Multi-Adversary Physical Deception: Comparing MADDPG and DDPG under Scaled Adversaries

Steffi Roy
Khoury College of Computer Sciences
Northeastern University
roy.ste@northeastern.edu

Abstract

In adversarial multi-agent environments, cooperative agents must act strategically under incomplete information. This project studies deceptive behavior in the Physical Deception task, where cooperative agents learn to mislead an adversary about the location of a true target landmark. We implement and compare Multi-Agent Deep Deterministic Policy Gradient (MADDPG), which uses Centralized Training with Decentralized Execution (CTDE), against Independent DDPG across three agent configurations: 2 cooperative agents vs 1 adversary, 2 cooperative vs 2 adversaries, 3 cooperative vs 1 adversary and 3 cooperative vs 2 adversaries. Training runs over 10,000 episodes using a modified PettingZoo simple_adversary environment. We evaluate performance through reward curves, adversary reach rate, and coverage spread.

1 Introduction

In multi-agent reinforcement learning (MARL), agents learn through interaction in shared environments where other agents may have competing objectives and incomplete information. When agents must infer intent from observed behavior: movement patterns, positioning, timing rather than from direct signals, this naturally gives rise to strategic reasoning. Agents may benefit not only from acting effectively in the environment, but also from actively influencing how their opponents interpret the situation. This higher-order reasoning, where actions are chosen partly for their effect on an opponent's beliefs, is the basis of deceptive behavior in MARL.

The Physical Deception task, introduced by Lowe et al. [1] and formalized in OpenAI's Multi-Particle Environments, provides a controlled setting to study this phenomenon. A team of cooperative agents knows the location of a true target landmark among several decoys. The adversary can see all landmarks but does not know which one is the real target, and must infer it by observing the cooperative agents' movements. The cooperative agents are rewarded for reaching the target and penalized if the adversary also reaches it, creating an incentive to cover multiple landmarks simultaneously and deceive the adversary about the true target's identity.

The central algorithmic challenge in this setting is non-stationarity. In multi-agent settings, each agent's policy is part of the environment seen by every other agent. As all agents learn simultaneously, the effective environment experienced by any individual agent is constantly shifting. Lowe et al. [1] demonstrate that the probability of taking a gradient step in the correct direction decreases exponentially with the number of agents under naive policy gradient methods.

Lowe et al. [1] address non-stationarity through the CTDE framework. During training, each agent's critic is

given access to the observations and actions of all agents a centralized view that provides a stationary learning signal. At execution time, however, each agent acts only on its own local observations. The paper reports that MADDPG cooperative agents successfully deceive the adversary approximately 94% of the time with $N = 2$ landmarks, compared to much lower rates for DDPG agents.

This project implements MADDPG and compares it against Independent DDPG across three agent configurations in a modified version of the simple_adversary environment. The goal is to evaluate whether the centralized training signal of MADDPG produces measurably better deceptive coordination, and to examine how coordination dynamics change as the number of cooperative agents and adversaries scales.

2 Methods

2.1 Environment

We use a modified version of PettingZoo's `simple_adversary_v3` environment [3]. The original environment supports only one adversary, so we created a custom version (`custom_simple_adversary.py`) by editing the PettingZoo source to accept a `num_adversaries` parameter. The `make_world` function was extended to instantiate multiple adversary agents while preserving the original distance-based reward structure.

All agents can see the positions of all landmarks. The critical information asymmetry is that cooperative agents observe which landmark is the true target (an extra 2-dimensional goal vector prepended to their observation), while the adversary does not receive this information. The adversary must infer the target purely from the cooperative agents' movement patterns.

The reward structure is as follows. Cooperative agent reward has two components:

$$r_{\text{coop}} = \underbrace{-\min_{i \in \text{good}} d(a_i, \text{target})}_{\text{coverage term}} + \underbrace{\min_{j \in \text{adv}} d(a_j, \text{target})}_{\text{deception term}} \tag{1}$$

where $d(a, \text{target})$ denotes Euclidean distance from agent a to the true target landmark. The coverage term rewards getting any cooperative agent close to the target. The deception term is a positive bonus that grows as adversaries are kept further from the target — when deception is working, this term can outweigh the coverage penalty and push the total cooperative reward positive. The adversary reward is:

$$r_{\text{adv}} = -d(a_{\text{adv}}, \text{target}) \tag{2}$$

which is always non-positive, approaching zero only when the adversary reaches the target. Episodes run for `max_cycles=25` steps with continuous 5-dimensional action spaces. four configurations are evaluated: 2 good vs 1 adversary (baseline), 2 good vs 2 adversaries, and 3 good vs 2 adversaries.

2.2 Algorithms

MADDPG. Each agent maintains its own actor network $\mu_i(o_i)$ mapping local observations to continuous actions, and a centralized critic $Q_i(x, a_1, \ldots, a_N)$ that receives the concatenated observations and actions of all agents. The centralized critic is updated using:

$$\mathcal{L}(\phi_i) = \mathbb{E}\left[(Q_i(x, a_1, \ldots, a_N) - y)^2\right], \quad y = r_i + \gamma Q_i'(x', a_1', \ldots, a_N') \tag{3}$$

where Q_i' is a target network updated via soft update $\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i'$. The actor is updated by maximizing the critic's Q-value estimate:

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}\left[\nabla_{\theta_i}\mu_i(o_i) \cdot \nabla_{a_i} Q_i(x, a_1, \ldots, a_N)\big|_{a_i=\mu_i(o_i)}\right] \tag{4}$$

At execution time the critic is discarded and each actor uses only its own local observation.

2.3 Actor-Critic Network Architecture

Each agent in MADDPG maintains two separate neural networks: an actor and a critic. Their inputs, outputs, and roles are fundamentally different.

2.3.1 Actor Network

The actor network $\mu_{\theta_i} : \mathcal{O}_i \to \mathcal{A}_i$ maps an agent's own local observation to a continuous action. It is a two-layer MLP:

$$a_i = \mu_{\theta_i}(o_i) = \tanh\left(W_3 \cdot \text{ReLU}\left(W_2 \cdot \text{ReLU}(W_1 o_i + b_1) + b_2\right) + b_3\right) \tag{5}$$

where $W_1 \in \mathbb{R}^{64\times d_o}$, $W_2 \in \mathbb{R}^{64\times 64}$, $W_3 \in \mathbb{R}^{d_a\times 64}$, and d_o, d_a are the observation and action dimensions respectively, computed dynamically from the environment. The tanh output bounds all actions to $[-1, +1]$.

Key constraint: the actor only ever sees its own observation o_i — both during training and at execution time. It never receives information about other agents' observations, goals, or actions.

2.3.2 Critic Network

The critic network $Q_{\phi_i} : \mathcal{X} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$ maps the joint state and joint actions of all agents to a scalar Q-value estimate for agent i. It is also a two-layer MLP:

$$Q_i = Q_{\phi_i}(x, a_1, \ldots, a_N) = W_3 \cdot \text{ReLU}\left(W_2 \cdot \text{ReLU}(W_1[x; a_1; \ldots; a_N] + b_1) + b_2\right) + b_3 \tag{6}$$

where $x = [o_1; o_2; \ldots; o_N]$ is the concatenation of all agents' observations and $[x; a_1; \ldots; a_N]$ is the full joint input vector. The output layer uses no activation Q-values are unbounded real numbers.

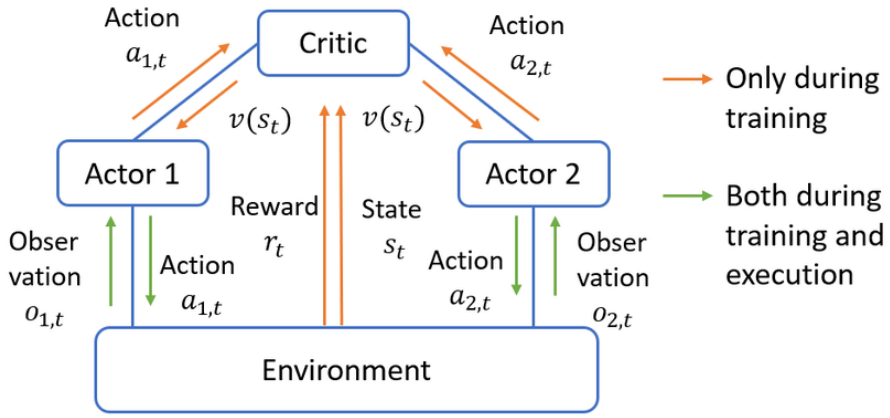


Figure 1: MADDPG actor-critic architecture under Centralized Training with Decentralized Execution (CTDE). Orange arrows indicate connections used only during training. Both actors send their actions $a_{1,t}$ and $a_{2,t}$ to the centralized critic, which returns the value estimate $v(s_t)$ as a gradient signal. Green arrows indicate connections used both during training and execution each actor receives its own local observation $o_{i,t}$ from the environment and outputs its action $a_{i,t}$. At execution time, the critic is discarded and each actor acts independently using only its own observation. Figure adapted from [4].

2.3.3 Target Networks

Both the actor and critic have corresponding target networks $\mu'_{\theta'_i}$ and $Q'_{\phi'_i}$ — delayed copies used to compute stable training targets. They are updated via soft update after every step:

$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i, \qquad \phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i, \qquad \tau = 0.01 \tag{7}$$

Without target networks, the training target y in the critic loss would shift every step as the critic weights update, causing oscillation. Moving only 1% toward the current weights per step keeps the target stable while still gradually improving.

2.4 Q-Value Update and Training

2.4.1 Replay Buffer

All experience is stored in a shared replay buffer as joint tuples:

$$\mathcal{D} = \{(x^{(t)}, a_1^{(t)}, \ldots, a_N^{(t)}, r_1^{(t)}, \ldots, r_N^{(t)}, x'^{(t)}, \text{done}^{(t)})\} \tag{8}$$

The buffer holds $|\mathcal{D}| = 100{,}000$ transitions. At each training step, a random minibatch of size 64 is sampled. Random sampling breaks temporal correlation between consecutive steps. Without it, consecutive observations are nearly identical and the network overfits to recent experience.

Training begins only after a warm-up phase fills the buffer with $10 \times \text{batch_size} = 640$ transitions from random actions.

2.4.2 Critic Update — Q-Value Calculation

The critic for agent i is updated using the Bellman equation. Given a sampled minibatch $\{(x^{(j)}, \mathbf{a}^{(j)}, r_i^{(j)}, x'^{(j)})\}_{j=1}^{B}$:

Step 1 — Compute the target Q-value using target networks:

$$y^{(j)} = r_i^{(j)} + \gamma \cdot Q'_{\phi'_i}\Big(x'^{(j)},\ \mu'_{\theta'_1}(o_1'^{(j)}),\ \ldots,\ \mu'_{\theta'_N}(o_N'^{(j)})\Big) \tag{9}$$

where $\gamma = 0.95$ is the discount factor. The target actor μ' provides the next actions so no exploration noise is needed for the target.

Step 2 — Compute current Q-value using the live critic:

$$\hat{Q}^{(j)} = Q_{\phi_i}(x^{(j)}, a_1^{(j)}, \ldots, a_N^{(j)}) \tag{10}$$

Step 3 — Minimize the Bellman error:

$$\mathcal{L}(\phi_i) = \frac{1}{B}\sum_{j=1}^{B}\Big(\hat{Q}^{(j)} - y^{(j)}\Big)^2 \tag{11}$$

Gradients $\nabla_{\phi_i}\mathcal{L}$ are backpropagated through the critic network and weights are updated via Adam optimizer. Importantly, the target value $y^{(j)}$ is treated as a constant (computed with `torch.no_grad()`) — gradients do not flow through target networks.

Once the critic is updated, it serves as a differentiable scoring function for the actor. The actor for agent i is updated to maximize the expected Q-value

Independent DDPG. Each agent trains its own actor and critic independently. The critic receives only the agent's own observation and action, with no information about other agents. This is the naive multi-agent baseline and violates stationarity as other agents' policies change during training.

Network architecture. Both actor and critic networks are two-layer MLPs with 64 hidden units and ReLU activations. Actor outputs use tanh activation to bound actions to $[-1, +1]$. All network input dimensions are computed dynamically from the environment at runtime. Training uses a shared replay buffer of 100,000 joint transitions $(o_{\text{all}}, a_{\text{all}}, r_{\text{all}}, o'_{\text{all}}, \text{done})$, batch size 64, optimizer with learning rate 0.001, soft target update $\tau = 0.01$, and Gaussian exploration noise decaying from $\sigma = 0.3$ to $\sigma = 0.01$. A warm-up phase fills the buffer before any gradient updates begin.

2.5 Evaluation Metrics

We compute four metrics over the last 100 training episodes and 20 dedicated evaluation episodes:

- **Cooperative reward (last 100):** Mean per-episode total reward for cooperative agents.
- **Adversary reward (last 100):** Mean per-episode adversary reward. Less negative means the adversary is more successful.
- **Adversary reach rate:** Percentage of evaluation episodes where any adversary gets within distance 0.1 of the true target. Directly comparable to Lowe et al. Table 4.
- **Coverage spread:** Mean pairwise distance between cooperative agents at episode end. Higher values indicate agents are distributing across landmarks the geometric signature of deceptive behavior.

3 Results

3.1 Learning Curves

Figure 2 shows smoothed learning curves across all eight training runs. Most configurations converge to positive cooperative reward within 10,000 episodes, confirming that agents learn to successfully deceive the adversary under the corrected reward formulation. Two configurations show notable instability: MADDPG 3g_1a oscillates substantially through episode 6,000 before partially stabilizing, and MADDPG 3g_2a exhibits a sharp dip around episode 4,000 before recovering. Both are consistent with the non-stationarity effects described by Lowe et al. [1], where larger agent counts amplify training instability.

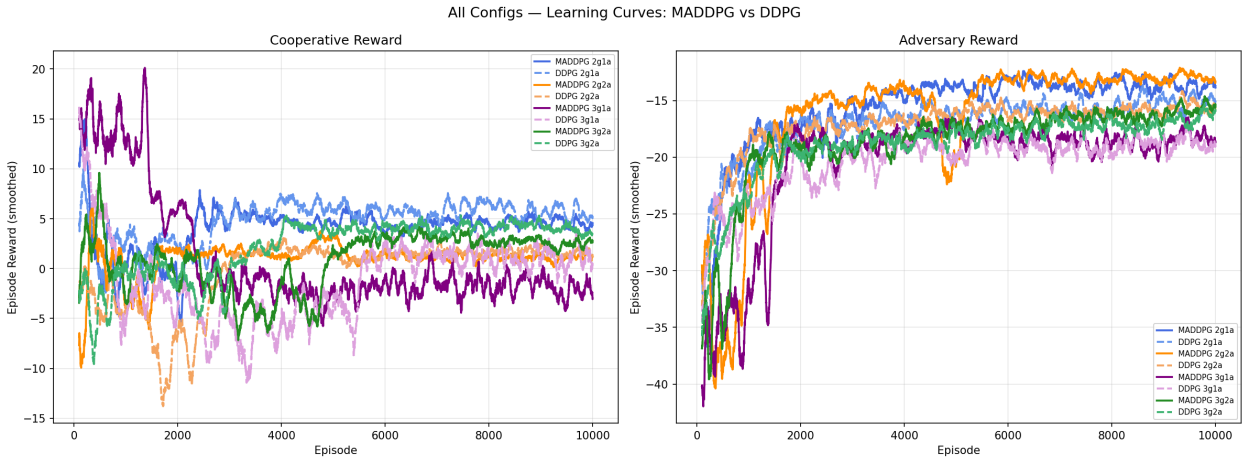


Figure 2: Smoothed learning curves for all eight training runs (4 configurations $\times$ 2 algorithms) over 10,000 episodes. Left: cooperative reward. Right: adversary reward.

3.2 Quantitative Results

Table 1 summarizes evaluation metrics across all configurations and algorithms.

Table 1: Evaluation metrics across all configurations. Coop Reward and Adv Reward are averaged over the last 100 training episodes. Adv Reach Rate is the fraction of evaluation episodes where any adversary reaches within distance 0.1 of the true target. Coverage Spread is the mean pairwise distance between cooperative agents at episode end. Convergence Step is the first episode where the 100-episode rolling average cooperative reward exceeds 50% of the random policy baseline.

Config	Algo	Coop Reward	Adv Reward	Adv Reach Rate	Coverage Spread	Conv. Step
2g_1a	MADDPG	4.535	−13.857	0.350	0.566	99
2g_1a	DDPG	5.088	−15.105	0.170	0.428	99
2g_2a	MADDPG	1.222	−13.327	0.480	0.409	284
2g_2a	DDPG	1.194	−15.558	0.420	2.817	2906
3g_1a	MADDPG	−2.984	−18.672	0.120	2.763	99
3g_1a	DDPG	0.544	−18.841	0.130	3.168	99
3g_2a	MADDPG	2.769	−15.552	0.320	3.582	133
3g_2a	DDPG	3.491	−15.999	0.330	1.235	944

3.2.1 Baseline: 2 cooperative agents vs 1 adversary

Both algorithms achieve positive cooperative reward at convergence (4.535 for MADDPG, 5.088 for DDPG) and converge at episode 99, indicating the 2v1 task is straightforward for both centralized and independent learning. The adversary reach rate reveals a notable difference: DDPG cooperative agents hold the adversary to 17% reach rate compared to 35% for MADDPG. This suggests that in the simplest configuration, independently trained agents find a coverage strategy that is at least as effective as the centralized approach, and in this case slightly more effective at blocking the adversary.

3.2.2 Scaled adversaries: 2 cooperative agents vs 2 adversaries

Adding a second adversary reduces cooperative reward substantially to approximately 1.2 for both algorithms. While final rewards are similar, convergence speed differs dramatically: MADDPG reaches the threshold at episode 284 while DDPG requires 2,906 episodes a $10\times$ difference. This is the strongest evidence in the experiment that centralized training provides a meaningful advantage when the coordination problem scales.

Coverage spread also diverges: DDPG agents spread to 2.817 while MADDPG agents cluster at 0.409 yet achieve equivalent reward. This suggests MADDPG agents coordinate a more efficient joint coverage strategy, while DDPG's higher spread reflects slow uncoordinated divergence during the extended convergence period rather than deliberate deception.

3.2.3 More cooperative agents: 3 cooperative agents vs 1 adversary

The 3v1 configuration produces the most unexpected result in the experiment. With three cooperative agents against a single adversary, MADDPG achieves a negative cooperative reward (-2.984) while DDPG achieves a positive reward (0.544), making this the only configuration where MADDPG is clearly outperformed on cooperative reward. Both algorithms converge at episode 99 and hold the adversary to the lowest reach rate across all configurations (12% and 13%), confirming that the adversary is rarely finding the target.

The negative cooperative reward for MADDPG despite excellent adversary suppression suggests that MADDPG's three agents are not efficiently covering the target they are keeping the adversary away but failing to maintain close proximity to the target themselves. The pronounced oscillation visible in the MADDPG 3g_1a learning curve through episode 6,000 supports this interpretation: the centralized critic with three actors and one adversary may be harder to stabilize than with fewer agents, causing the actors to adopt suboptimal positional strategies. DDPG agents, learning independently, may avoid this instability and find a more stable individual coverage policy.

3.2.4 More cooperative agents: 3 cooperative agents vs 2 adversaries

Adding a second adversary to the 3-agent cooperative team partially recovers performance relative to 3v1, with both algorithms achieving positive cooperative reward (2.769 for MADDPG, 3.491 for DDPG). MADDPG again converges faster (episode 133 vs 944 for DDPG), a $7\times$ difference, consistent with the convergence advantage seen in 2v2.

Coverage spread is notably higher for MADDPG (3.582) than DDPG (1.235). Unlike the 2v2 case where MADDPG's lower spread corresponded to efficient coordination, here MADDPG's higher spread coexists with lower cooperative reward than DDPG. This mirrors the 3v1 pattern and suggests that with three cooperative agents, MADDPG may be over-distributing spreading agents further than necessary at the cost of target coverage.

4 Conclusions

This project implements MADDPG and Independent DDPG on the Physical Deception task across four agent configurations, ranging from 2 cooperative agents vs 1 adversary to 3 cooperative agents vs 2 adversaries. The corrected reward formulation produces positive cooperative rewards in most configurations, confirming that trained agents successfully deceive the adversary.

Two clear patterns emerge across configurations. First, MADDPG consistently converges faster than DDPG when the coordination problem is non-trivial. In the 2v2 configuration MADDPG converges $10\times$ faster, and in the 3v2 configuration $7\times$ faster. In simple configurations (2v1, 3v1) both algorithms converge at the same speed, confirming that MADDPG's advantage is specifically tied to the complexity of joint coordination. This supports the theoretical motivation of MADDPG: centralized training provides a stationary learning signal that becomes increasingly valuable as the number of agents and the size of the joint action space grow.

Second, raw cooperative reward at convergence does not consistently favor either algorithm. DDPG matches or slightly exceeds MADDPG on final reward in three of four configurations. The most striking exception is the 3v1 configuration, where MADDPG produces negative cooperative reward despite holding the adversary to the lowest reach rate in the entire experiment. This reveals a failure mode specific to MADDPG: with three actors sharing a centralized critic against a single adversary, the critic's optimization pressure may drive agents toward adversary suppression at the expense of target coverage, a form of reward hacking that independent DDPG agents avoid by optimizing their own coverage individually.

Coverage spread proved to be a nuanced and context-dependent metric. In the 2v2 configuration, MADDPG's lower spread indicates efficient joint coordination. In the 3v1 and 3v2 configurations, MADDPG's higher spread corresponds to worse coverage, suggesting over-distribution. This highlights that coverage spread is only meaningful when interpreted alongside adversary reach rate and cooperative reward together spread alone does not indicate whether deception is working.

The central practical takeaway is that MADDPG's value lies primarily in sample efficiency rather than asymptotic performance. When simulation or training time is limited, MADDPG reaches effective policies substantially faster. When training time is unconstrained, independent DDPG can match MADDPG's final performance and in some cases particularly when the number of cooperative agents exceeds the number of adversaries may produce more stable and better-performing policies.

An interesting direction for future work is extending this framework to real-time settings, where agents must act without waiting for a full replay buffer update cycle. Real-Time Reinforcement Learning [2] eliminates action delay by updating the critic at every step rather than sampling from a buffer, which is critical for physical deployment scenarios such as multi-robot coordination or adversarial cybersecurity systems where threats and responses occur on continuous timescales rather than discrete simulated episodes.

References

[1] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[2] S. Ramstedt and C. Pal, "Real-Time Reinforcement Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. https://arxiv.org/abs/1911.04448

[3] J. Terry, B. Black, N. Grammel et al., "PettingZoo: Gym for Multi-Agent Reinforcement Learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[4] W. Mao, J. Yin, Y. Liu, B. Cho, Y. Chen, W. Rao, and Y. Xiao, "Multi-agent Reinforcement Learning-based Capacity Planning for On-demand Vehicular Fog Computing," *TechRxiv*, 2023. https://doi.org/10.36227/techrxiv.24184047.v2

Code

The full implementation, training scripts, and evaluation code are available at:
https://github.com/Steffi-Roy/MADDPG_DDPG_MultiAdversary_Physical_Deception