6: Both sides of cooperating agent rewards with both distance-based partial observations and altering environment dynamics in decentralized phase. "Infer" is our approach and "None" is MADDPG.

cal during the centralized training because agents only use the CC-WGAN inference in the decentralized phase. After centralized training, we let MADDPG continue learning while treating the partial observability as random noise, whereas our approach infers the missing data. Except for Fig. 7, the policies and CC-WGAN continue updating in the decentralized phase.

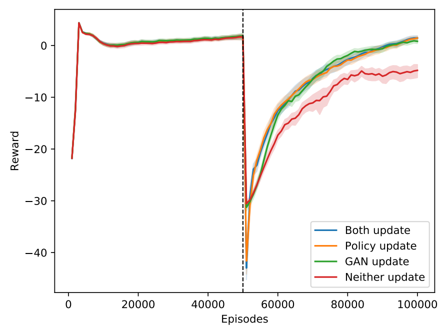
We did not test against the other CTDE methods discussed in Related Work because these methods use recurrent critics or policies. As such, they condition on past trajectories and would likely overcome the partial observability implicitly. Since MADDPG may also be used with recurrent units, we would like to compare recurrent MADDPG with and without generative inference against some of the recurrent CTDE methods in future work.

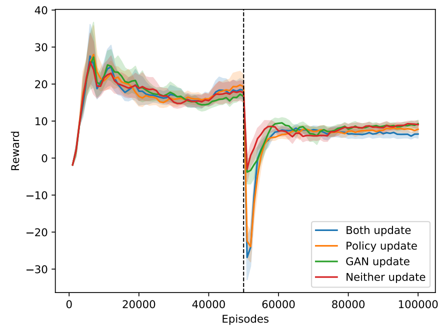
In Fig. 5, we show the cooperating agents' reward for all agents using MADDPG, MADDPG with CC-WGAN inference, and DDPG. In this plot we only introduce partial observability when agents are farther than the partially observable distance $d_P = 1$.

Fig. 6 shows the reward for cooperating agents with both partial observability and altered environment dynamics in the decentralized phase to evaluate the capability of fine-tuning to another environment. The overall reward is much lower here than Fig. 5 which suggests the CC-WGAN is not well-suited to switching its modeled observation distribution. Deploying into a real world scenario is an important application, which is addressed by sim-to-real transfer (Peng et al. 2018).

In Fig. 7, we compare the total reward for our method with four options of decentralized updates, where the decentralized phase has both partial observability and altered environment dynamics. All agents use the CC-WGAN inferred observations when choosing actions. The curves show the difference in whether the policy and CC-WGAN update on inferred observations in the decentralized phase.

Lastly, in Fig. 8, we show the CC-WGAN's reconstruction MSE during training over several partial observability distances $d_P \in [0.0, 2.0]$. When $d_P = 0$, the model is effectively using IL in the decentralized phase; when $d_P = 2$, the model is usually using complete observations. We show this reconstruction plot because the agents' observations are low-dimensional (i.e., not images as GANs are usually used for). MSE may not be a good metric for this error, however it was more informative than cosine similarity. We initially expected the error to be lower for larger d_P, but it is clear that the CC-WGAN reconstruction error has little to do with the partial observability distance. We would like to investigate the conditions under which the CC-WGAN gives better predictions.


(a) Physical Deception


(b) Predator-Prey

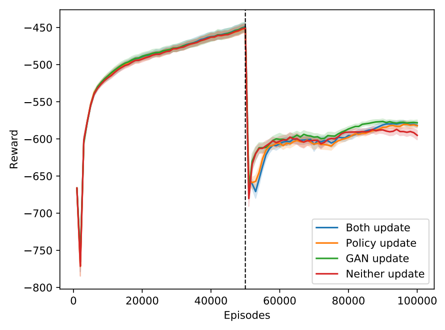

(c) Cooperative Navigation

Figure 7: Total reward for our approach with all four combinations of whether agent policies and CC-WGAN continue to update on inferred observations during decentralized phase.

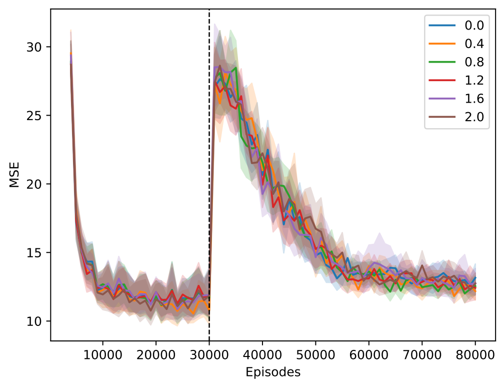


Figure 8: Mean squared error CC-WGAN reconstruction between latest joint observations o and inferred observations $\hat{o}$ for the predator-prey scenario with altered dynamics. Curves show varying partial observability distances d_P had little effect on error.

Discussion

The results shown here reveal properties about context-based modeling of observations in MARL and the scenarios under which our current version is appropriate.

As CC-WGAN learns a joint observation distribution by sampling joint observations from its replay buffer $\mathcal{B}_{\mathcal{G}}$, it has no temporal coherence: its inference is independent from the previous step given the current context. Without a model of observation trajectories, it is ill-suited for dynamic tasks with no clear stable behaviors under the partial observability.

This problem can be seen in Fig. 5 depending on the scenario's need for non-local information and the stability of optimal behavior. Inferring other agents' observations is useful when the task requires non-local coordination like the physical deception and predator-prey scenarios. Cooperative navigation agents can move to a different landmark if another agent is covering the same landmark, but may take slightly longer. Without temporal coherence, the CC-WGAN has trouble modeling non-stationary observation distributions like predator-prey. In contrast, physical deception and cooperative navigation have a stable optimal policy. In summation, our approach works best with a stable observation distribution (physical deception and cooperative navigation) and is useful in tasks requiring non-local coordination (physical deception and predator-prey). This is why our reward is significantly higher in physical deception, slightly higher in cooperative navigation, and slightly lower in predator-prey.

As seen in Fig. 7, when the CC-WGAN and policy update on inferred observations the decentralized reward dropoff is more drastic than without updating on the inference. This is due to the co-adaptation between the agents' policies and CC-WGAN inference: the CC-WGAN must learn based on new observations being generated by agents choosing actions based on the inferred observations from the CC-WGAN. Also it appears that having either policy updates or GAN updates on inferred observations gives roughly the same benefit.

Conclusions and Future Work

In this paper we reviewed the recent trend of MARL methods utilizing centralized training with decentralized execution (CTDE) and identified that none of the methods may continue learning in the decentralize phase without adding explicit communication. We proposed to learn a context conditional generative model during the centralized training phase that allows for a popular CTDE actor-critic method to continue learning in the decentralized phase, and showed that this addition allows for increased reward and coordination in three continuous multi-agent tasks.

Our approach is useful for completing partial observations in Markovian environments where decentralized environment dynamics closely match the centralized training dynamics. In non-Markovian environments where agents should condition on trajectories, recurrent policies or critics would help solve the problem. Our experiments show that context is useful in settings when there is a stable optimal behavior for agents, but training on trajectories may be able to learn more difficult observation distributions. Lastly, we would like to show the efficacy of our approach in larger, more difficult environments.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein GAN. *arXiv*.

Denton, E.; Gross, S.; and Fergus, R. 2016. Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks. *arXiv*.

Foerster, J. N.; Assael, Y. M.; De Freitas, N.; and Whiteson, S. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Neural Information Processing Systems (NIPS)*.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. In *Association for the Advancement of Artificial Intelligence*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT press.

Goodfellow, I.; Pouget-Abadie, J.; and Mirza, M. 2014. Generative Adversarial Networks. *arXiv preprint arXiv: ...* 1–9.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 5767–5777.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *AAMAS Workshop*, volume 10642 LNAI, 66–83.

Haarnoja, T.; Ha, S.; Zhou, A.; Tan, J.; Tucker, G.; and Levine, S. 2018. Learning to Walk via Deep Reinforcement Learning. In *International Conference on Machine Learning*.

Hausknecht, M., and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Neural Information Processing Systems (NIPS)*.

Marinescu, A.; Dusparic, I.; Taylor, A.; Cahill, V.; and Clarke, S. 2014. Decentralised Multi-Agent Reinforcement Learning for Dynamic and Uncertain Environments. In *Association for the Advancement of Artificial Intelligence*.

Mnih, V.; Silver, D.; and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. *arXiv* 1–9.

Pathak, D.; Krähenbühl, P.; Donahue, J.; Darrell, T.; and Efros, A. A. 2016. Context Encoders: Feature Learning by Inpainting. *arXiv*.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *International Conference on Robotics and Automation*.

Rashid, T.; Samvelyan, M.; Schroeder De Witt, C.; 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*.

Schmidhuber, J., and Hochreiter, S. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Silver, D.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press, 2nd edition.

van Oord, A.; Kalchbrenner, N.; and Kavukcuoglu, K. 2016. Pixel Recurrent Neural Networks. In *International Conference on Machine Learning*, 1747–1756.

Zhang, C., and Lesser, V. 2013. Coordinating Multi-Agent Reinforcement Learning with Limited Communication. In *Autonomous Agents and Multi-Agent Systems*.

Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; and Basar, T. 2018. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. In *International Conference on Machine Learning*.

Zhang, W.; Ma, L.; and Li, X. 2018. Multi-agent reinforcement learning based on local communication. *Cluster Computing*.