Curiosity in Multi-Agent Reinforcement Learning

Lukas Schäfer

Master of Science

Informatics

School of Informatics

University of Edinburgh

2019

Abstract

Multi-agent reinforcement learning has seen considerable achievements on a variety of tasks. However, suboptimal conditions involving sparse feedback and partial observability, as frequently encountered in applications, remain a significant challenge. In this thesis, we apply curiosity as exploration bonuses to such multi-agent systems and analyse their impact on a variety of cooperative and competitive tasks. In addition, we consider modified scenarios involving sparse rewards and partial observability to evaluate the influence of curiosity on these challenges.

We apply the independent Q-learning and state-of-the-art multi-agent deep deterministic policy gradient methods to these tasks with and without intrinsic rewards. Curiosity is defined using pseudo-counts of observations or relying on models to predict environment dynamics.

Our evaluation illustrates that intrinsic rewards can cause considerable instability in training without benefiting exploration. This outcome can be observed on the original tasks and against our expectation under partial observability, where curiosity is unable to alleviate the introduced instability. However, curiosity leads to significantly improved stability and converged performance when applied to policy-gradient reinforcement learning with sparse rewards. While the sparsity causes training of such methods to be highly unstable, additional intrinsic rewards assist training and agents show intended behaviour on most tasks.

This work contributes to understanding the impact of intrinsic rewards in challenging multi-agent reinforcement learning environments and will serve as a foundation for further research to expand on.

Acknowledgements

This project is the accumulation of a stressful and highly rewarding year of studies in Edinburgh. Along the way, many provided guidance and motivation. First, I would like to express my deepest gratitude to my supervisor Dr. Stefano V. Albrecht. His counsel and support were invaluable during the initial development and execution of this project. Besides assisting in this project, he largely contributed to my academic growth, inviting me to his research group and guiding my path towards a subsequent PhD.

Secondly, I want to thank Filippos Christianos and Georgios Papoudakis at the autonomous agents research group. Their consultation regarding reproducibility of baseline experiments and efficient implementation of RL algorithms was invaluable to the success of this project. I am also very thankful for the help provided by Muhammad Arrasy Rahman and Ibrahim Ahmed. Their insight into computing services of the university and beyond was highly appreciated and assisted initial evaluation.

I also want to thank the RL reading group at the University of Edinburgh for their critical discussion of fundamental underlying ideas around the concept of curiosity. Their comments contributed to my understanding of the approaches and raised essential ideas that shaped the further project. Especially, I want to thank Muhammad Arrasy Rahman, Georgios Papoudakis and Filippos Christianos for their remarks.

My brother Daniel Schäfer and Lukas Schwitzgebel provided considerable feedback on the latest draft. I am grateful for their remarks which resulted in improved clarity and language of this work.

The German Academic Exchange Service (DAAD) made my studies in Edinburgh possible with their generous support through the graduate scholarship. I also want to specifically thank my parents and whole family. Their unconditional financial and emotional support allowed for this year abroad and motivated me.

Lastly, I want to thank my friends in Edinburgh. Special thanks goes to Iris Yang, Andrei-Alexandru Apostoae, Hansun Lee and Kiyoon Kim. Their friendship and encouragement during my time in Edinburgh made for an eventful year I am always going to remember fondly.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Lukas Schäfer*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Humans constantly interact with other people or machines to perform tasks. They learn such interactive behaviour from a young age, largely relying on observation and an intrinsic curiosity [Berlyne, 1965].

Multi-agent reinforcement learning (MARL) is concerned with such behaviour learning of multiple agents. Throughout training, agents continuously interact with the environment and receive feedback in the form of numeric rewards. While MARL research has already led to successful applications, particularly in game-playing [Mnih et al., 2015; OpenAI, 2018a; Silver et al., 2017; Vinyals et al., 2019], it still faces considerable challenges under suboptimal conditions. Such conditions include *partial observability*, under which agents only receive incomplete information about the environment. Many autonomous systems operate under imperfect information, making this challenge essential for future applicability of MARL. Another challenge often encountered is *sparsity of rewards*, i.e. agents do not receive constant, meaningful feedback, but only get such rewards sparsely.

1.2 Overview

Solving tasks by learning the intended behaviour under such conditions requires extensive exploration. In this context, *exploration* refers to the attempt of various strategies to discover new promising behaviour. There are many possible approaches for efficient exploration [Thrun, 1992]. In this dissertation, we apply *curiosity* [Schmidhuber, 1991a] to MARL, rewarding agents for discovering novel or poorly understood parts

of the environment. Agents are trained to compute these *intrinsic rewards* [Chentanez et al., 2005] to motivate exploration and assist learning independent of environmental rewards.

We apply three recent definitions of such intrinsic rewards to value-based and policy gradient MARL methods. Our research focuses on the application of curiosity in MARL in general and its impact on exploration. The influence of decentralised curiosity, where each agent independently computes its intrinsic rewards, as well as applying a joint curiosity model, is analysed. Our evaluation on the multi-agent particle environment considers the original cooperative and competitive tasks. Additionally, modified variations of these tasks are used to analyse the impact of curiosity under partial observability and with sparse rewards.

We find that curiosity only introduces slight disturbance during training on the original tasks. Surprisingly, the same can also be observed under partial observability, where neither joint nor decentralised curiosity could assist exploration. However, we find that curiosity leads to considerable improvements in training stability and converged performance for policy gradient MARL during training with sparse rewards.

1.3 Outline

This dissertation is structured as follows. Chapter 2 introduces preliminary information on MARL research and necessary formalisation. We describe applied baseline approaches, the general concept of intrinsic reward and its application in reinforcement learning. The third chapter explains our methodology involving the concrete implementation of all algorithms including their network architecture and optimisation objectives. After highlighting these details, the independent and joint curiosity definitions for MARL are compared and the training cycle is illustrated. In Chapter 4, the experimental conditions of our evaluation, as well as the conducted tasks, are explained. This also includes our modifications for partial observability and sparse rewards. Lastly, we propose future research topics building on top of our findings before we conclude with final remarks on this project.

For reproducibility, we include implementation dependencies and a complete list of parameters chosen for the evaluation in appendices A and B. For more illustrations of the evaluation results, we include figures and tables in appendix C.

Chapter 2

Background

2.0.1 Multi-Agent Reinforcement Learning

MARL is a general framework for learning behaviour in multi-agent systems. It is the extension of *reinforcement learning* (RL) which has led to various successful applications for single-agent tasks. In MARL, each agent is trained to learn a policy for action-selection by repeatedly interacting with the environment and optimising their policies. The objective of such training is to learn a policy that maximises future cumulative rewards.

Due to its wide field of applications, MARL research becomes increasingly prominent. Many autonomous tasks require multiple parties to interact with each other and hence a framework for *cooperative* and *competitive* behaviour learning is required.

Despite its existing success in complex environments, e.g. the video games Dota [OpenAI, 2018a] and StarCraft [Vinyals et al., 2019], there are still challenges that have to be addressed. One of the essential dilemmas of RL and MARL in particular is the balance of *exploration* and *exploitation*. Agents need to explore the environment to discover new behaviour to guarantee eventually finding the optimal policy. However, this sacrifices short-term rewards, which could be achieved by greedily exploiting already existing knowledge. Such greedy exploitation simply follows the behaviour appearing most promising so far. This problem becomes increasingly complex in MARL, as exploration does not just depend on the action choice of a single agent, but all involved agents. In most tasks, it is desirable for exploration to be achieved collaboratively rather than letting each agent explore independently, which ignores the presence of other agents.

Another challenge of MARL is the *credit-assignment problem* concerned with

identifying the responsible agents and actions for received rewards. The complexity of this task is closely connected to the sparsity of rewards, i.e. the frequency in which feedback is provided. If agents are continuously given meaningful feedback for each of their actions, it becomes much easier to identify their impact. Imagine a team of football players scoring a goal. It is highly complex to identify which individual actions of each player contributed towards the goal and if so to what extent.

2.1 Markov Decision Process

Sequential decision-making problems, as solved by RL, are usually formalised as *Markov decision processes* (MDPs) [Howard, 1964]. Formally an MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ with a set of states $\mathcal{S}$, actions $\mathcal{A}$ and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ describes the environment dynamics in the form of transition probabilities $\mathcal{P}(s,a,s')$ of reaching state s' after applying action a in s. Similarly, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ represents the reward $\mathcal{R}(s,a,s')$ for such a transition and γ denotes the discount factor to define the expected return in Equation (2.1).

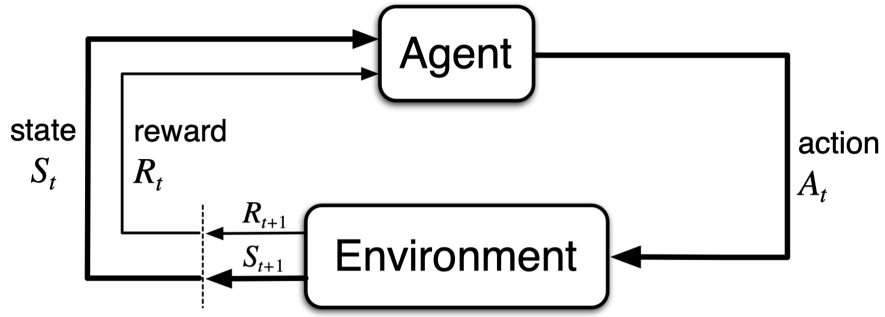

Figure 2.1: Agent-environment interaction in an MDP. Figure taken from Sutton and Barto [2018]

Figure 2.1 illustrates the general control loop of MDP tasks. The agent repeatedly receives the current state s_t and reward r_t for the previous decision and decides its action a_t based on the inputs. s_t and a_t together determine the next state s_{t+1} as well as the new reward r_{t+1} to conclude the cycle. The loop generally assumes the *Markov property*, i.e. future states and rewards only depend on the last state and action. Hence optimal decisions can be made solely based on the current state.

The goal of any RL agent in these tasks is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that decides which actions to take based on the current state or observation. Such a policy is trained to maximise the expected (discounted) return G_t, i.e. the discounted sum of future

rewards:

$$G_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1} \tag{2.1}$$

The discount factor $\gamma \in [0,1)$ is introduced to account for continuous tasks without terminal time step, balancing short- and long-term rewards.

2.1.1 Partially Observable Stochastic Game

Usually, it is unrealistic to assume that agents have full information about the state of the environment. Imagine e.g. the game of Poker. Each player only has information about open cards and its own hand but is unable to observe other players' cards. To represent such tasks involving multiple agents, *partially observable stochastic games* (POSGs) are introduced [Hansen et al., 2004]. Similar to MDPs, these problems can be described as tuples $(I, \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, O, \mathcal{R}, \gamma)$ where $\mathcal{S}$, $\mathcal{P}$ and γ are defined as for a simple MDP. $I = \{1, ..., N\}$ is a set of agents, $\mathcal{A} = A_1 \times ... \times A_N$ denotes the joint action set. Similarly, $\Omega = \Omega_1 \times ... \times \Omega_N$ is the set of joint observations. Instead of receiving information about the full state, agents receive these observations based on the transition function $O : \mathcal{S} \times \mathcal{A} \times \Omega \to [0,1]$ with $O(s,a,o)$ representing the probability to receive a certain observation o when joint action a is applied in state s. Each agent receives individual rewards based on its reward function $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. For collaborative tasks, these reward functions will often be identical.

2.2 Reinforcement Learning Methods

2.2.1 Foundation

There are multiple approaches in RL to solve MDPs by learning the optimal policy π^* that maximises the expected return

$$\mathbb{E}_{\pi^*}\left[G_t \mid a_t = \pi^*(s_t)\right] = \max_{\pi} \mathbb{E}_{\pi}\left[G_t \mid a_t = \pi(s_t)\right] \tag{2.2}$$

This expected return is often formalised as the *state-value function* $V_\pi(s)$ of a policy describing the expected return by following the policy in state s

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid s_t = s\right] && (2.3)\\ &= \mathbb{E}_\pi\left[r_t + \gamma\, G_{t+1} \mid s_t = s\right] && (2.4)\\ &= \sum_a \pi(a \mid s) \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma\, V_\pi(s')\right] && (2.5) \end{aligned}$$

Equation (2.5) represents the so-called recursive *Bellman equation*. Similarly, an *action-value function* $Q_\pi(s,a)$ can be defined as the expected reward after applying action a in s

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid s_t = s, a_t = a\right] \tag{2.6}$$

$$= \sum_{s'} P(s,a,s')\left[R(s,a,s') + \gamma V_\pi(s')\right] \tag{2.7}$$

$$= \sum_{s'} P(s,a,s')\left[R(s,a,s') + \gamma \sum_{a'} \pi(a' \mid s')\, Q_\pi(s',a')\right] \tag{2.8}$$

These equations serve as the foundation for *temporal-difference* (TD) learning [Sutton, 1988] methods, which are iteratively updating value function approximations after each step taken in the environment

$$V(s) = V(s) + \alpha\left(R(s,a,s') + \gamma V(s') - V(s)\right) \tag{2.9}$$

where α is a predetermined step-size referred to as *learning rate*.

2.2.2 Q-Learning

Q-learning [Watkins and Dayan, 1992] is the most common TD algorithm and aims to compute an action-value function approximation using off-policy updates, i.e. a' is chosen without considering the followed policy π.

$$Q(s,a) = Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.10}$$

2.2.2.1 Deep Q-Learning

However, such tabular value-function approaches do not scale to tasks with high-dimensional state-spaces. It can not be expected that the same state will be encountered many times, hence generalisation is essential to learn a representative value function. This difficulty can be addressed using neural networks. Such networks serve as powerful function approximations and have arguably been one of the most influential advances in AI [LeCun et al., 2015].

Deep Q-networks (DQNs) reached human-level performance on multiple Atari games [Mnih et al., 2013, 2015]. The approach trains a network to approximate the Q-function by minimising the following loss with respect to network parameters θ

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}}\left[\left(r + \gamma \max_{a'} Q(s',a';\bar{\theta}) - Q(s,a;\theta)\right)^2\right] \tag{2.11}$$

where $\bar{\theta}$ are delayed parameters which are periodically updated towards θ. Such a target network structure assists in stabilising training. Furthermore, the introduction of a *replay memory* [Lin, 1992] $\mathcal{D}$ is crucial for training. It allows to efficiently sample large batches of (uncorrelated) experience steps (s, a, r, s') to use for optimisation.

2.2.2.2 Independent Q-Learning

Independent Q-learning (IQL) [Tan, 1993] is the logical extension of Q-learning for multi-agent systems. Each agent is simply maintaining its Q-function conditioned on its actions and state or observations in case of partial observability. Thus, agents (wrongly) assume that the environment remains stationary despite the presence of further actors. While this leads to suboptimal performance in cooperative games [Matignon et al., 2012], the approach remains a simple MARL baseline, which can easily be generalised using DQNs.

2.2.3 Policy Gradient Methods

Value function approaches are widely used in RL, but they are restricted to finite and discrete action sets. *Policy gradient methods* aim to overcome this limitation by directly computing a policy without intermediate value function. This makes those approaches better suited for environments of many or continuous actions, but often sacrifice convergence speed and stability [Duan et al., 2016]. Policy parameters θ are optimised using the gradient of a score function J. Such a function is approximately sampled based on the policy gradient theorem [Sutton et al., 2000]: For any differentiable policy π, its gradient $\nabla J(\theta)$ can be written as

$$\nabla J(\theta) = \sum_s d_\pi(s) \sum_a Q_\pi(s,a)\ \nabla \pi(a \mid s;\theta) \tag{2.12}$$

$$= \mathbb{E}_\pi \left[Q_\pi(s_t, a_t)\ \nabla \ln \pi(a_t \mid s_t;\theta) \right] \tag{2.13}$$

where $d_\pi(s)$ represents the on-policy distribution of states encountered when following π. Based on this theorem, θ can be updated as below

$$\theta_{t+1} = \theta_t + \alpha\ Q_\pi(s_t, a_t)\ \nabla \ln \pi(a_t \mid s_t;\theta_t) \tag{2.14}$$

where $Q_\pi(s_t, a_t)$ is approximated by e.g. the total return during an episode as used by REINFORCE [Williams, 1992].

Actor-critic methods [Konda and Tsitsiklis, 2000] use a value function approximation $\hat{V}$, the *critic*, to update the policy referred to as the *actor*.

$$\theta_{t+1} = \theta_t + \alpha \left(r_{t+1} + \gamma \hat{V}(s_{t+1}; w) - \hat{V}(s_t; w) \right) \nabla \ln \pi(a_t \mid s_t; \theta_t) \tag{2.15}$$

While the critic and actor both require optimisation, actor-critic algorithms proved to be efficient and often more stable than individual value function or policy approaches [Mnih et al., 2016].

2.2.3.1 Multi-Agent Deep Deterministic Policy Gradient

Multi-agent deep deterministic policy gradient (MADDPG) [Lowe et al., 2017] is an extension of the widely used deep deterministic policy gradient (DDPG) [Lillicrap et al., 2015] algorithm for multi-agent systems. It trains a critic and actor network for each agent with the general architecture outlined in Figure 2.2.

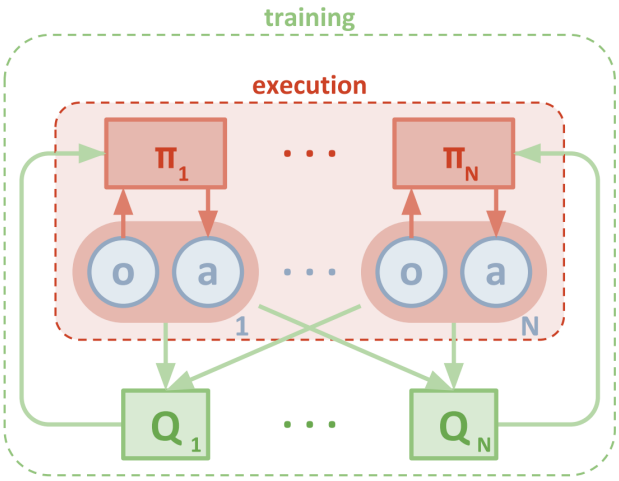

Figure 2.2: Outline of the MADDPG architecture. Each agent maintains its policy network π_i to choose actions based on its individual observations o_i. Joint observations and actions are the inputs for critic networks, Q_i. Figure taken from Lowe et al. [2017]

The critics are conditioned on the joint observations and actions for centralised training. However, execution remains decentralised as only the policy networks are required. These can be applied independently for each agent using their own observations as input.

Formally, each agent is training its policy π_i with parameters θ_i using the respective gradient

$$\nabla J(\theta_i) = \mathbb{E}_{o, a_i \sim \pi_i} \left[Q_i(o, a_1, \ldots, a_N) \nabla \ln \pi_i(a_i \mid o_i; \theta_i) \right] \tag{2.16}$$

where Q_i corresponds to the centralised critic of agent i. Its input consists of the joint observation $o = (o_1, ..., o_N) \in \Omega^N$ as well as the chosen actions of all agents $a_1, ..., a_N$. The policy input solely consists of the individual observation o_i to choose action a_i. Centralised critics for deterministic action policies are optimised with respect to the following loss function

$$\mathcal{L}(\theta_i) = \mathbb{E}_{o,a,r,o' \sim \mathcal{D}} \left[\left(r_i + \gamma \, \bar{Q}_i(o', a'_1, ..., a'_N) \mid_{a'_j = \bar{\mu}_j(o_j)} - Q_i(o, a_1, ..., a_N) \right)^2 \right] \quad (2.17)$$

where μ_j corresponds to the deterministic policy of agent j and $\bar{\mu}_j$, $\bar{Q}_j$ represent the target policy and critic respectively with delayed parameters. In practice, tuples (o, a, r, o') of joint observations of current o and next observation o' as well as chosen actions a and received rewards r will be stored in an experience replay $\mathcal{D}$ for efficient training.

MADDPG was shown to be a powerful method for cooperative and competitive behaviour learning alike and led to state-of-the-art performance on the multi-agent particle environment[1] [Lowe et al., 2017].

2.3 Intrinsic Rewards

Intrinsic motivation is a widely researched area in psychology concerned with activities motivated by their inherent satisfaction rather than their consequences [Ryan and Deci, 2000]. Such motivation leads to exploratory behaviour, which can be observed in many organisms [White, 1959], particularly human infants [Berlyne, 1965].

Inspired by these concepts, a variety of computational methods have been proposed to replicate curiosity for efficient exploration in RL [Barto, 2013] and robotics [Oudeyer and Kaplan, 2009; Oudeyer et al., 2008, 2007]. These exploration bonuses are a form of *reward shaping*, trying to compute additional rewards to support training [Devlin and Kudenko, 2012]. While some reward shaping has been applied to multi-agent systems [Babes et al., 2008], curiosity has largely been restricted to single-agent RL. After elaborating on existing ideas in RL, we will briefly refer to the limited research done on intrinsic rewards for MARL.

2.3.1 Count-based Curiosity

Most definitions of intrinsic reward as exploration incentives are based on uncertainty in the environment. One way of estimating such uncertainty is by counting observa-

[1] More information on this environment will be provided in Section 4.1.1 as it serves as our primary evaluation testbed.

tions, or observation-action pairs. The more frequent parts of the environment are visited, the less novel and hence interesting they are for exploration. Therefore, the agent is rewarded inverse proportionally to the count of encountered observations. This approach can easily be applied to small, discrete state-spaces, but it requires sophisticated methods to deal with continuous or large state spaces. In such environments, observations are often only encountered once or not at all. Thus, generalisation, as for value function approaches, is necessary.

Bellemare et al. [2016] overcome this challenge by introducing pseudo-counts for observation-action pairs. These are derived using a density model predicting the recoding probabilities of states. The pseudo-counts improved exploration and thereby learning progress considerably in multiple Atari games [Bellemare et al., 2013], most notably the exploration-heavy game Montezuma's Revenge. The approach was further refined with an advanced density model for visual observations and loosened requirements on the model overall [Ostrovski et al., 2017] to improve stability and learning speed further.

Tang et al. [2017] propose generalising by computing approximate locally-sensitive hash-values of observations. These are used to maintain hash tables of counts as a density model and reward the agent as previously inverse proportional to the counts. Locality-sensitive hashing is used to group neighboured observations based on similarity metrics [Andoni and Indyk, 2008]. The paper specifically applies the SimHash [Charikar, 2002] function measuring angular distance based on the sign of a randomised mapping. This makes the approach computationally efficient and comparably simple.

2.3.2 Prediction-based Curiosity

Instead of counting observations, environment uncertainty can also be estimated using predictions on environment dynamics. The impact of actions on the environment is predicted and agents are rewarded for their prediction error to reduce uncertainty over time. Schmidhuber [1991b] introduced such a curiosity bonus and defined the intrinsic reward as the discrepancy between observed and predicted observation.

Schmidhuber also identified the problem of stochastic observations, which due to their randomness introduce unpredictability. The proposed solution of boredom aimed to diminish the curiosity rewards for observations not considered "learnable". However, it remains a major challenge to efficiently identify such observations.

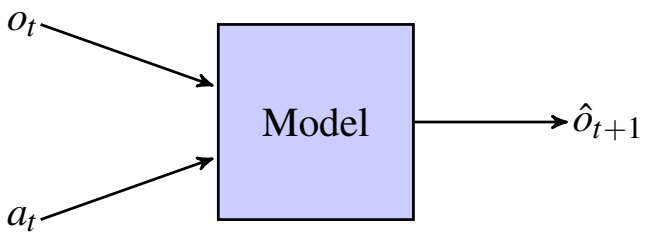

Figure 2.3: Most prediction-based curiosity approaches are constructing models of environment dynamics to predict the upcoming observation o_{t+1} using current observation o_t and chosen action a_t.

Stadie et al. [2015] constructed a similar network, encoding observations using an autoencoder network [Hinton and Salakhutdinov, 2006], which was optimised online during RL training. Such efficient representations enabled environment prediction in high-dimensional environments and led to considerable exploration in Atari games.

2.3.2.1 Intrinsic Curiosity Module

Pathak et al. [2017] propose the *intrinsic curiosity module* (ICM) to learn efficient observation representations. They suggested training a network to predict the applied action given original and next observation representations. The entire architecture is outlined in Figure 2.4.

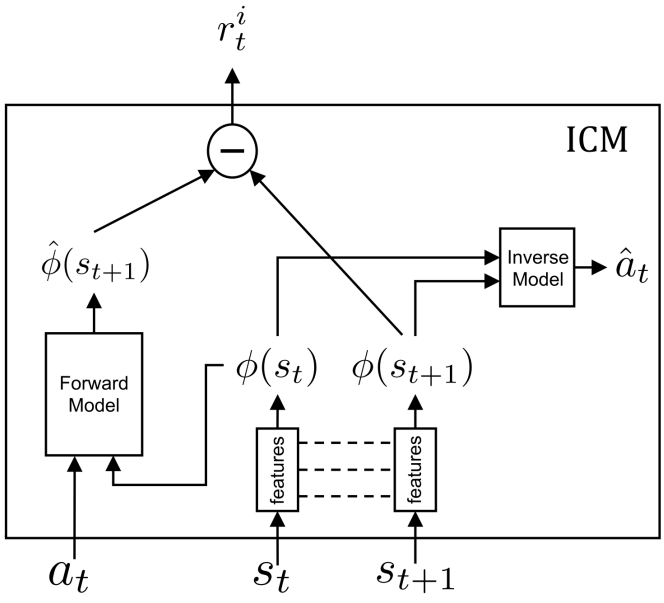

Figure 2.4: The ICM learns feature representation ϕ for states through the inverse model predicting applied action a_t. A prediction $\hat{\phi}(s_{t+1})$ of the upcoming state representation is computed using the forward model. Figure taken from Pathak et al. [2017]

Through such self-supervised training, the network is incentivised to only include features in the observation representations, which affect the predicted action or are potentially affected by it. The computed intrinsic reward is defined as the difference in predicted and encountered observation representations. Due to the trained representation network, the approach led to considerable progress in domains of high-dimensional input, impressively even in the absence of extrinsic rewards or presence of noise occupying parts of observations.

2.3.2.2 Random Network Distillation

While ICM has been highly effective in some complex tasks, it suffers from the so-called "noisy-TV" problem [Burda et al., 2018a]. Noise, which is integrated in the environment, consistently leads to high curiosity and disturbs training. *Random network distillation* (RND) [Burda et al., 2018b] has been proposed to deal with such stochastic observations. Other than previous prediction-based curiosity concepts, it only attempts to compute observation representations without considering actions, as shown in Figure 2.5. Observation novelty is estimated based on the distance of computed representations of a prediction and a randomly initialised, fixed target network. Hence, the prediction network is trained to mirror the target representations.

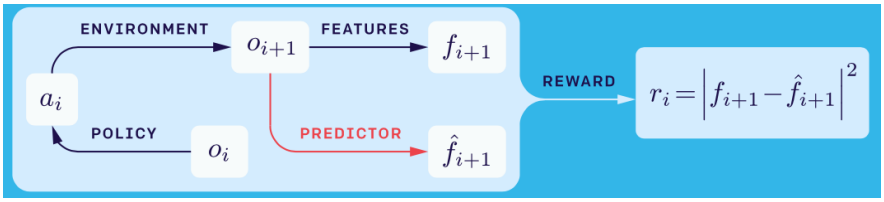

Figure 2.5: Outline of the feature prediction computed by RND. The trained predictor network computes its predicted representation solely relying on observation o_{i+1}. Figure taken from OpenAI [2018b]

Due to the lack of consideration of actions and effects in the environment, the approach is invariant to noisy observations. However, it assumes poor generalisation among observations to avoid premature decay of curiosity.

2.3.3 Intrinsic Rewards in Multi-Agent Reinforcement Learning

There have been few attempts of applying intrinsic rewards in MARL, presumably due to the increased challenge in multi-agent systems. Non-stationarity makes environment

predictions largely unreliable and count-based approaches face increased scalability problems due to the large joint action- and observation-space.

However recently, novel approaches to apply curiosity as exploration incentives to MARL have been proposed. *Independent centrally-assisted Q-learning* (ICQL) [Böhmer et al., 2019] introduces a centralised, intrinsically motivated agent to typical IQL to stabilise the impact of such exploration incentives. This agent is only applied during training and shares a replay buffer with decentralised IQL agents, which are trained using extrinsic reward alone. Through this framework, the unreliable influences of potentially misleading intrinsic rewards can be avoided while making use of its exploration incentives.

Lastly, Iqbal and Sha [2019] define an ensemble of intrinsic rewards based on simple count-based novelty bonuses. Each agent maintains such a novelty function $\frac{1}{N^{\zeta}}$ where N is a count of observation occurrences and ζ represents a decay rate. Intrinsic rewards are defined as varying combinations of all agents' novelty functions. They are e.g. formulated as the minimum or average over all novelties. For each agent, a soft actor-critic policy [Haarnoja et al., 2018] is simultaneously trained for every reward by using a shared replay buffer. Additionally, a high-level selector policy is trained to dynamically decide which policy should be applied for a given task. Such a decision is conducted for each agent at the beginning of a new episode. This dynamic selection consistently led to performance comparable or better than the best individual intrinsic reward approach for any task and eliminates the need for preselecting these bonuses.

Chapter 3

Methodology

The primary objective of this project is to evaluate the impact of curiosity as exploration incentives for MARL in varying competitive and cooperative tasks. Chapter 2 already introduced the challenge of this approach and possible solutions were reviewed. In this chapter, the applied techniques as well as their novel integration into the MARL framework will be presented before the training cycle is outlined.

3.1 Baseline Algorithms

In order to evaluate the influence of curiosity on MARL, we consider value-based and policy gradient methods as baselines. *Independent Q-learning* (IQL) [Tan, 1993], briefly introduced in Section 2.2.2.2, is a simple and commonly used value-based algorithm. It aims to learn individual action-value functions for each agent, which are used for action selection. As the policy gradient approach, *multi-agent deep deterministic policy gradient* (MADDPG) [Lowe et al., 2017] will be the evaluated. The multi-agent actor-critic method, referred to in Section 2.2.3.1, reached state-of-the-art performance on our baseline evaluation tasks. One of the main advantages of MADDPG is its flexibility to be applied to competitive and collaborative tasks.

In the following subsections, we will elaborate on our implementations of these baseline algorithms. Let d_i^o be the dimensionality of a single observation and d_i^a of an action for agent i . d^o and d^a are used to denote the dimensionality of joint observations and actions respectively, hence

$$d^o = \sum_{i=1}^{N} d_i^o \qquad (3.1) \qquad\qquad d^a = \sum_{i=1}^{N} d_i^a \qquad (3.2)$$

where N is the number of agents. Similarly, N_i^a denotes the number of possible

actions for agent i (in discrete action spaces).

3.1.1 Independent Q-Learning

IQL is implemented using *deep Q-networks* (DQNs) [Mnih et al., 2013, 2015]. Each agent maintains its own DQN, computing the action-values $Q_i(o_i, a_i)$ for each action a_i given its input observation o_i. The initial input represents the observation of fixed dimensionality d_i^o and two intermediate layers compute hidden representations of size d^h. Both hidden layers apply the rectified linear activation function ReLU $f(x) = \max(x, 0)$ for non-linearity. The final output layer computes a scalar value estimate for each possible action of the respective agent as shown in Figure 3.1.

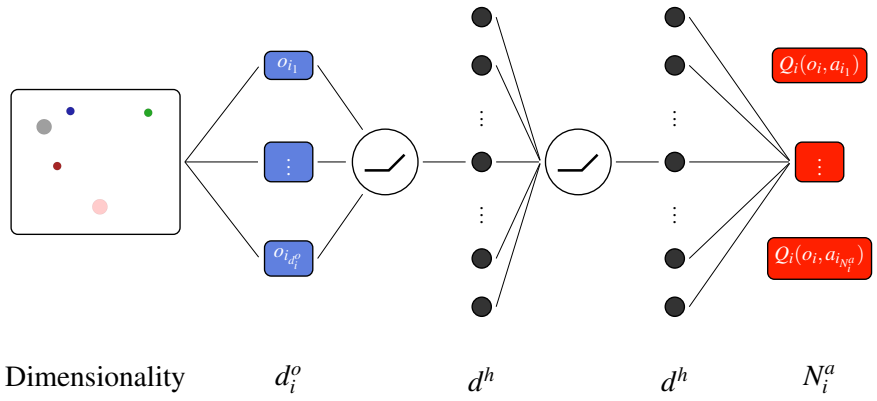

Figure 3.1: Illustration of DQN architecture for agent i with two intermediate layers of hidden representation size d^h and ReLU activation function.

These networks are trained to estimate Q-targets (Equation (3.3)) by minimising the mean-squared error (MSE) over minibatches of experience tuples (o, a, r, o') as shown in Equation (3.4). Each DQN is optimised individually on experience including only the agent's own observations o_i, o_i', action a_i and reward r_i.

$$y_i = r_i + \gamma \max_{a_i'} \bar{Q}_i(o_i', a_i'; \bar{\theta}_i) \tag{3.3}$$

$$\mathcal{L}(\theta_i) = \mathbb{E}_{o_i, a_i, r_i, o_i' \sim \mathcal{D}} \left[\left(Q_i(o_i, a_i; \theta_i) - y_i \right)^2 \right] \tag{3.4}$$

Each target value y_i of agent i is computed using a target network $\bar{Q}_i$ with delayed parameters $\bar{\theta}_i$.

To maintain exploration during training, every agent is using an ε-greedy policy. Such policies are frequently applied and select the greedy action, i.e. the action $a_i =$

$\text{argmax}_a Q_i(o_i, a)$ maximising the value function for the current observation o_i, with probability $1 - \varepsilon$. With probability ε, a random action is selected instead. This way, the agent constantly explores and ε is slowly decayed during training to converge to an optimal policy. It should be noted that such a policy and the architecture of DQN assume a discrete action space.

3.1.2 Multi-Agent Deep Deterministic Policy Gradient

For MADDPG, each agent is training two separate networks. The actor and critic are both defined as deep neural networks with three layers. While the actor policy network receives individual observations of size d_i^o as input, the critic receives joint observations and actions of total dimensionality $d^o + d^a$. As for DQNs, both intermediate representations have dimensionality d^h and the ReLU activation function is applied. The critic output is a single scalar value representing the Q-value of the joint action given the underlying state of the observations for agent i. Both architectures are illustrated in Figures 3.2 and 3.3.

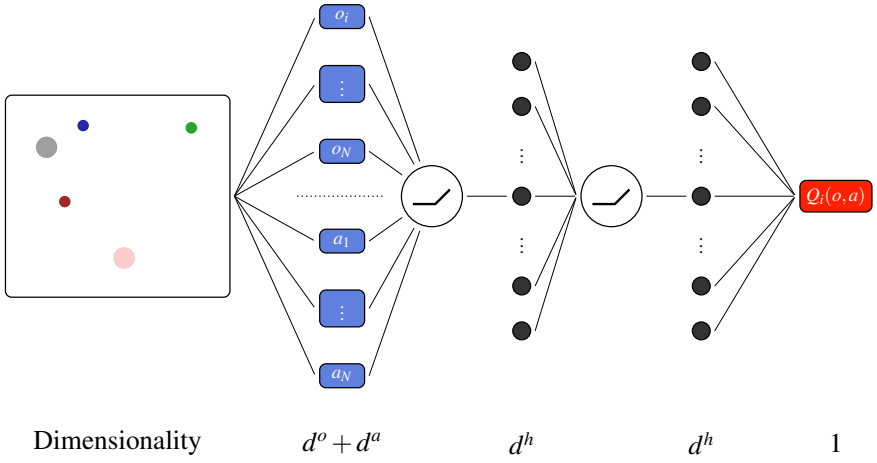

Figure 3.2: Illustration of MADDPG critic architecture for agent i with two intermediate layers of hidden representation size d^h and ReLU activation function. The action-value output is over the joint observations $o = (o_1, ..., o_N)$ and actions $a = (a_1, ..., a_N)$.

Optimisation of the critic is done as for IQL, minimising an MSE loss to estimate Q-targets. However, instead of maximising over Q-values, the target policies $\bar{\pi}$ are used for action selection given the next observations o'. The critic is receiving joint action and observation as its input to compute the respective Q-value for agent i. In the following equations, θ_i^Q and θ_i^π denote the critic and actor parameters of agent i respectively.

$$y_i = r_i + \gamma \, \bar{Q}_i(o'_1, ..., o'_N, a'_1, ..., a'_N; \bar{\theta}_i^Q)\Big|_{a'_k = \pi_k(o'_k; \bar{\theta}_k^\pi)} \tag{3.5}$$

$$\mathcal{L}(\theta_i^Q) = \mathbb{E}_{o,a,r,o' \sim \mathcal{D}} \left[\left(Q_i(o_1, ..., o_N, a_1, ..., a_N; \theta_i^Q) - y_i \right)^2 \right] \tag{3.6}$$

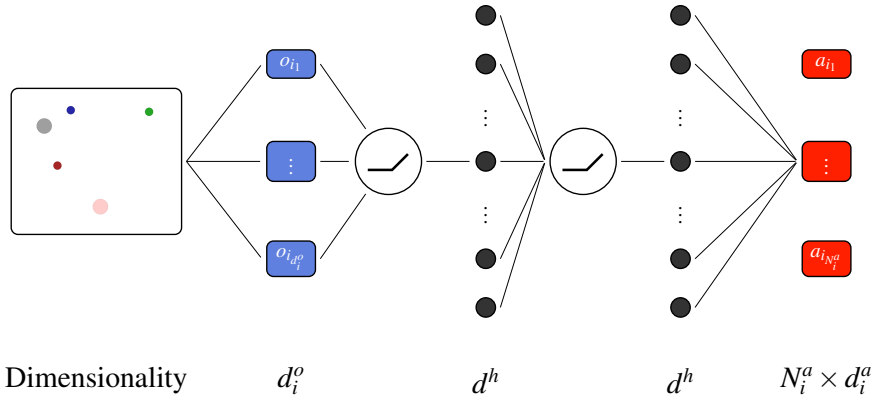

Figure 3.3: Illustration of MADDPG actor architecture for agent i with two intermediate layers of hidden representation size d^h and ReLU activation function.

Actor networks represent policies and hence are computing respective actions of dimensionality d_i^a directly. They are trained to minimise the negative action-value represented by the following loss.

$$\mathcal{L}(\theta_i^\pi) = \mathbb{E}_{o,a,r,o' \sim \mathcal{D}} \left[-Q_i \left(o_1, ..., o_N, a_1, ..., \hat{a}_i, ..., a_N; \theta_i^Q \right) \right] \Big|_{\hat{a}_i = \pi_i(o_i; \theta_i^\pi)} \tag{3.7}$$

Exploration for continuous action spaces uses decaying Gaussian noise, which is added to the policy output as suggested by Lowe et al. [2017]. In the case of discrete actions, a gumbel-softmax distribution [Jang et al., 2016] is applied to obtain a differentiable policy for optimisation. The implementation is based on the open-source project of Shariq Iqbal[1].

3.2 Curiosity Approaches

In Section 2.3, the idea of intrinsic rewards as exploration incentives has been presented. This work extends the MARL baselines with a general framework to include such intrinsic rewards and implements three different approaches for evaluation.

[1]The MADDPG open-source implementation using PyTorch is available under `https://github.com/shariqiqbal2810/maddpg-pytorch`.

Each curiosity approach computes a reward signal r^I based on experience samples (o, a, r^E, o') where r^E denotes the usual, extrinsic reward provided by the environment. Whenever curiosity is applied, the reward signal used during training of DQNs or MADDPG critics is defined as the following sum

$$r = r^E + \eta\, r^I \tag{3.8}$$

where η denotes a weighting factor. In the absence of intrinsic rewards, the reward is simply the extrinsic reward $r = r^E$.

3.2.1 Hash-based counting

The first considered exploration bonus is the count-based curiosity of Tang et al. [2017], referred to in Section 2.3.1. Our implementation is largely based on existing open-source code[2] and follows the proposed approach using the SimHash function [Charikar, 2002]. A projection matrix $A \in \mathbb{R}^{d^o \times d^k}$ for observation dimension d^o and key dimensionality d^k is randomly initialised from a standard normal distribution $\mathcal{N}(0,1)$. A is used to compute observation representations as follows

$$\phi(o) = sgn(A\ o) \tag{3.9}$$

where *sgn* denotes the sign function. During training, each encountered observation is encoded using ϕ and its representation counter $n(\phi(o))$ incremented. The intrinsic reward is defined as

$$r^I = \frac{1}{\max\left(1, \delta \cdot \sqrt{n(\phi(o))}\right)} \tag{3.10}$$

where δ denotes a decay factor.

3.2.2 Intrinsic Curiosity Module

As one of two prediction-based curiosity approaches, we implement the intrinsic curiosity module (ICM) [Pathak et al., 2017] as presented in Section 2.3.2.1. The feature representation $\phi(o)$ of observation o is computed using a feedforward neural network with three layers. Intermediate layers compute a representation of hidden curiosity size d^{h_c} and apply the ReLU activation function. The final representation has chosen dimensionality d^ϕ.

[2]The open-source implementation of hash-based counting as an exploration bonus is available under `https://github.com/openai/EPG/blob/master/epg/exploration.py`.

The inverse model, as depicted in Figure 2.4, predicts the applied action a_t given representations $\phi(o_t)$ and $\phi(o_{t+1})$. The initial layer computes a hidden representation of size d^{h_c}, applies ReLU for nonlinearity and the last layer computes the action representation.

The forward model predicts $\phi(o_{t+1})$ given $\phi(o_t)$ and a_t. It similarly consists of two layers, where the first computes a d^{h_c}-sized representation and applies the ReLU function. The last layer simply computes a representation of size d^{ϕ}.

Let θ^{ϕ} denote the parameters of the representation model while θ^I and θ^F are the parameters of inverse and forward model respectively. The forward model is trained to minimise the following loss of its predicted representation

$$\mathcal{L}(\theta^{\phi}, \theta^F) = \frac{1}{2}\mathbb{E}_{o,a,r,o' \sim \mathcal{D}} \left[\left(\phi(o'; \theta_i^{\phi}) - \hat{\phi}(o'; \theta^F) \right)^2 \right] \tag{3.11}$$

where $\hat{\phi}(o'; \theta_F) = f(\phi(o; \theta^{\phi}), a; \theta^F)$ is the predicted representation of the forward model.

Training of the inverse model depends on the action space. For discrete actions, a cross-entropy loss is minimised

$$\mathcal{L}(\theta^{\phi}, \theta^I) = \mathbb{E}_{o,a,r,o' \sim \mathcal{D}} \left[-log \left(\frac{exp(\hat{a}[a])}{\sum_j exp(\hat{a}[j])} \right) \right] \tag{3.12}$$

where $\hat{a} = i(\phi(o; \theta^{\phi}), \phi(o'; \theta^{\phi}); \theta^I)$ is the prediction of the applied action by the inverse model and a represents the truly applied, discrete action. For continuous actions, a MSE loss of $\hat{a}$ and a is minimised instead.

3.2.3 Random Network Distillation

Random network distillation (RND) [Burda et al., 2018b] is the last applied form of curiosity in this project. Its networks are essentially identical to the state representation model $\phi(o; \theta^{\phi})$ of ICM with three layers computing d^{h_c} intermediate representations with ReLU and output size d^{ϕ}. As already explained in Section 2.3.2.2, a fixed, target network $\bar{\phi}(o; \theta^{\bar{\phi}})$ is initialised with equal architecture. However, its parameters are fixed and the representation model is trained to mimic its outputs. This is achieved by minimising a MSE loss.

$$\mathcal{L}(\theta^{\phi}) = \mathbb{E}_{o \sim \mathcal{D}} \left[\left(\phi(o; \theta^{\phi}) - \bar{\phi}(o; \theta^{\bar{\phi}}) \right)^2 \right] \tag{3.13}$$

3.2.4 Independent and Joint Curiosity

All curiosity approaches will always be used to compute an intrinsic reward as shown in Equation (3.8). However, in multi-agent systems, there are two major ways how

such an exploration bonus r^I can be computed. Either, each agent individually calculates its own exploration bonus or a single curiosity model is constructed for all agents to use jointly. In the first case, each agent maintains its own curiosity prediction network or count-table respectively. This is used to compute the intrinsic reward r_i^I using its observation o_i (and in the case of ICM its action a_i and next observation o_i'). If curiosity is applied jointly, then a single curiosity model is constructed to compute exploration bonuses based on the joint observations o, o' and actions a.

It should be noted that especially for prediction-based curiosity approaches, such joint application appears preferable. Predicting the dynamics of the environment, as especially ICM aims to learn, appears very challenging solely based on individual observations and actions. In most MARL tasks, not just the individual action, but also actions of other agents influence the environmental state and impact obtained observations.

In this project, both approaches are considered and their respective impact on MARL will be evaluated.

3.3 Training

The detailed training cycle for MARL with curiosity is illustrated in Algorithm 1 below. It uses an episodic replay memory $\mathcal{D}$ of capacity $N_{\mathcal{D}}$ to store and sample minibatches for efficient training.

For generalisation, we simply annotate the MARL algorithm parameters of each agent with θ_i and the parameters of its applied curiosity approach with θ_i^C. Note that for IQL, θ_i represents the parameters of agent i's DQN, while for MADDPG $\theta_i = \{\theta_i^Q, \theta_i^\pi\}$ it represents the critic and actor parameters. Similarly, the curiosity parameters θ_i^C represent A_i, $\{\theta_i^\phi, \theta_i^F, \theta_i^I\}$ or $\{\theta_i^\phi, \theta_i^{\bar{\phi}}\}$ respectively. If curiosity is applied jointly, then $\theta_i^C = \theta^C$ for each agent $i = 1, ..., N$.

After the initialisation, N_{episodes} episodes are executed. Each episode ends either when a terminal state (done) or a maximum number of steps per episode, $N_{\text{max-episode-length}}$, is reached. During each episode, the agents repeatedly select actions $a = (a_1, ..., a_N)$ based on their policies π_i given their current, individual observation o_i. The applied policies π_i are ε-greedy policies for IQL with respect to the learned Q-function and the actor policies for MADDPG agents. The observed rewards r, terminal signal done and next observation o' are stored together with o as an experience tuple in $\mathcal{D}$.

Every $N_{\text{update-freq}}$ steps, parameters are optimised on a randomly sampled minibatch

Algorithm 1 Training Cycle for MARL with Curiosity

Initialise environment
Initialise replay memory $\mathcal{D}$ with capacity $N_{\mathcal{D}}$
Initialise MARL algorithm networks with random parameters θ_i
Initialise curiosity model with random parameters θ_i^C

$t \leftarrow 0$
for $episode = 1, ..., N_{\text{episodes}}$ **do**
    $o \leftarrow$ reset environment and get initial observation
    $done \leftarrow False$
    $episode\_length \leftarrow 0$
    while not $done$ and $episode\_length < N_{\text{max-episode-length}}$ **do**
        $a \leftarrow$ Select actions according to agent policies π_i for o_i
        r, $done$, $o' \leftarrow$ Execute a and observe r, $done$ and o'
        Store (o, a, r, o') in $\mathcal{D}$
        if $t \mod N_{\text{update-freq}} = 0$ and $|\mathcal{D}| \geq N_{\text{batch-size}}$ **then**
            $(o^B, a^B, r^B, o'^B) \leftarrow$ Sample batch of $N_{\text{batch-size}}$ experience steps from $\mathcal{D}$
            TRAIN(o^B, a^B, r^B, o'^B) ▷ Train networks
        end if
        $t \leftarrow t + 1$
        $episode\_length \leftarrow episode\_length + 1$
        $o \leftarrow o'$
    end while
end for

function TRAIN(o, a, r, o')
    for agent $i = 1, ..., N$ **do**
        Update parameters θ_i with gradient descent step minimising $\mathcal{L}(\theta_i)$
        Update parameters θ_i^C with gradient descent step minimising $\mathcal{L}(\theta_i^C)$
        $\bar{\theta}_i \leftarrow (1-\tau)\bar{\theta}_i + \tau\theta_i$ ▷ Soft target network updates
    end for
end function

of size $N_{\text{batch-size}}$ from $\mathcal{D}$. Optimisation of the curiosity and MARL algorithm parameters with respect to their losses is achieved as described in the previous sections. Each parameter optimisation is executed using the Adam gradient descent optimiser [Kingma and Ba, 2014]. For the MARL algorithm parameters θ_i, batch normalisation [Ioffe and Szegedy, 2015] is additionally used on each input minibatch to stabilise training. After parameter optimisation for each agent, we always execute a soft update of the target networks with delayed parameters $\bar{\theta}_i$ using step-size τ.

3.3.1 Curiosity Weighting and Decay

Lastly, it should be noted that the weighting of intrinsic rewards with η as well as their learning rate for optimisation of RND and ICM are crucial parameters for our approach. It is challenging to choose these parameters, because they are highly task-specific. If the intrinsic rewards are too large, then the agent is solely incentivised to explore without learning the intended behaviour. Otherwise, if η is chosen too small for the task, then the curiosity will have no impact at all. Likewise, a large learning rate determines the decay of curiosity which can cause exploration to become overwhelming or ineffective. These challenges will be addressed as part of our evaluation and proposed future work in Section 5.1.

Chapter 4

Evaluation

In order to evaluate the impact of curiosity on MARL, we conduct experiments using the *multi-agent particle environment* (MAPE) proposed by Lowe et al. [2017]. It includes collaborative and competitive tasks. We apply our described baseline algorithms as well as agents served with additional curiosity following the methods of Section 3.2.

In addition to evaluating these approaches on the plain tasks of the environment, the impact of partial observability and sparse rewards will be evaluated.

4.1 Experimental Setup

4.1.1 Multi-Agent Particle Environment

The MAPE contains multiple tasks for MARL. It is built on top of the RL environment toolkit Gym [Brockman et al., 2016] and is publicly available[1].

All tasks involve particles and landmarks in a continuous two-dimensional environment. Observations are provided as high-level features rather than visual frames and agents are continuously receiving informative reward signals. The action space for all tasks and agents are discrete, such that IQL can be employed. In the following section, all applied tasks will be summarised.

4.1.1.1 Cooperative communication

The cooperative communication task requires the collaboration of two agents to identify a goal among three possible landmarks and move towards it. While one agent is

[1] Environment available under `https://github.com/openai/multiagent-particle-envs`

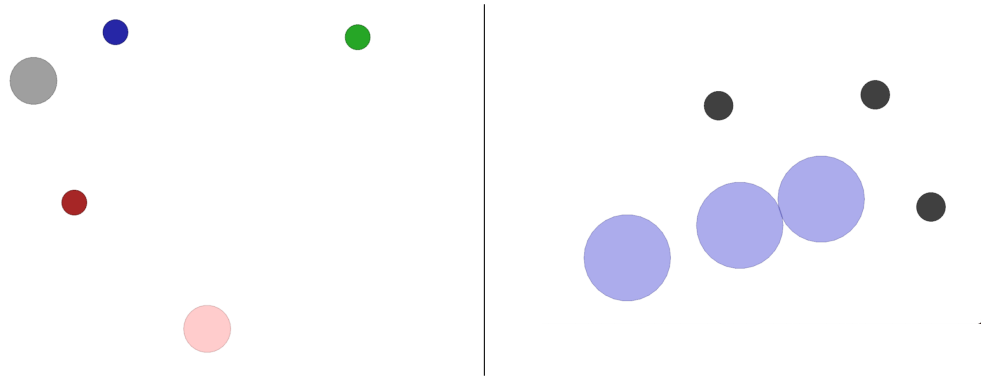

Figure 4.1: Rendering of cooperative communication (left) and cooperative navigation (right) environments. (Left) The speaker agent (grey) communicates the goal landmark colour to the moving listener agent (larger red entity). (Right) The agents (blue) have to move to the landmarks (black) while avoiding collision.

unable to move but receives information about the goal, the listening agent has to reach the communicated goal landmark. Figure 4.1 (left) visualises the task.

Observation space The listener agent receives its velocity, relative landmark positions and the communication of the speaking agent as observations. Meanwhile, the speaker agent only observes the colour of the goal landmark as three numeric values.

Action space The listener agent has the usual action space for the MAPE including five discrete actions corresponding to no movement as well as four directional movements (right, left, up, down). In the cooperative communication task, the speaker agent chooses between three discrete actions to communicate the goal landmark to the mobile agent.

Reward Both agents receive the same reward representing the negative squared Euclidean distance of the moving agent towards the goal landmark.

4.1.1.2 Cooperative Navigation

The cooperative navigation task involves three agents moving to landmarks while avoiding collisions with each other. The environment is shown in Figure 4.1 (right).

Observation space All agents receive their current velocity, position, relative landmark and agent positions as their input. Additionally, the observations also in-

clude communications. However, agents are not allowed to communicate during this task.

Action space The discrete action space for each agent involves the same five movement actions used by the listening agent in the collaborative communication task: standing still, moving right, left, up and down.

Reward Each agent receives the same reward signal. For each landmark, the negative minimum distance to any agent is computed and summed up. Additionally, any collisions of agents are punished with a negative reward of -1.

4.1.1.3 Physical deception

The physical deception task is one of the competitive tasks. Two collaborating agents have to move towards a marked goal landmark while avoiding another adversary agent from recognising and reaching it as well. This has to be achieved by moving towards both existing landmarks, such that the adversary agent is unable to recognise which landmark is the true goal location. An illustration can be found in Figure 4.2 (left).

Observation space The cooperative agents receive their relative goal position, relative position to other landmarks and agents. While relative landmark and agent positions are also provided to the adversary agent, it gets no information about the goal landmark.

Action space The action space for each agent again involves four discrete directional movement actions and an action to stand still.

Reward Cooperative agents receive rewards based on their negative minimum distance to the goal. Additionally, the reward includes a positive term for the distance of the adversary to the goal landmark to punish agents for letting the adversary close to the landmark. Meanwhile, the adversary receives its negative goal distance as rewards.

4.1.1.4 Predator-Prey

The last task is also of competitive nature. In the predator-prey environment, three adversary agents aim to catch an individual, faster agent. Two obstacle landmarks are placed in the environment for both teams to exploit, as seen in Figure 4.2 (right).

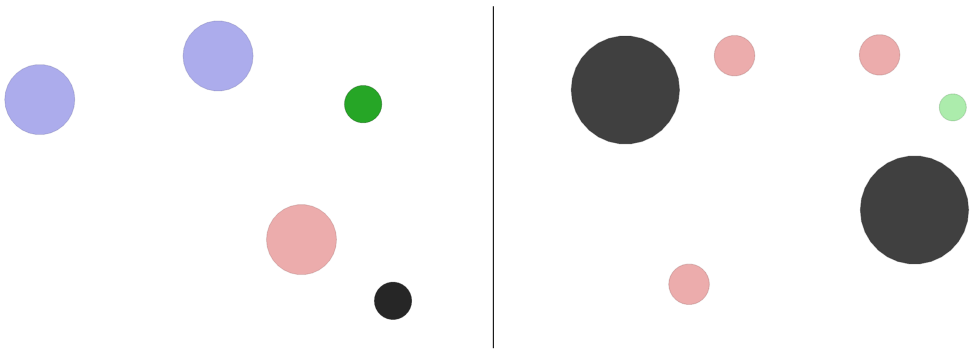

Figure 4.2: Rendering of physical deception (left) and predator-prey (right) environments. (Left) The collaborating agents (blue) have to move to the goal landmark (green) while deceiving the adversary agent (red). (Right) The adversary agents (red) aim to catch the faster fleeing agent (green).

Observation space The fleeing agent receives its velocity, position and relative landmark and agent positions. The adversary agents additionally get information about the velocity of the fleeing green agent.

Action space All agents decide among the identical five discrete movement actions as for previous tasks.

Reward The fleeing agent is punished for any collisions with adversary agents and leaving the visualised area of the screen (to force it to stay in this part of the environment). Adversary agents on the other side are collectively rewarded if they collide with the individual agent.

4.1.2 Modified Tasks

4.1.2.1 Partial Observability

Curiosity has the primary objective of incentivising targeted exploration in novel parts of the environment. Such exploration is primarily required in tasks which restrict the observation and hence knowledge of agents about their environment. Therefore, we implement modified versions of all presented tasks involving partial observability. This is achieved by limiting agents' field of view. They only receive landmark and agent positions as well as velocities for entities that are in a predefined radius of $d_{\text{partial-obs}}$ around their location. Obscured observation parts are replaced with 0-entries to maintain dimensionality. The only exception is the speaking agent of the cooperative communication. It always receives the goal colour as its input due to its inability

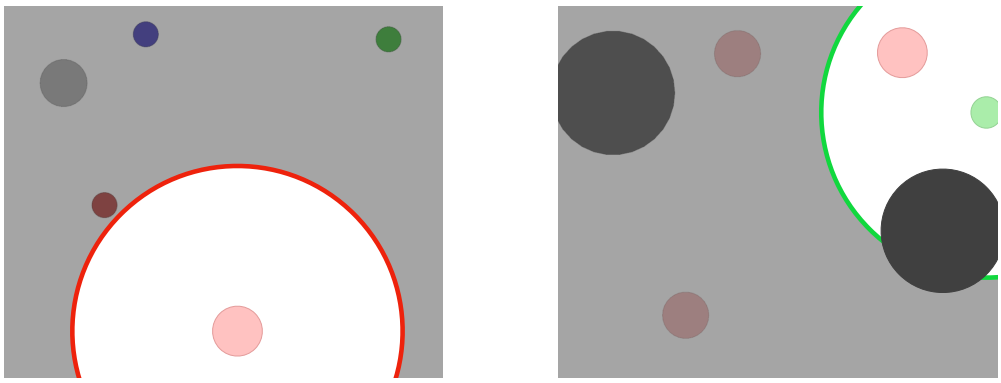

Figure 4.3: Rendering of partially observable cooperative communication (left) and predator-prey (right) task indicating the viewing field of the listener (red) and fleeing (green) agent respectively for $d_{\text{partial-obs}} = 0.5$.

to move.

4.1.2.2 Sparse Rewards

Besides partial observability, we also expect curiosity to impact performance in tasks with only sparse rewards. Such feedback is often insufficient to learn intended behaviour or considerably slows down training. Hence, we conduct experiments for each task with sparse rewards. When sparse rewards are applied, agents only receive non-zero rewards every $N_{\text{sparse-freq}}$ steps. The provided rewards are computed as the average extrinsic feedback agents would have received during these iterations.

4.1.3 Configurations

All baseline algorithms are evaluated on each task and their partially observable counterpart with and without the three curiosity approaches. Additionally, we employ all configurations using sparse rewards. Every definition of curiosity will be applied jointly as well as independently for all agents.

Training is executed over $25,000$ episodes with (up to) 25 steps each. Every 100 steps, models are optimised and five evaluation episodes are executed to track average rewards obtained without exploration. For partial observability, we use a visibility radius of $d_{\text{partial-obs}} = 0.5$ (as visualised in Figure 4.3). Sparse rewards are providing informative feedback every $N_{\text{sparse-freq}} = 25$ steps, such that agents receive such rewards once every episode. Each configuration is run for every task on three different random seeds to obtain representative results. The full list of parameters chosen for

the evaluation can be found in Appendix B.

4.1.4 Hardware and Software Setup

Evaluation is conducted using the Google Cloud Platform [2] for its Compute Engine service. Each machine used for evaluation is equipped with four virtual CPUs, 26GB of system memory and a NVIDIA Tesla K80 GPU with 12GB GDDR5 memory used for training.

4.2 Results

In this section, we will present the results of the evaluation and analyse the findings. Figures indicate the mean performance over the three runs executed for each configuration with shaded areas corresponding to one standard deviation. Reported rewards for each episode represent the reward accumulated over the executed 25 steps.

4.2.1 Baseline Results

Among all tasks, we find that the two baselines perform similarly for all tasks with slight advantages for MADDPG on the cooperative communication and navigation as well as the predator-prey task compared to IQL. The episode rewards are illustrated in Figure 4.4.

However, closer observation of the behaviour of the agents shows noticeable differences. As already reported by Lowe et al. [2017], IQL is unable to learn the intended speaker communication on the cooperative communication task. Hence the listener agent just learns to move in the middle of all landmarks in the environment. While this consistently reaches comparably good rewards, only in MADDPG do the agents reliably learn the intended communication and movement patterns.

4.2.2 Curiosity Results

Similar results can be obtained using different curiosity variations. For the usual particle environment tasks, there is little discrepancy to be observed between joint and individual curiosity for each of the approaches. This can be seen for each of the tasks. Figure 4.5 shows the rewards for MADDPG and IQL using the varying individual and

[2] The service is available at `https://cloud.google.com`.

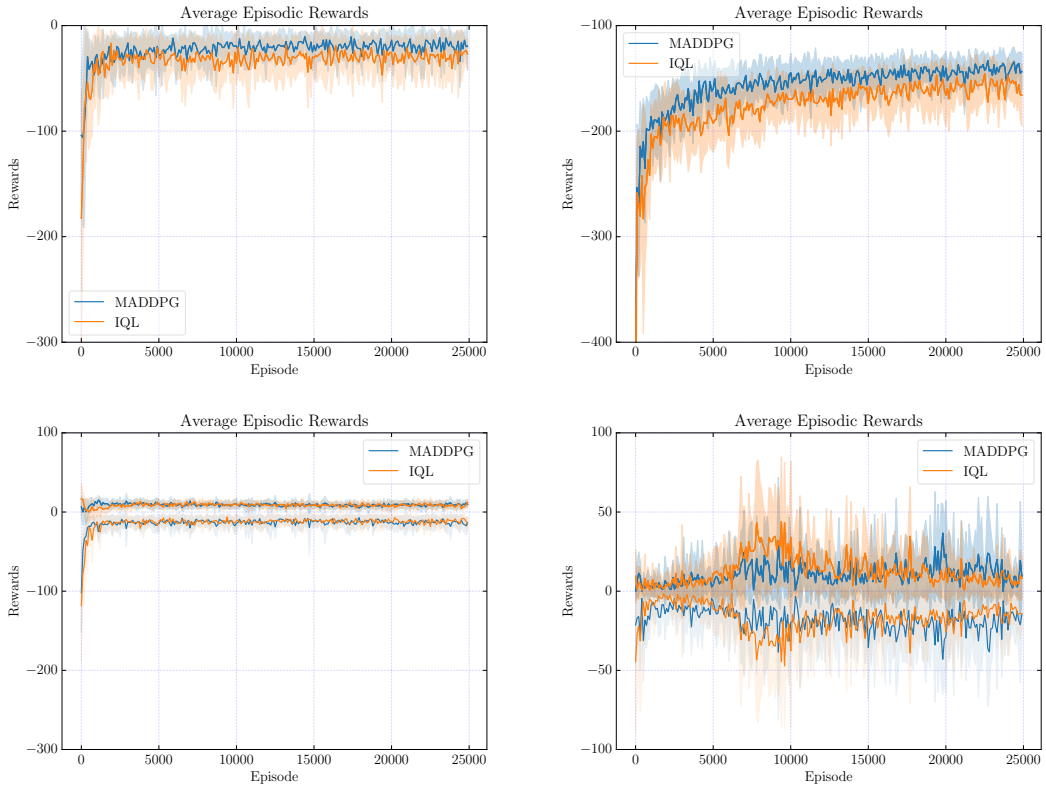

Figure 4.4: Episodic reward of baselines MADDPG (blue) and IQL (orange) for cooperative communication (top left), cooperative navigation (top right), physical deception (bottom left) and predator-prey task.

joint curiosities at the example of the cooperative communication task as well as the underlying intrinsic rewards.

Generally, curiosity input seems to disrupt training. However, such disturbance appears less influential for IQL agents, where there is almost no difference depending on the curiosity approach or its application mode. Presumably, this is the case due to the independent approach of IQL, where each agent individually computes its Q-function without considering other agents. While additional intrinsic rewards might affect individual agents, for MADDPG the joint critic and actor network are both trained concurrently and depend on each other. Hence, such approaches appear more sensitive to changes in the reward signal. Furthermore, we can observe that joint curiosity shows more stable performance than the decentralised curiosity application for MADDPG. This effect can be explained by inspecting the difference in computed intrinsic reward for the varying approaches and application as illustrated in the third column of Figure 4.5. Joint intrinsic rewards for all agents lead to reduced variance of the intrinsic

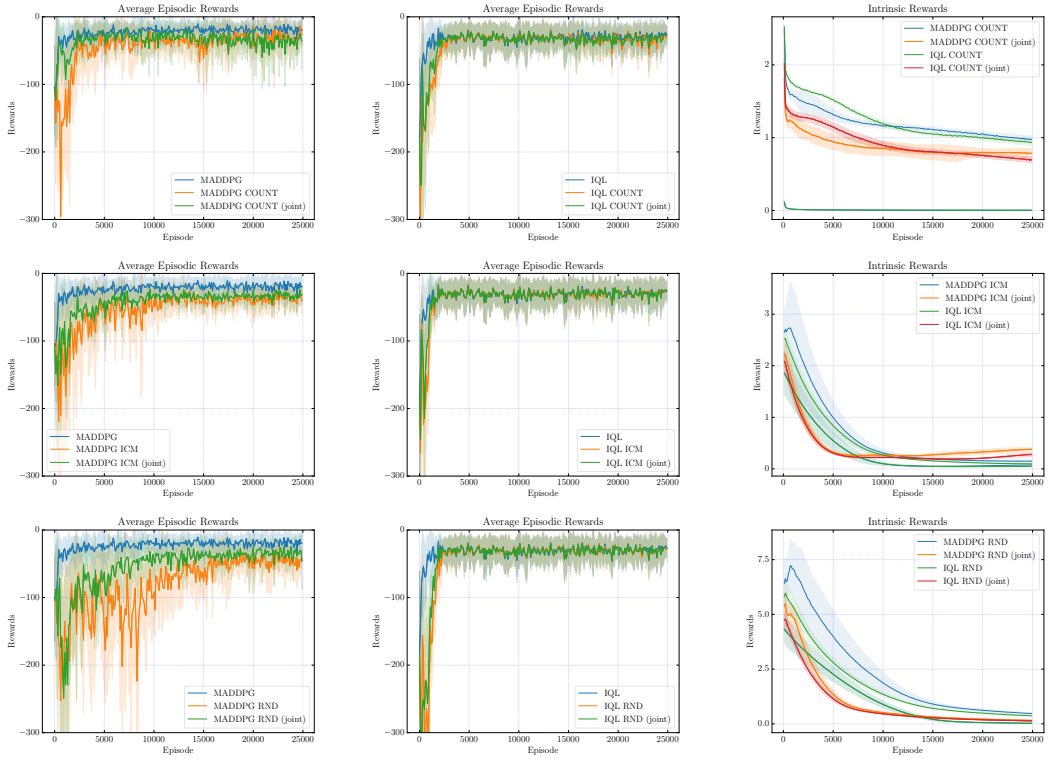

Figure 4.5: Episodic reward of MADDPG (left column) and IQL (middle column) with individual and joint count (1st row), ICM (2nd row) and RND (3rd row) curiosity on the cooperative communication task. The right column shows intrinsic rewards for MADDPG and IQL with the respective curiosity applied individually and jointly.

signal. Similarly, we notice that the intrinsic rewards decay slightly faster for joint curiosity. Those effects are to be expected. For joint curiosity, a single curiosity model is trained using samples from every agent instead of training separate models, which all require optimisation.

Lastly, it can be observed that the intrinsic rewards computed by the hash-based counting largely vary from agent to agent. This can be seen in the top right figure for individual curiosities, but only appears on the cooperative communication task due to the varying observation sizes as the speaker agent receives low-dimensional inputs. The smaller the observations, the higher is the likelihood of "hits" in the hash table, which corresponds to a quicker decay in reward.

Illustrations of the episodic rewards for each task and curiosity configuration can be found in Appendix C, Figures C.1 to C.4. Similarly, Figure C.21 shows the development of intrinsic rewards for count-based, ICM and RND curiosity applied individually as well as jointly for IQL and MADDPG on all considered tasks.

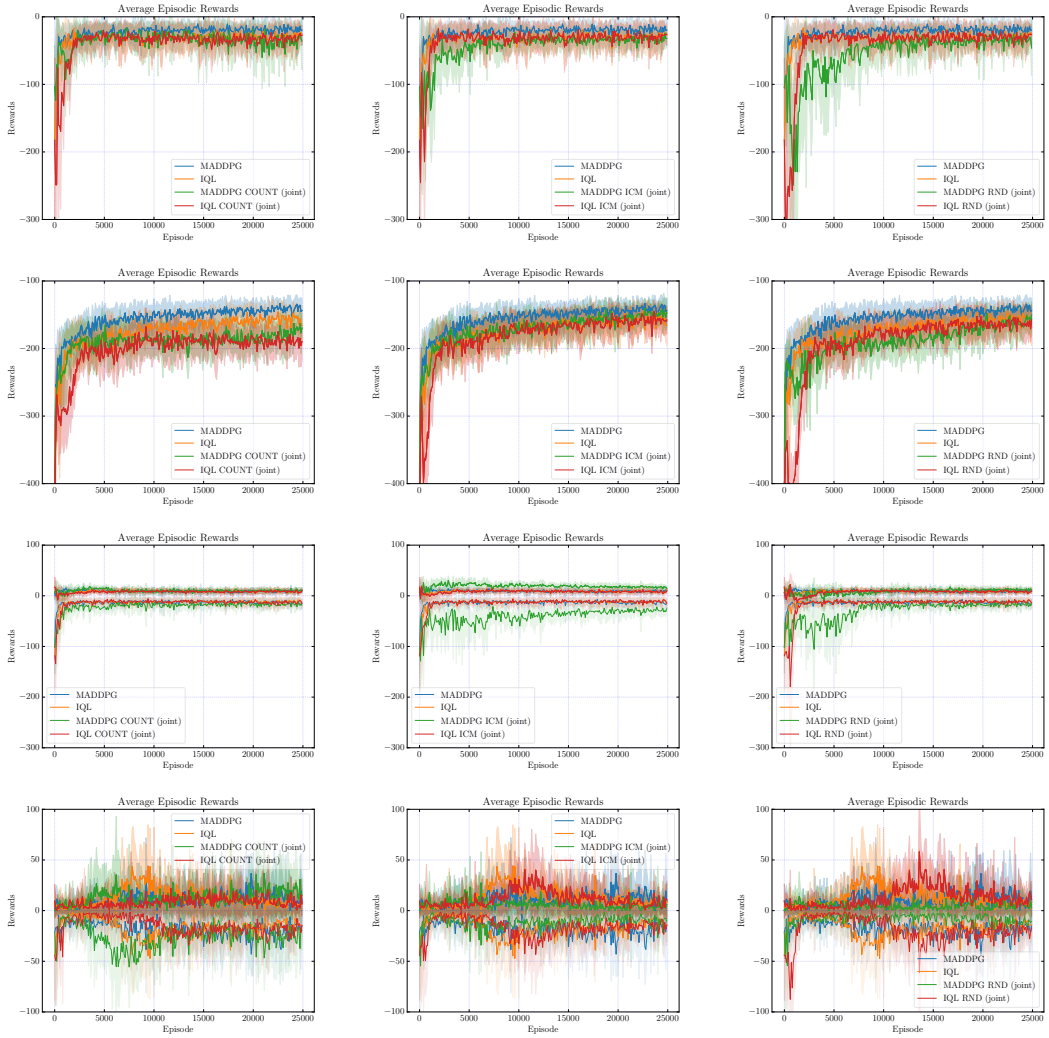

Figure 4.6: Episodic reward of baselines MADDPG and IQL compared with count, ICM and RND curiosity for cooperative communication (1st row), cooperative navigation (2nd row), physical deception (3rd row) and predator-prey (4th row) task.

Figure 4.6 compares joint rewards obtained for all four tasks and curiosities applied with the plain baseline performances. While the difference with and without curiosity applied is subtle on most tasks, we observe that additional curiosities slightly harm performance on the cooperative communication, cooperative navigation and physical deception tasks. Also, we can again observe that the impact of curiosity on MADDPG is considerable, while it hardly affects performance for IQL. For MADDPG, performance on mentioned tasks slightly decays with curiosity.

For the predator-prey task, we observe an increase in stability of the reward signal for MADDPG applied with the ICM and RND curiosities. This is caused by the increased success of the fleeing agent escaping its adversaries. While it might appear as

a positive outcome at first, it seems that it is primarily caused by the chasing agents being unable to learn the appropriate behaviour when curiosity is applied.

4.2.3 Partial Observability Results

As expected, introducing partial observability leads to significantly increased variance in performance. Due to imperfect information depending on agent and landmark locations, initial positions of all entities have considerable influence on the problem difficulty. If agents are initialised without seeing most or even any landmarks and agents, then progress requires challenging exploration.

The effect of increased variance becomes especially visible when directly comparing the baseline algorithms with and without partial observability. Figure 4.7 shows the episodic rewards obtained by MADDPG and IQL on the original tasks as well as under partial observability. Additionally, the performance with joint ICM curiosity is highlighted as an exemplary curiosity.

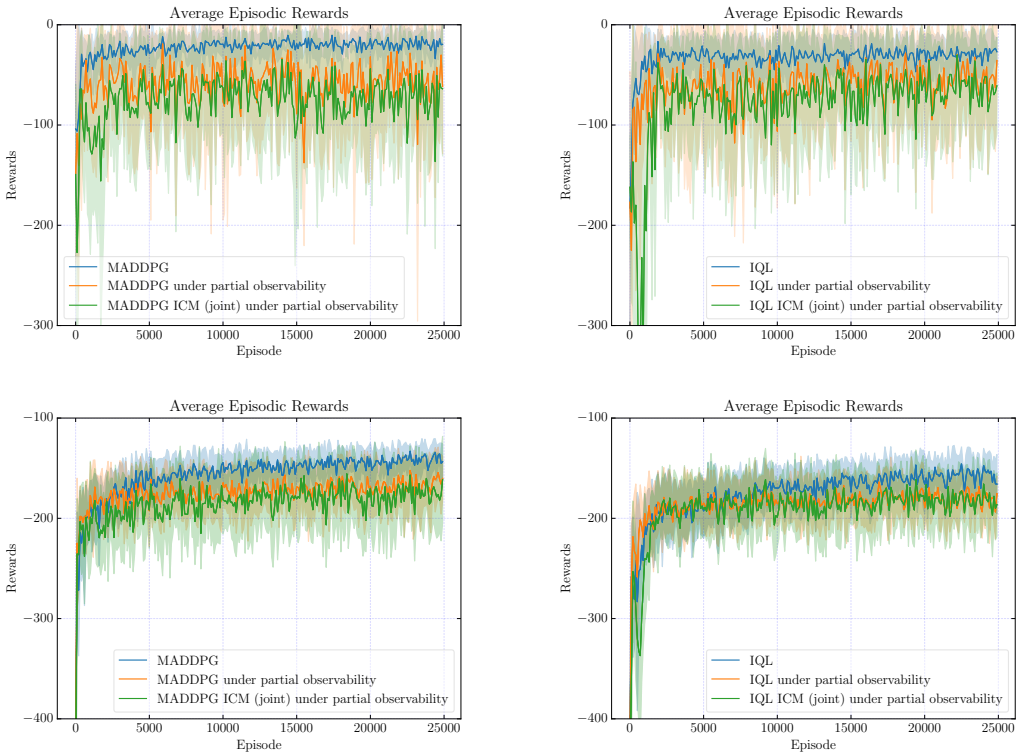

Figure 4.7: Episodic rewards for MADDPG (left) and IQL (right) with joint ICM curiosity for cooperative communication (top) and cooperative navigation (bottom) task with and without partial observability.

The discrepancy between full and partial observability is especially noticeable for the tasks cooperative communication and physical deception. While training on the predator-prey task, presumably due to its competitive nature with multiple adversaries, is generally variant, partial observability hardly affects agents in the cooperative navigation task. This is the case because agents can receive decent rewards through staying stagnant and only moving to landmarks in their visible surrounding, which is the observed behaviour.

For the predator-prey task, we observe that the rewards remain around 0 indicating successful escape of the fleeing agent (which is only punished for being caught or leaving the frame). This is unsurprising in partial observability, where the chasing adversaries have a harder task in spotting and successfully chasing the agent. Surprisingly, this primarily occurs when curiosity is applied.

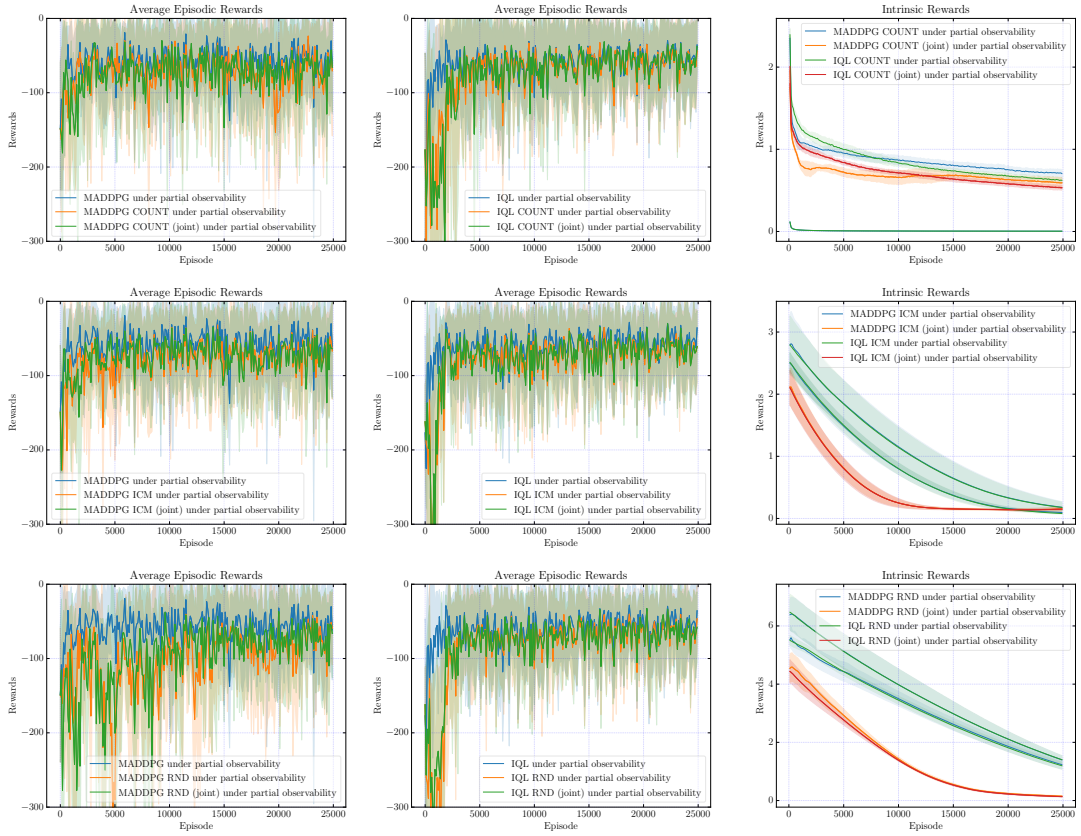

Figure 4.8: Episodic reward of MADDPG (left) and IQL (right) with individual and joint count (top), ICM (middle) and RND (bottom) curiosity on the cooperative communication task under partial observability. The right images show intrinsic rewards for MADDPG and IQL with the respective curiosity applied individually and jointly.

Unfortunately, we observe that curiosity is unable to alleviate the introduced vari-

ance or assist guided exploration as intended. Neither joint nor decentralised curiosity application reliably assist training. Figure 4.8 shows the episodic and intrinsic rewards obtained when applying multiple curiosity configurations on the cooperative communication task under partial observability.

All evaluation results under partial observability, including intrinsic rewards, can be found in Figures C.5 to C.12 and Figure C.22.

4.2.4 Sparse Rewards Results

While curiosity is unable to assist training under partial observability, a significant improvement can be observed in the sparse rewards setting for MADDPG (Figure 4.9).

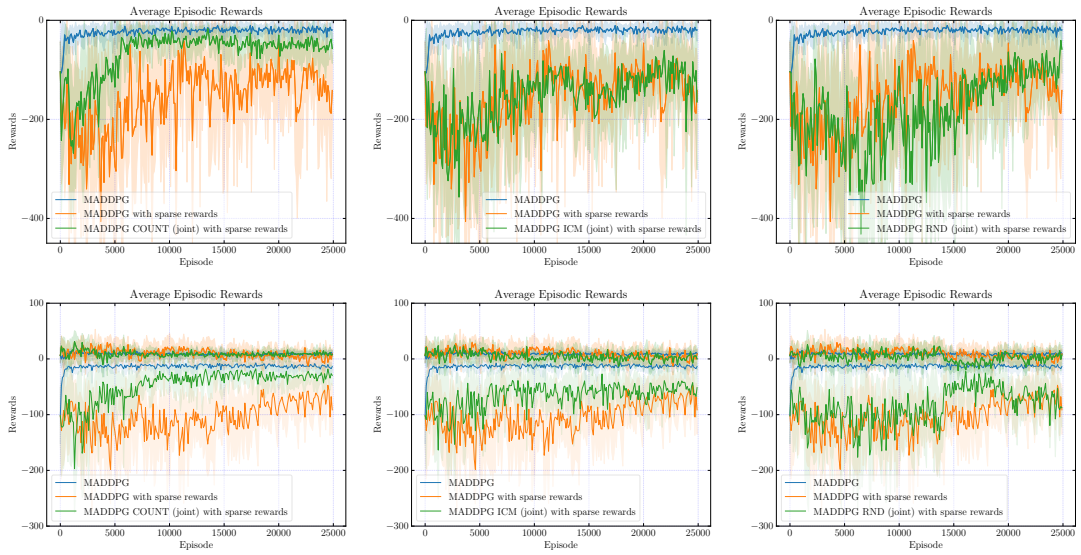

Figure 4.9: Episodic rewards for MADDPG with joint curiosities for cooperative communication (top) and physical deception (bottom) tasks with and without sparse rewards.

Without any curiosity signal, MADDPG agents are unable to solve any of the tasks when only trained with sparse reward signals. Agents either remain still or randomly move around without any apparently learned behaviour. However, once curiosity signals are provided during training, MADDPG agents learn strong policies. Baseline MADDPG trained with continuous rewards still outperforms any curiosity configuration trained with sparse rewards, but considerable improvement can be found in some tasks.

In the cooperative communication task, curiosity trained speakers are not reliable, but sometimes able to learn the correct communication policy. However, the listener agents appear to ignore such communication and behave similarly than the agents in

IQL by staying close to the middle of all landmarks. For the competitive predator-prey and physical deception tasks, considerable learning in fleeing, chasing behaviour as well as moving towards landmarks respectively can be found. On the other hand, no guided behaviour can be observed on the cooperative navigation task. This holds for any configuration trained with sparse rewards.

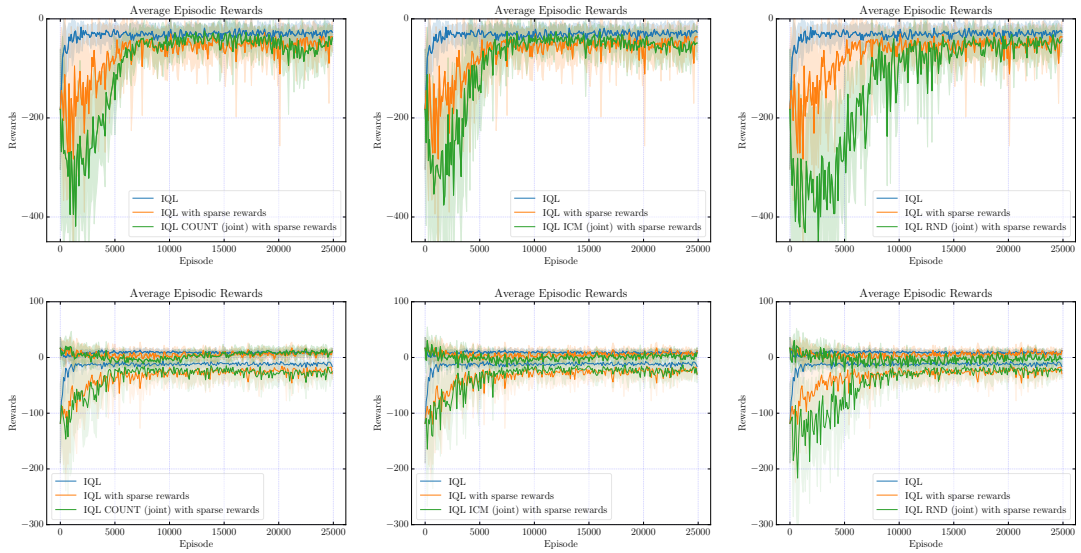

Figure 4.10: Episodic rewards for IQL with joint curiosities for cooperative communication (top) and physical deception (bottom) tasks with and without sparse rewards.

Similar to the observed results under partial observability, we find that IQL is more invariant to the introduced challenge of sparse rewards and provided curiosity. Generally, IQL agents reach comparable performance when trained with and without curiosity when trained with sparse rewards. Additionally, performance is only slightly affected by the additional challenge of sparse rewards. The primary difference in IQL with sparse rewards is the number of episodes it takes for the training to converge. Without continuous rewards, it takes longer for such convergence. However, the performance after converging to stable behaviour is similar to the one of IQL trained with constant rewards.

4.2.5 Time Overhead

When considering curiosity, the additional cost of application must not exceed its use. Therefore, we report the additional time required to train with such intrinsic rewards. Table 4.1 shows the training time of all configurations on the original tasks. It should be noted that sparse rewards do not impact training time (noticeably). However, the

introduction of partial observability increases computation time. This is the case due to a slight overhead in the environment, which has to compute distances for agent observations in each iteration. The difference can be observed for all tasks and configuration in Tables C.1 to C.3.

Training time (in s)	MADDPG	IQL
Baseline	1181.35 ± 6.70	1061.44 ± 7.83
COUNT	1236.65 ± 20.41	1131.53 ± 11.71
COUNT (joint)	1236.98 ± 7.39	1156.89 ± 4.51
ICM	1307.24 ± 2.94	1196.75 ± 10.97
ICM (joint)	1311.74 ± 4.32	1207.72 ± 5.70
RND	1251.21 ± 14.22	1150.68 ± 4.96
RND (joint)	1236.57 ± 4.09	1151.78 ± 9.52

Training time (in s)	MADDPG	IQL
Baseline	2035.47 ± 4.13	1909.75 ± 22.34
COUNT	2172.93 ± 21.78	1964.71 ± 16.52
COUNT (joint)	2179.26 ± 20.34	1963.43 ± 19.51
ICM	2251.91 ± 29.74	2073.12 ± 37.87
ICM (joint)	2267.04 ± 23.62	2100.06 ± 16.47
RND	2160.37 ± 20.87	2002.03 ± 26.07
RND (joint)	2172.50 ± 22.52	1986.23 ± 32.44

Training time (in s)	MADDPG	IQL
Baseline	1796.15 ± 18.79	1602.39 ± 12.84
COUNT	1881.22 ± 20.89	1718.93 ± 8.23
COUNT (joint)	1874.70 ± 9.69	1739.01 ± 8.75
ICM	2004.54 ± 20.41	1835.24 ± 8.21
ICM (joint)	1988.79 ± 13.43	1805.95 ± 10.15
RND	1880.30 ± 4.01	1742.24 ± 18.48
RND (joint)	1878.81 ± 15.08	1739.74 ± 16.20

Training time (in s)	MADDPG	IQL
Baseline	2689.98 ± 16.77	2467.60 ± 9.16
COUNT	2839.22 ± 18.36	2562.10 ± 10.55
COUNT (joint)	2829.41 ± 35.68	2563.47 ± 6.70
ICM	2986.11 ± 18.22	2711.24 ± 43.49
ICM (joint)	2929.34 ± 17.64	2683.16 ± 39.37
RND	2869.65 ± 54.09	2524.62 ± 23.56
RND (joint)	2821.58 ± 5.83	2533.52 ± 24.25

Table 4.1: Training time on the cooperative communication (top left), cooperative navigation (top right), physical deception (bottom left) and predator prey (bottom right) task.

As expected, training time primarily scales with the number of agents, which explains the difference in training time among tasks.

Additionally, it can be noticed that training for IQL with DQNs is slightly faster than MADDPG training. This is unsurprising given that MADDPG requires optimisation of two separate networks during training. Similarly, ICM requires optimisation of the inverse- and forward models with a generally larger network structure. This causes ICM training to be costlier than RND with comparably small networks. The count-based curiosity with computationally inexpensive hash computation and RND appear to be comparable in additional training time.

Chapter 5

Future Work

In this work, we proposed a general framework for curiosity application as exploration incentives in MARL. In the evaluation, remaining shortcomings were identified motivating further research on the concept.

5.1 Intrinsic Reward Scaling

The application of curiosity, similar to most forms of reward shaping, is highly task-dependent. The primary cause for this variability is highlighted in Section 3.3.1. Accurate scaling of intrinsic rewards has a significant impact. If rewards are too large, they may heavily distort training. On the other hand, small intrinsic rewards will have no impact at all. Similarly, the speed of curiosity decay is essential to allow algorithms to learn from additional exploration without being disrupted. In the case of slow decay, training is presumably negatively affected as the optimal, true value function can only be learned in the absence of intrinsic rewards (if applied as additional rewards as defined in Section 3.2).

Therefore, it is worth considering automated scaling approaches for intrinsic rewards. Stadie et al. [2015] propose a simple approach to scale such exploration bonuses. They normalise the intrinsic reward signal, defined as the prediction error e_T by dividing by the largest error encountered and an additional decay constant $C > 0$. Similarly, our intrinsic reward signal r_t^I at time t could be normalised as follows

$$\bar{r}_t^I = \frac{r_t^I}{t \cdot C \cdot \max_{t' \leq t} r_{t'}^I} \tag{5.1}$$

with similar linear decay using C.

5.2 Curiosity-Assignment Problem

From our experiments, it seems that joint curiosity application is preferable due to its improved stability and synchronised decay compared to decentralised curiosity. Introducing such exploration bonus for MARL appears promising, but comes with a considerable challenge. Currently, we compute curiosity over the joint observations (and actions for ICM) and provide each agent with the same intrinsic reward. This appears suboptimal in most applications. Often, only few agents will be responsible or largely contributing to the received reward bonus. Hence, such agents should be exclusively rewarded rather than rewarding all agents, many of which did not contribute to the rewarded exploration.

This challenge is largely correlated to the credit-assignment problem for MARL, referred to in Section 2.0.1, as it essentially poses the same challenge for intrinsic rewards. Any method to reason about the individual contribution of agents to the computed curiosity could enhance exploration guidance. Foerster et al. [2018] proposed a counterfactual baseline for multi-agent policy gradients in an approach similar to MADDPG. A centralised critic shared among all agents enables reasoning about the expected advantage of choosing a single action over other actions for each agent. A similar approach could be applied to curiosity in MARL by introducing an additional head for the value function network (like DQN or MADDPG critic) of each agent. This head would be trained to estimate the received joint curiosity based on joint action and observation. Using such a curiosity critic, we could identify an agent's contribution to received intrinsic reward by approximating the curiosity that would have been received for other actions of that particular agent.

5.3 Communicated Curiosity

While joint curiosity appears superior to individual curiosity application, it requires centralised training as enforced by common MARL algorithms like MADDPG [Lowe et al., 2017] or COMA [Foerster et al., 2018]. However, in some applications such training is impossible and therefore decentralised curiosity has to be applied. In these cases, communication among agents could be a possible solution to coordinate exploration efforts. Communication has already led to considerable improvements in cooperative MARL tasks [Foerster et al., 2016]. Curiosity signals could be shared across such communication channels and each individually motivated agent could compute a

combined intrinsic reward based on all received curiosities, similar to the centralised approach proposed by Iqbal and Sha [2019], for flexible and stable exploration.

5.4 Decentralised Curiosity

Independent application of curiosity appears to be much more variant than joint application. Hence, it would be advantageous to be able to apply such joint curiosity even during decentralised training. We propose to research the applicability of modelling approaches for predicted joint curiosity. Each agent could construct models of other agent policies to predict their action choices. This could be used to compute pseudo-joint curiosities based on individual observations and joint actions, where actions of other agents are predicted using trained models.

Such modelling approaches are already widely used in multi-agent systems to reason about the behaviour of other agents [Albrecht and Stone, 2018] and could assist decentralised curiosity application next to communication.

5.5 Noisy Environment

Curiosity approaches are known to be sensitive to noise as illustrated by Burda et al. [2018a]. For count-based approaches, noisy environments can cause observations to be misleadingly interpreted as different, while prediction-based curiosity would be unable to foresee stochastic influences. This leads to agents remaining curious, which can negatively affect training, and loses its purpose as guided exploration incentives. Hence, we propose to research the impact of noise on curiosity in MARL.

It should be distinguished between “static noise”, which is additionally received as input without disturbing other parts of observations, and “environmental noise”. Latter is integrated into the environment. Therefore, agents can actively choose to step into it to constantly receive misleading curiosity. Such environmental noise corresponds to the noisy-TV problem introduced by Burda et al. [2018a]. Proposed “static noise” is comparable to the white noise added around the original input by Pathak et al. [2017] in their experiments.

Both forms of noise were implemented for the evaluated particle environment tasks and are publicly available. We encourage experimentation with these tasks for further research of curiosity in multi-agent systems.

Chapter 6

Conclusion

This dissertation aimed to research the influence of curiosity as exploration incentives for MARL. We evaluated the impact on policy-gradient and value-based MARL in competitive and cooperative tasks.

When applied for the original MAPE tasks, curiosity leads to considerable distortion of training. It is not just unable to provide useful, guided exploration, but harms training stability, convergence and final performance. This can be observed on all tasks using any of the curiosity approaches. Against our initial expectation, similar results are obtained under partial observability. While curiosity appears to not harm performance, it does not alleviate the instability introduced by incomplete information about the environmental state.

However, curiosity has a considerable effect on policy-gradient methods when trained with sparse rewards. Without additional intrinsic rewards, training of such MARL appears highly unstable and unable to learn any guided behaviour. The baseline performance trained with continuous rewards is not reached even with additional curiosity, but training becomes more reliable and agents show considerable learning on most tasks. On the other hand, value-based MARL appears to be more stable under such conditions in general and is hardly influenced by curiosity.

We hope that the research conducted during this project shows the prospect of intrinsic rewards and serves as a foundation for further research to expand on. Chapter 5 highlights some possible directions to improve decentralised and centralised curiosity application in MARL. Furthermore, we motivate the challenge of noisy environments, which has not been addressed yet for intrinsic rewards in MARL.

Appendix A

Implementation Dependencies

The entire project is implemented in *Python* 3.7 [Python Core Team, 2019] and scientific computation uses the *Numpy* [Oliphant, 06] library, version 1.15. All deep learning networks for baseline algorithms and curiosity methods were defined with *PyTorch* 1.1.0 [Paszke et al., 2017]. A full list of all such dependencies is shown in Table A.1.

Our implementations and respective documentation are publicly available under `https://github.com/LukasSchaefer/MSc_Curiosity_MARL.git` Table A.1 shows a full list of software dependencies for the project.

Name	Description	Version	Citation	Source
Python	programming language	3.7	Python Core Team [2019]	`https://www.python.org`
NumPy	scientific computation library	1.15	Oliphant [06]	`https://numpy.org`
PyTorch	deep learning platform	1.1.0	Paszke et al. [2017]	`https://pytorch.org`
Matplotlib	2d plotting library	3.0.3	[Hunter, 2007]	`https://matplotlib.org`
OpenAI Gym	RL environment toolkit	0.10.5	Brockman et al. [2016]	`https://gym.openai.com`
MAPE[1]	MARL environment	/	Lowe et al. [2017]	[2]

Table A.1: Dependencies for Implementation

Matplotlib [Hunter, 2007] was used to generate all plots found in this dissertation.

[1] Multi-agent particle environment

[2] Modified version for partial-observability available under `https://github.com/LukasSchaefer/multiagent-particle-envs`. Original version can be found under `https://github.com/openai/multiagent-particle-envs`.

Appendix B

Evaluation Parameterisation

Table B.1 shows a full list of hyperparameters used for the evaluation.

ε-greedy policies in IQL are initially using ε of 1. After, the exploration factor is continuously decayed over the number of episodes, N_{episodes}, until a value of 0.01 is reached. This is achieved by multiplying ε with a decay factor at each step. The decay factor is computed as $\sqrt[n]{0.01}$, where $n = N_{\text{episodes}} \cdot N_{\text{max-episode-length}}$. This ensures that after n environment steps $\varepsilon = (\sqrt[n]{0.01})^n \cdot 1.0 = 0.01$.

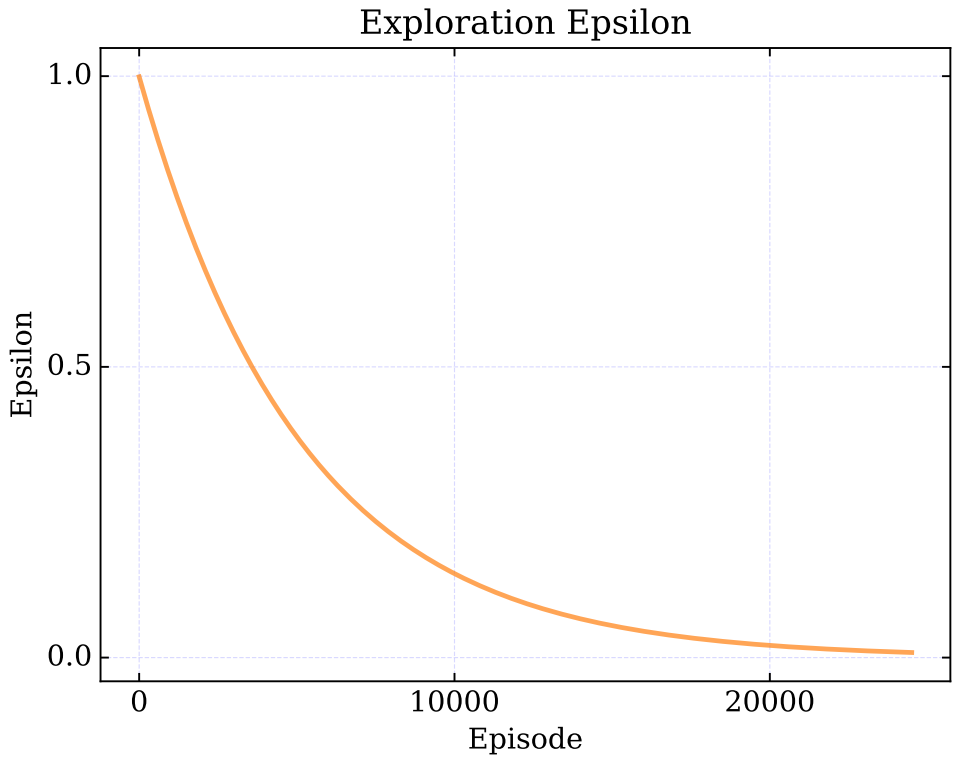

Figure B.1: ε-decay for 25,000 episodes each having 25 steps.

Name	Description	Value
N_{episodes}	total number of episodes	$25,000$
$N_{\text{max-episode-length}}$	maximum length of an episode	25
$N_{\text{update-freq}}$	number of environment steps before optimisation	100
d^h	hidden dimensionality of algorithm networks	128
α_{MADDPG}	learning rate for MADDPG optimiser	0.01
α_{IQL}	learning rate for IQL optimiser	0.001
γ	discount factor	0.9
τ	soft target update factor	0.01
$N_{\mathcal{D}}$	replay buffer capacity	$1,000,000$
$N_{\text{batch-size}}$	replay buffer batch size	$1,024$
η	scaling weight factor for curiosity reward	5
d^{h_c}	hidden dimensionality of curiosity networks	128
d^{ϕ}	dimensionality of state representation of curiosity	64
α_c	learning rate for curiosity optimiser	$2e^{-6}$
d^k	dimensionality of key for hash-count based curiosity	32
δ	decay rate for hash-count based curiosity	1
$N_{\text{eval-freq}}$	number of episodes before evaluation episodes	100
$N_{\text{eval-eps}}$	number of episodes for each evaluation	5
$d_{\text{partial-obs}}$	distance for partial observability	0.5
$N_{\text{sparse-freq}}$	number of environment steps before rewards	25

Table B.1: Hyperparameters for evaluation

Appendix C

Evaluation Results

The following appendix lists plots and tables illustrating the performance of all configurations and curiosities on all tasks. First, the episodic rewards obtained by each applied configuration is reported for each task, including partial observable versions and tasks with sparse rewards.

Secondly, the intrinsic rewards computed by each approach applied decentralised and jointly are illustrated. Lastly, the training time for each configuration is highlighted.

C.1 Episodic Rewards

C.1.1 Multi-Agent Particle Environment

C.1.1.1 Cooperative Communication

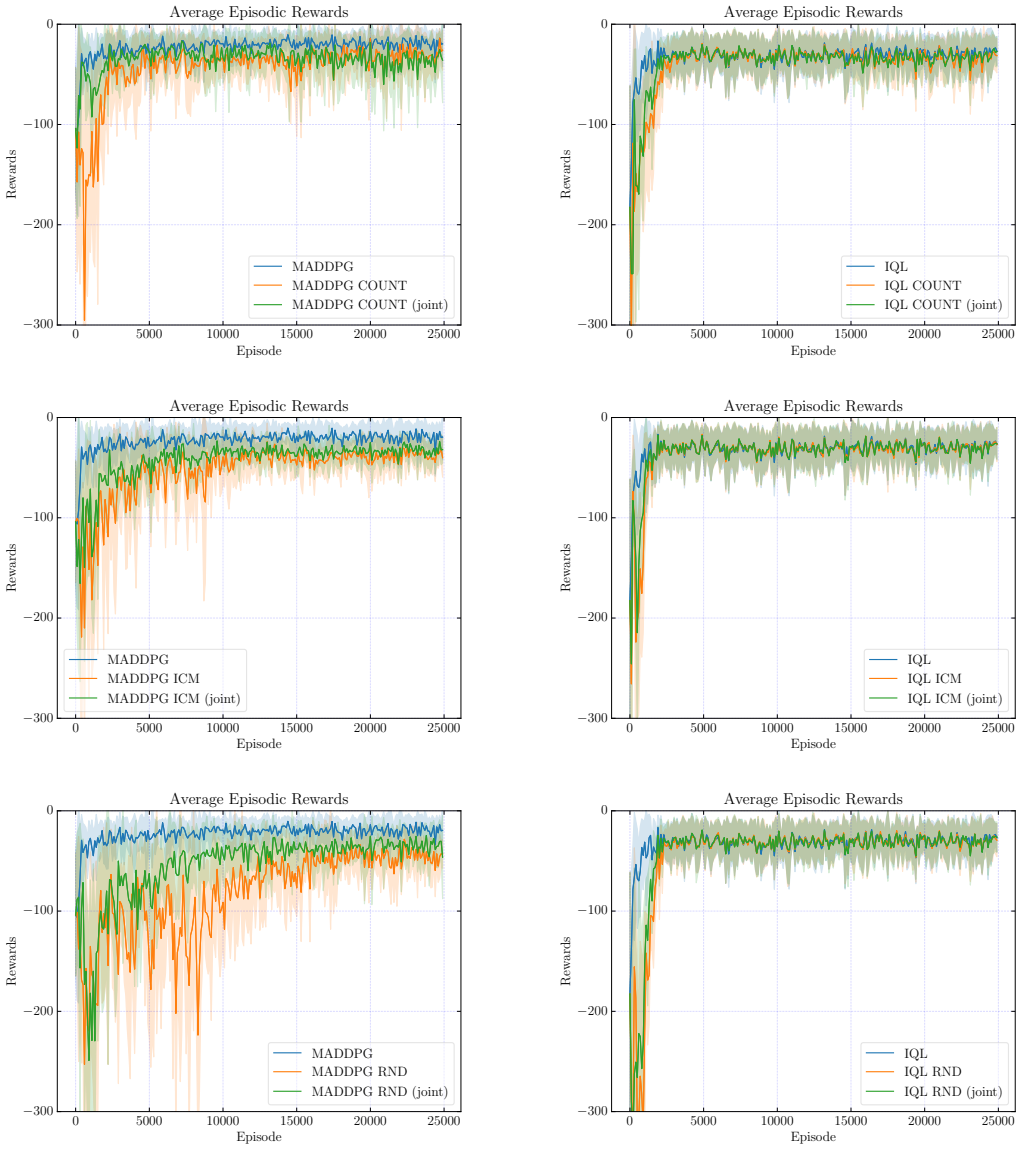

Figure C.1: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative communication task.

C.1.1.2 Cooperative Navigation

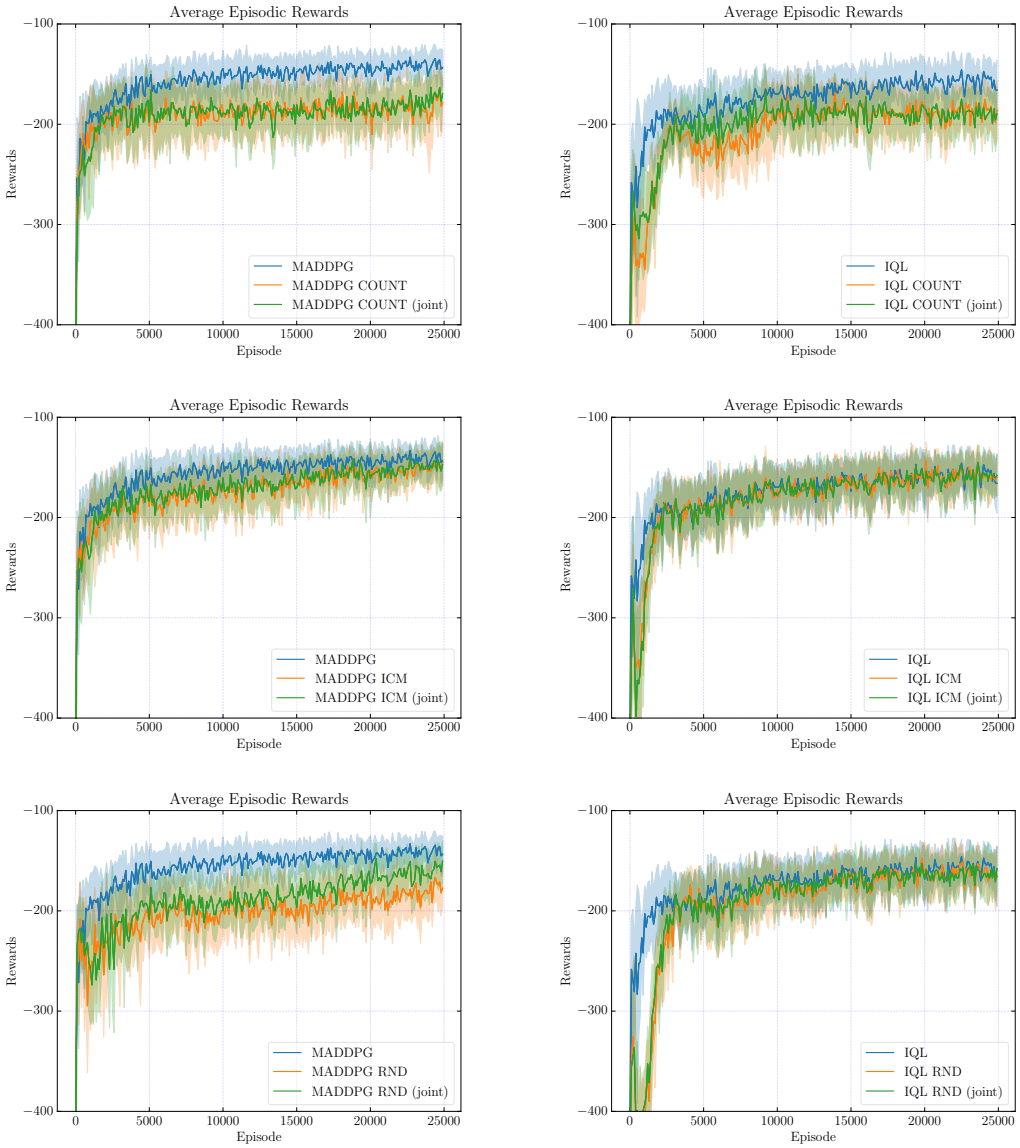

Figure C.2: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative navigation task.

C.1.1.3 Physical Deception

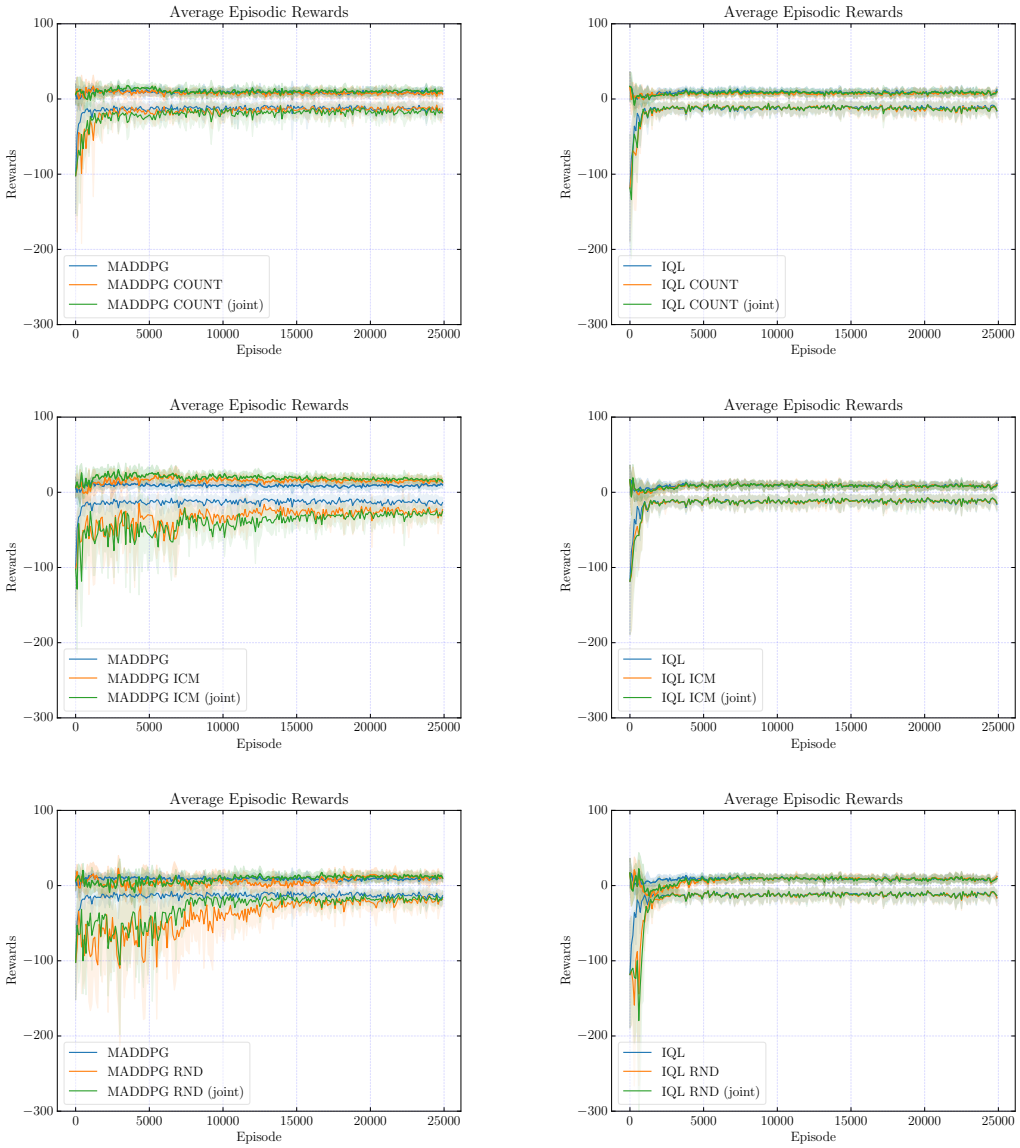

Figure C.3: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for physical deception task.

C.1.1.4 Predator Prey

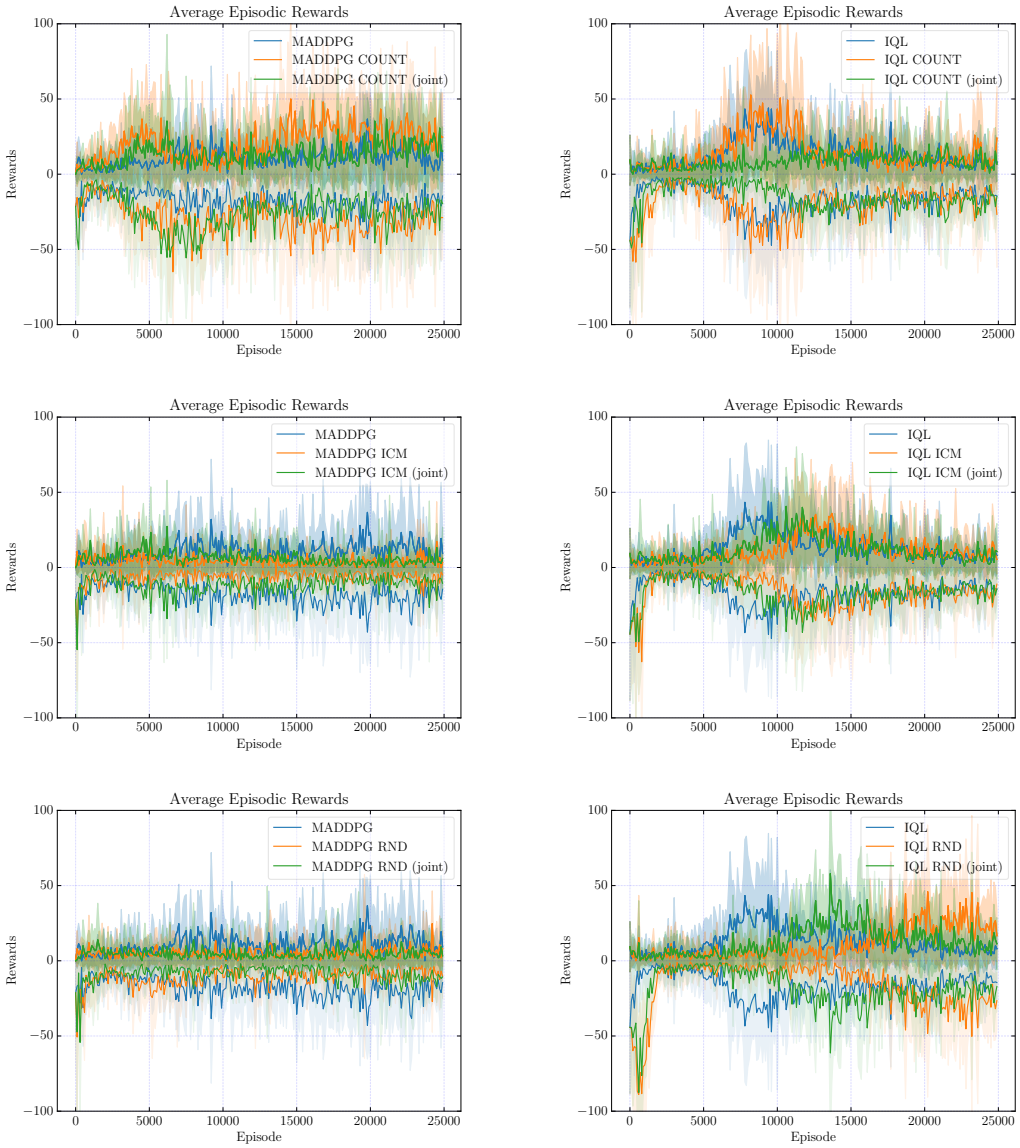

Figure C.4: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for predator prey task.

C.1.2 Multi-Agent Particle Environment under Partial Observability

C.1.2.1 Cooperative Communication

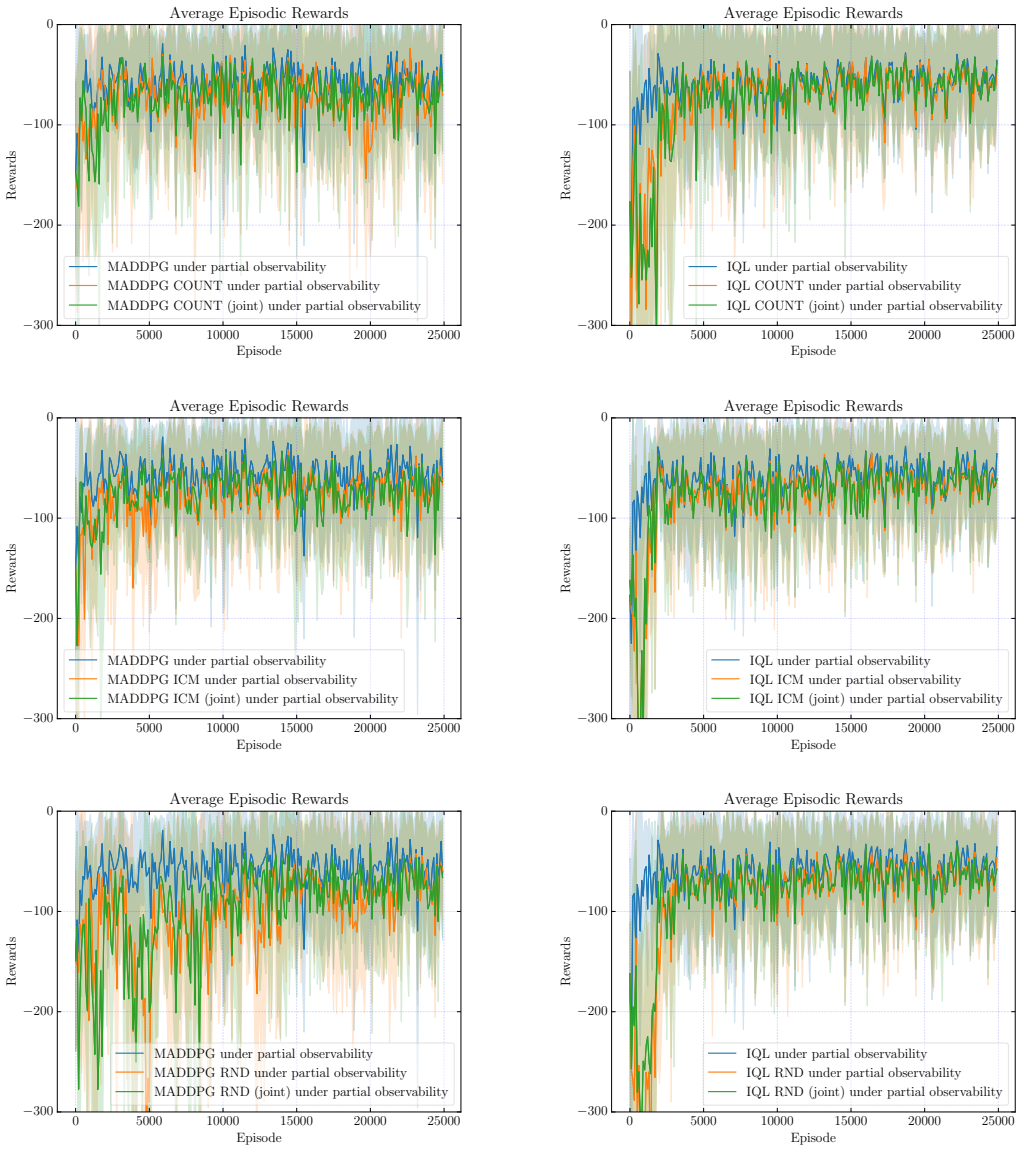

Figure C.5: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative communication task under partial observability.

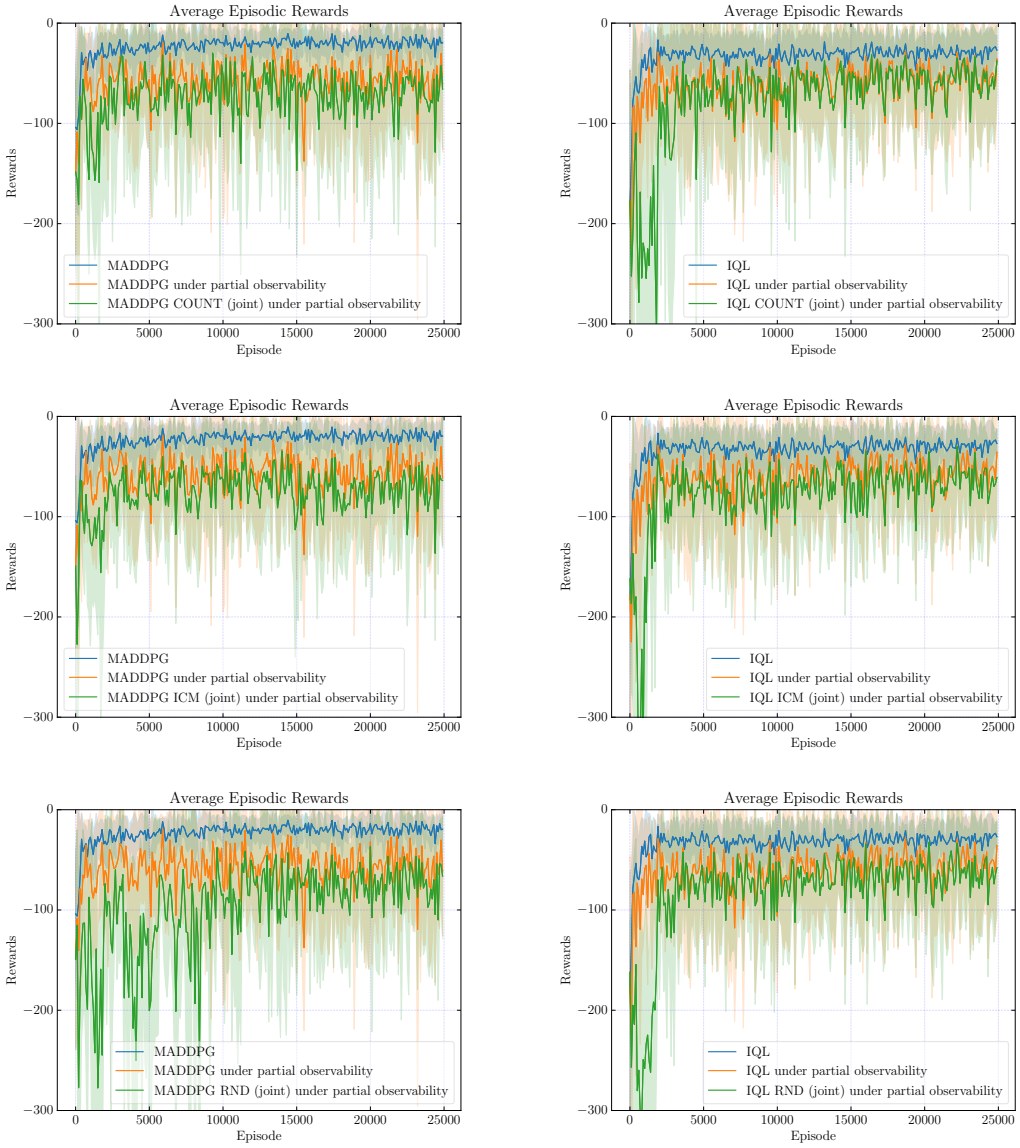

Figure C.6: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for cooperative communication task with and without partial observability.

C.1.2.2 Cooperative Navigation

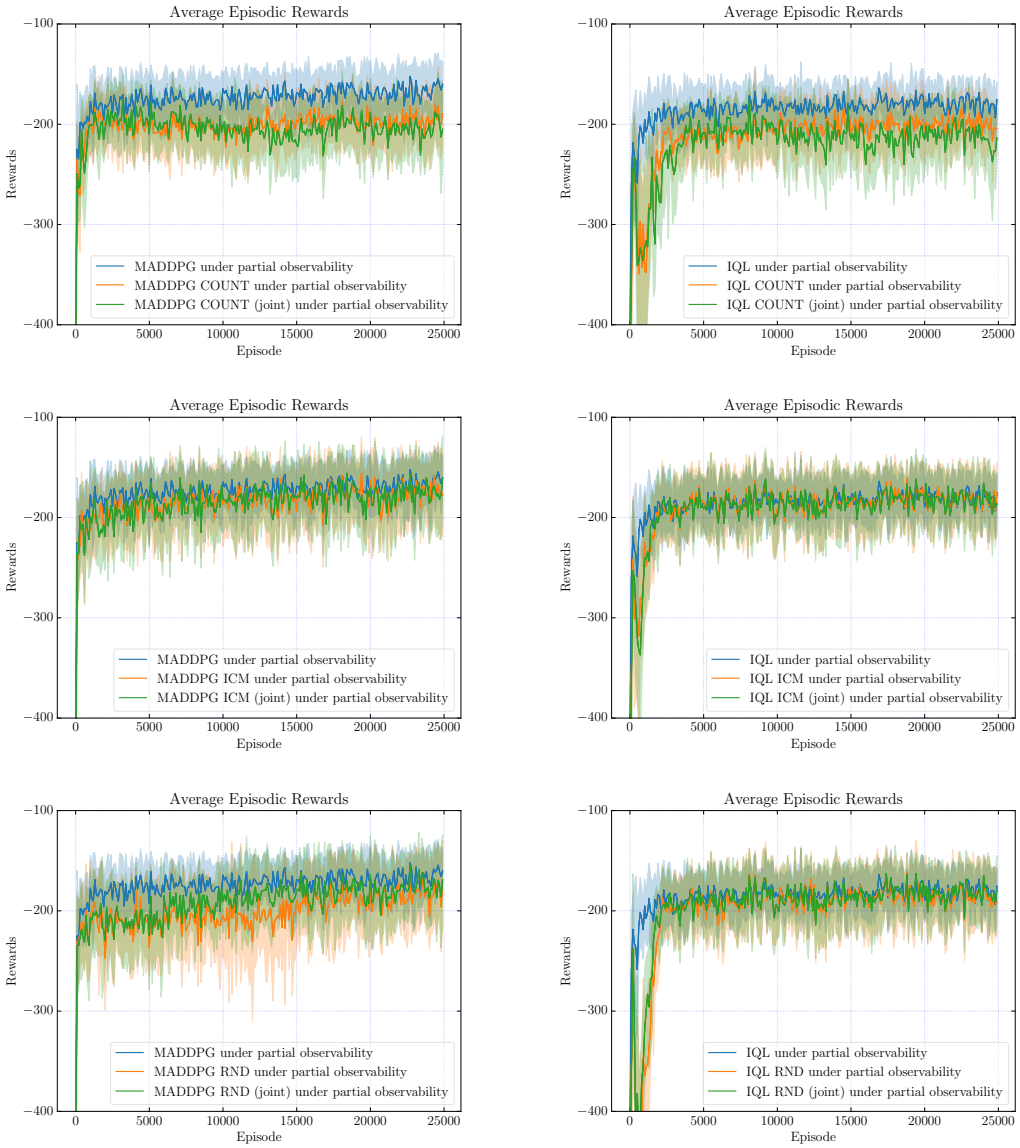

Figure C.7: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative navigation task under partial observability.

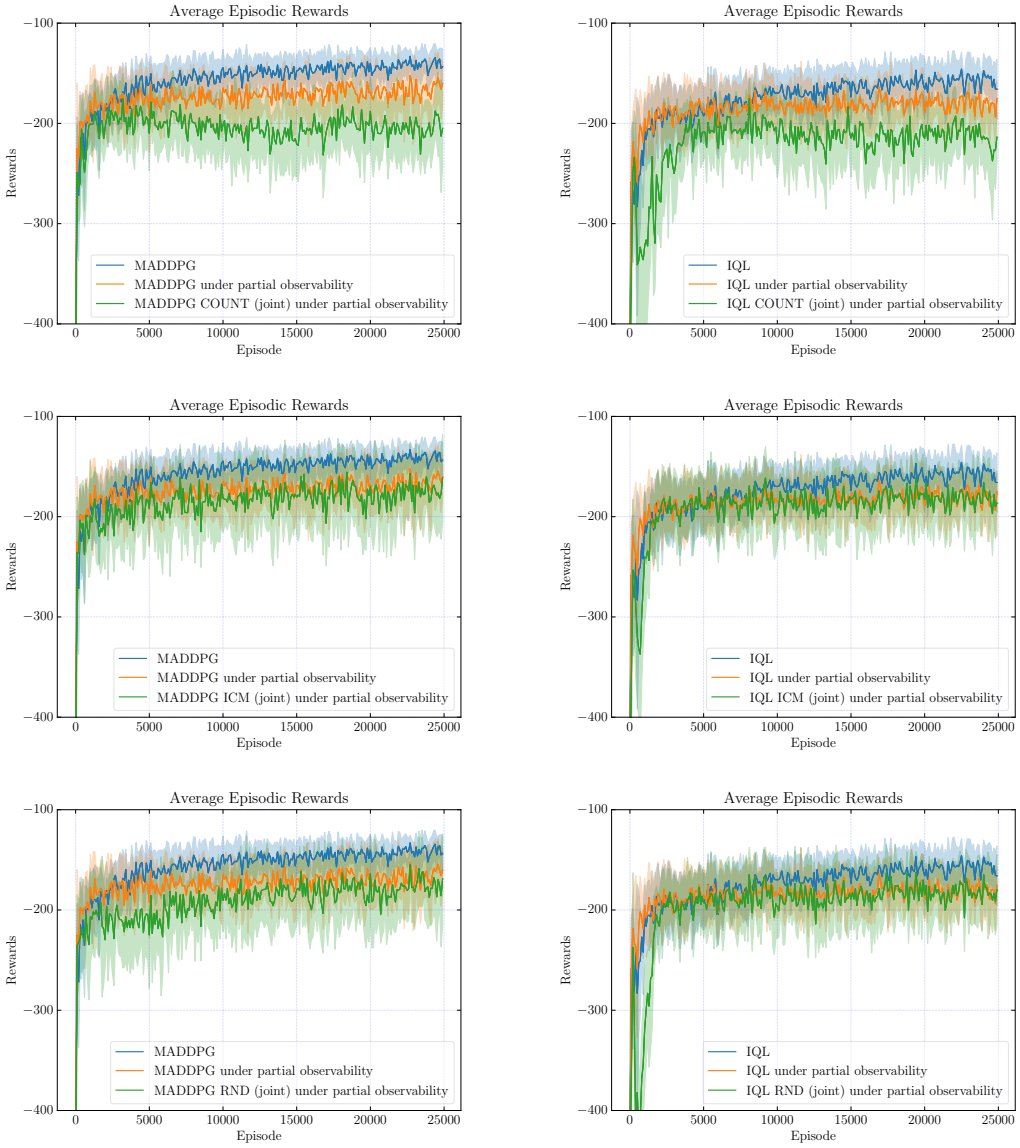

Figure C.8: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for cooperative navigation task with and without partial observability.

C.1.2.3 Physical Deception

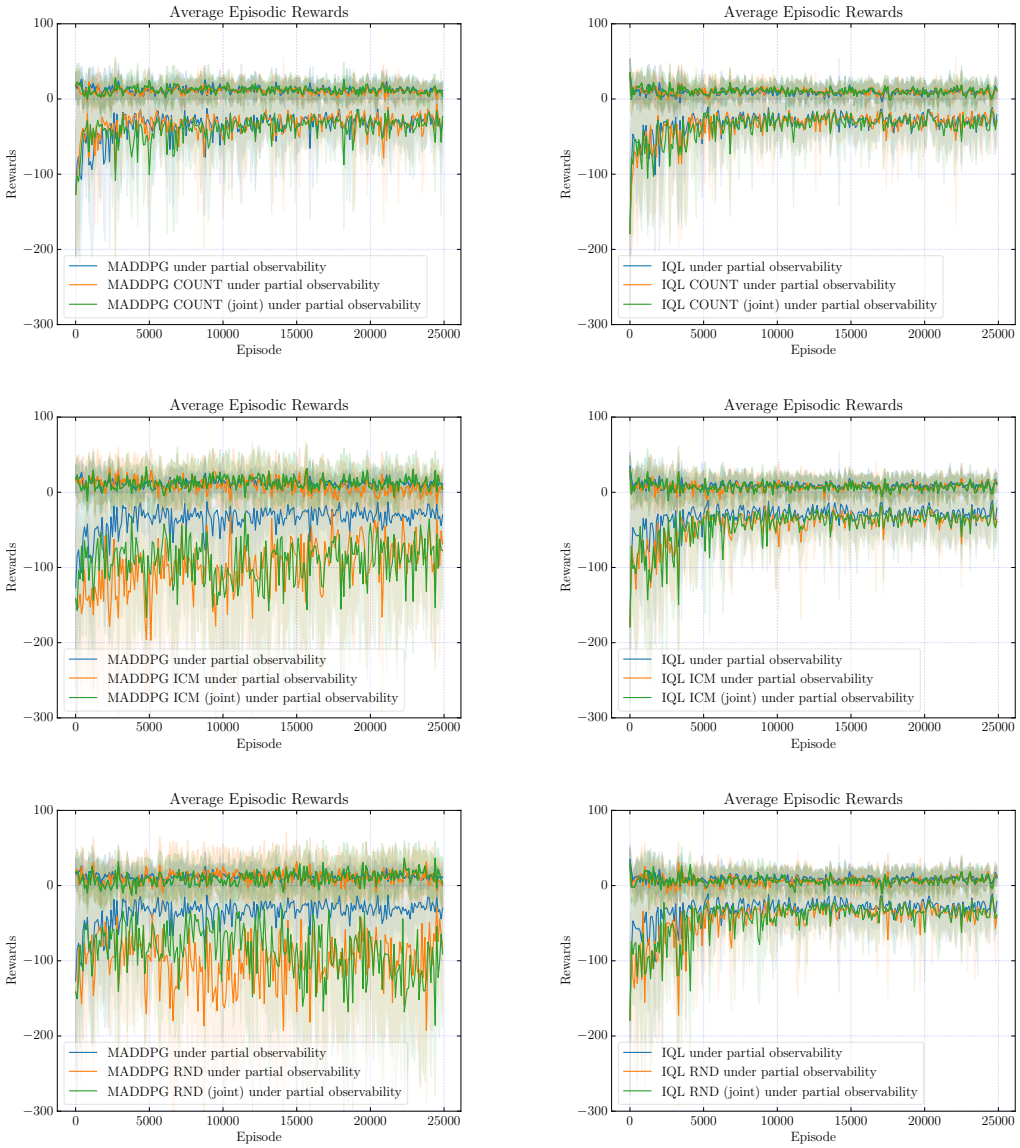

Figure C.9: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for physical deception task under partial observability.

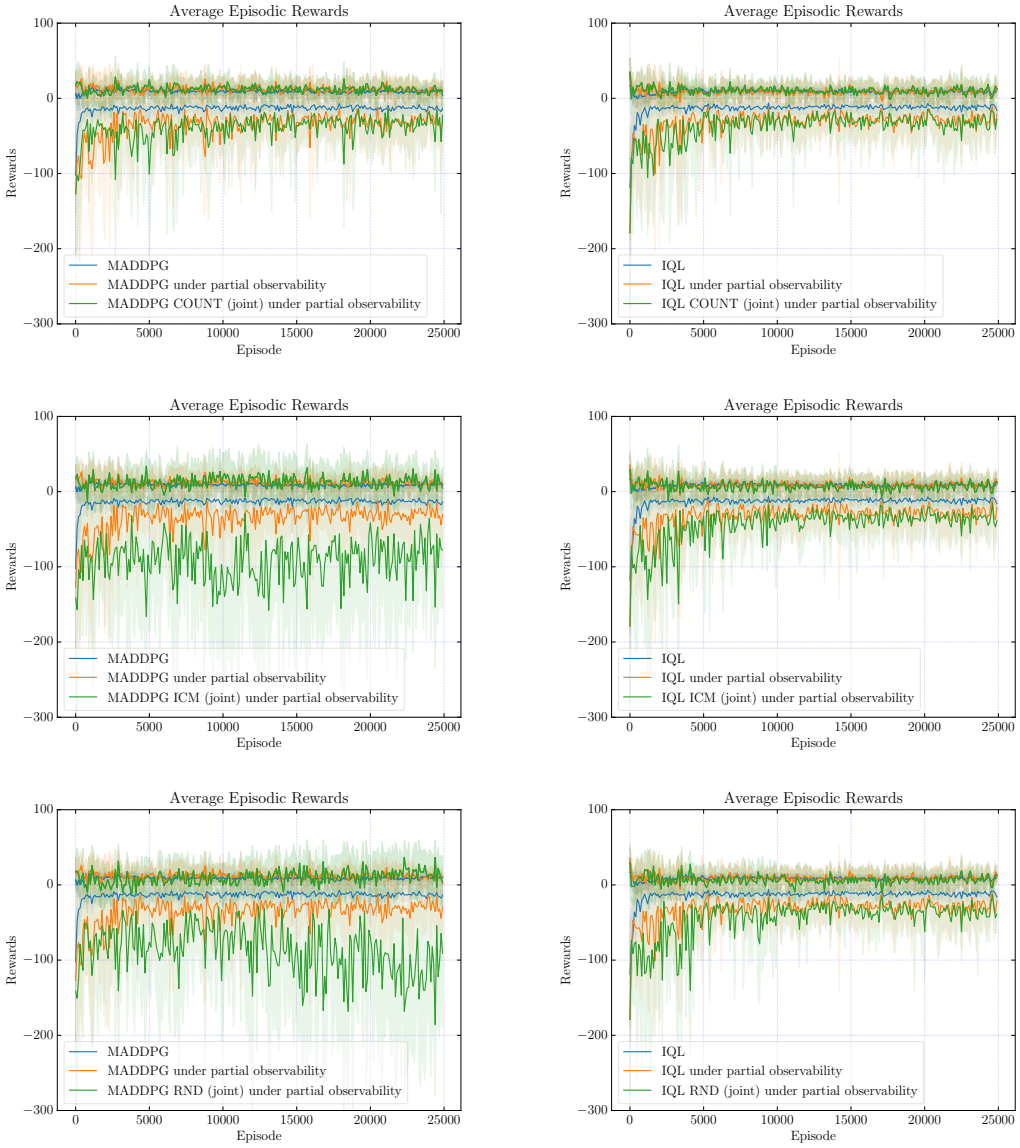

Figure C.10: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for physical deception task with and without partial observability.

C.1.2.4 Predator Prey

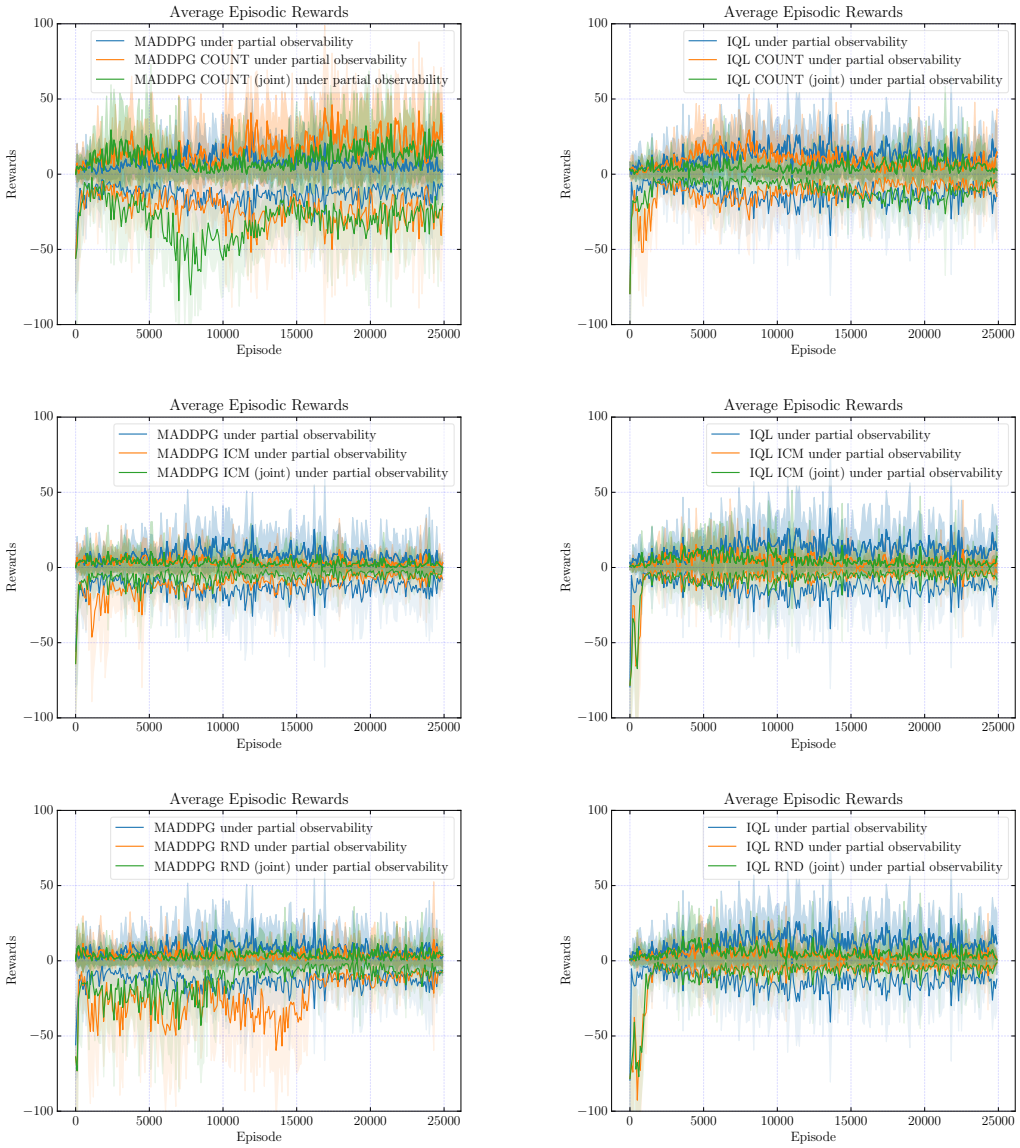

Figure C.11: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for predator prey task under partial observability.

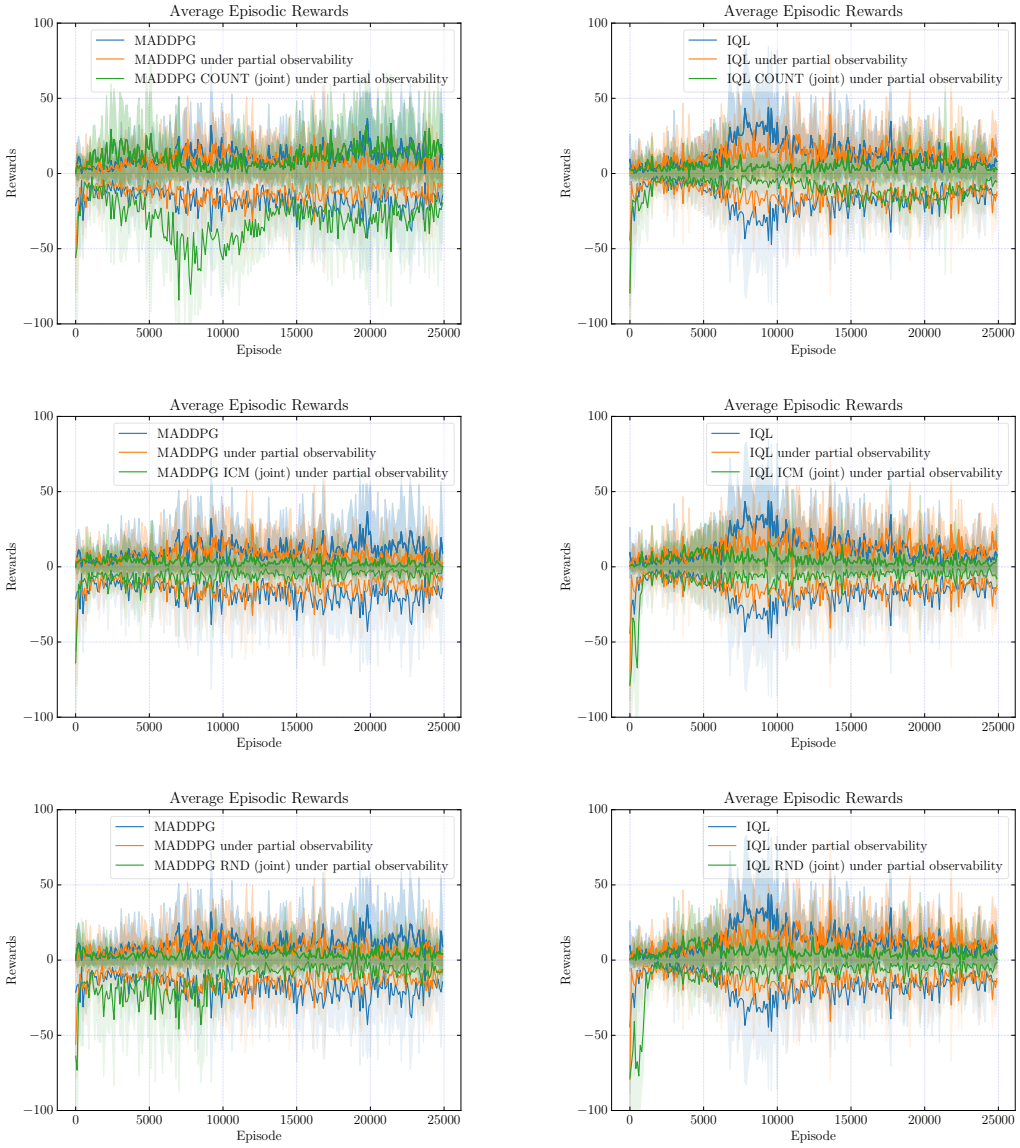

Figure C.12: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for predator prey task with and without partial observability.

C.1.3 Multi-Agent Particle Environment with Sparse Rewards

C.1.3.1 Cooperative Communication

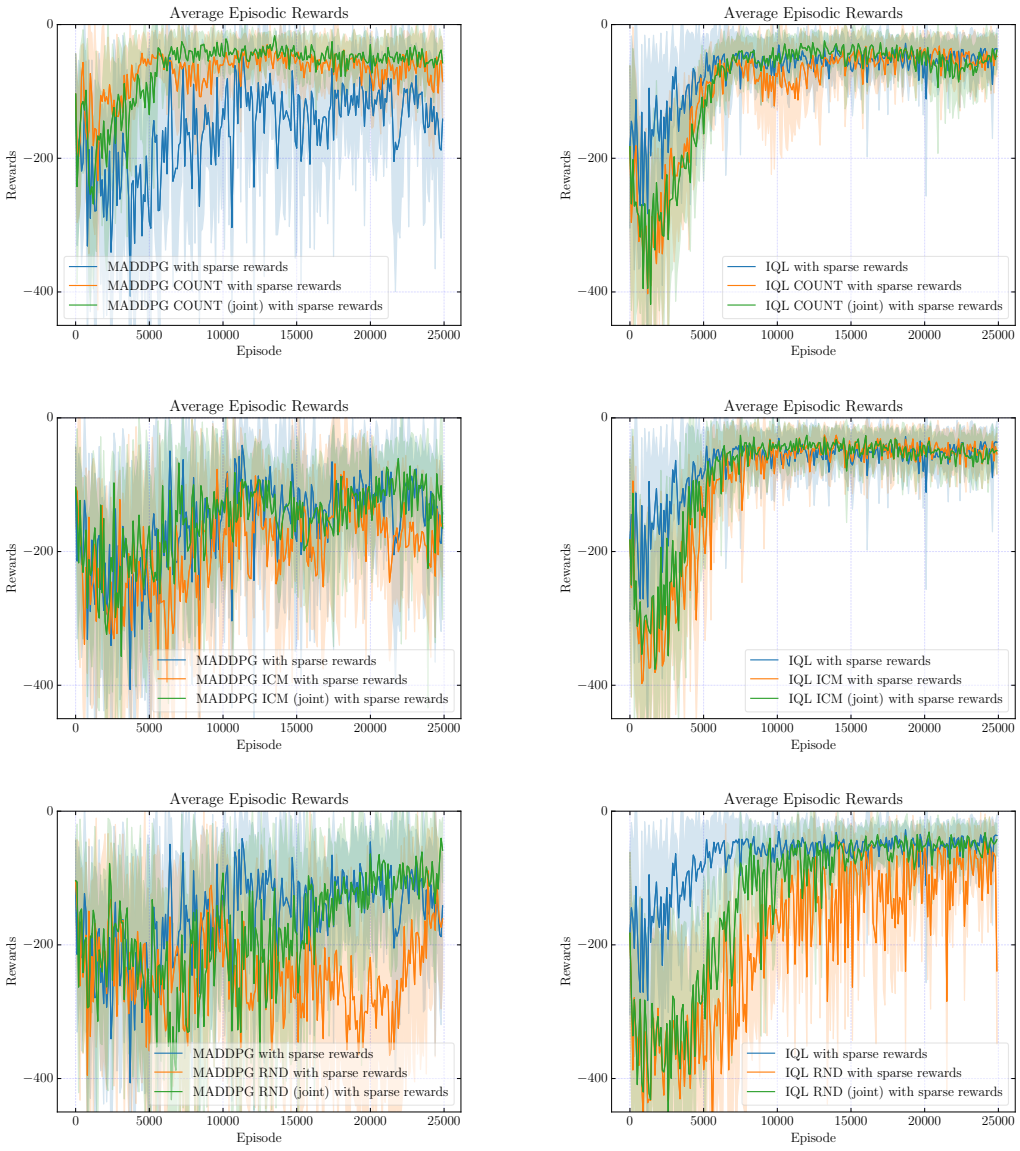

Figure C.13: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative communication task with sparse rewards.

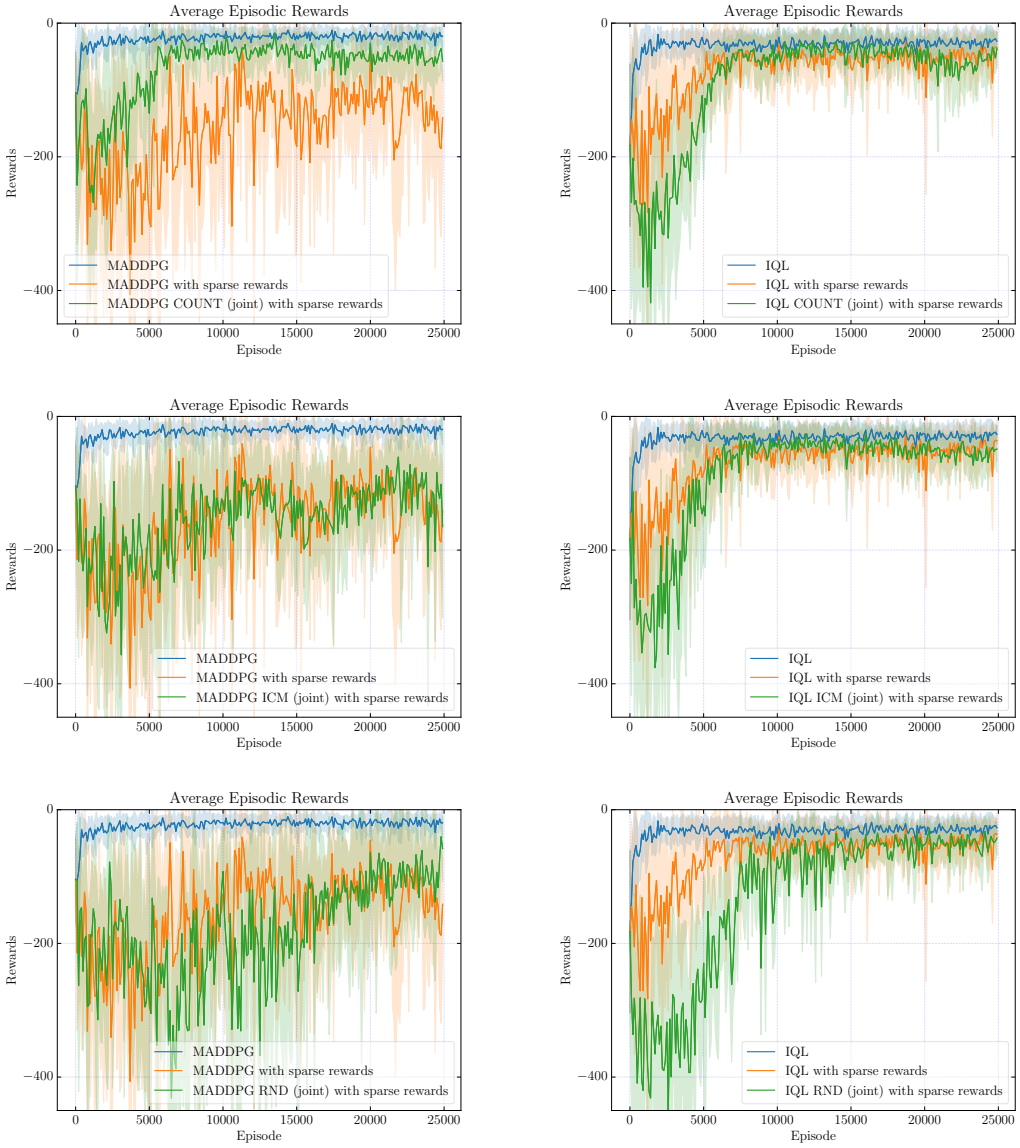

Figure C.14: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for cooperative communication task with and without sparse rewards.

C.1.3.2 Cooperative Navigation

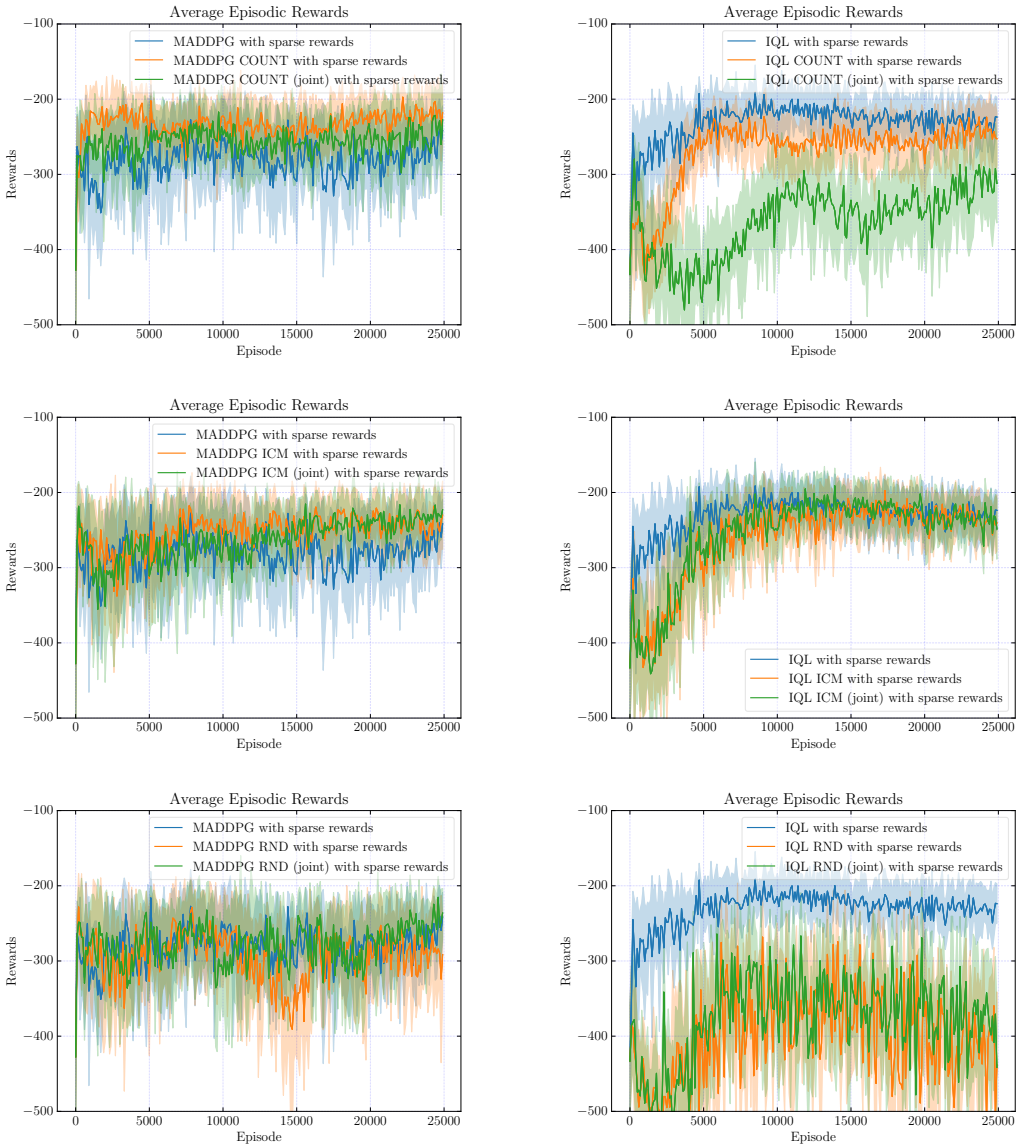

Figure C.15: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for cooperative navigation task with sparse rewards.

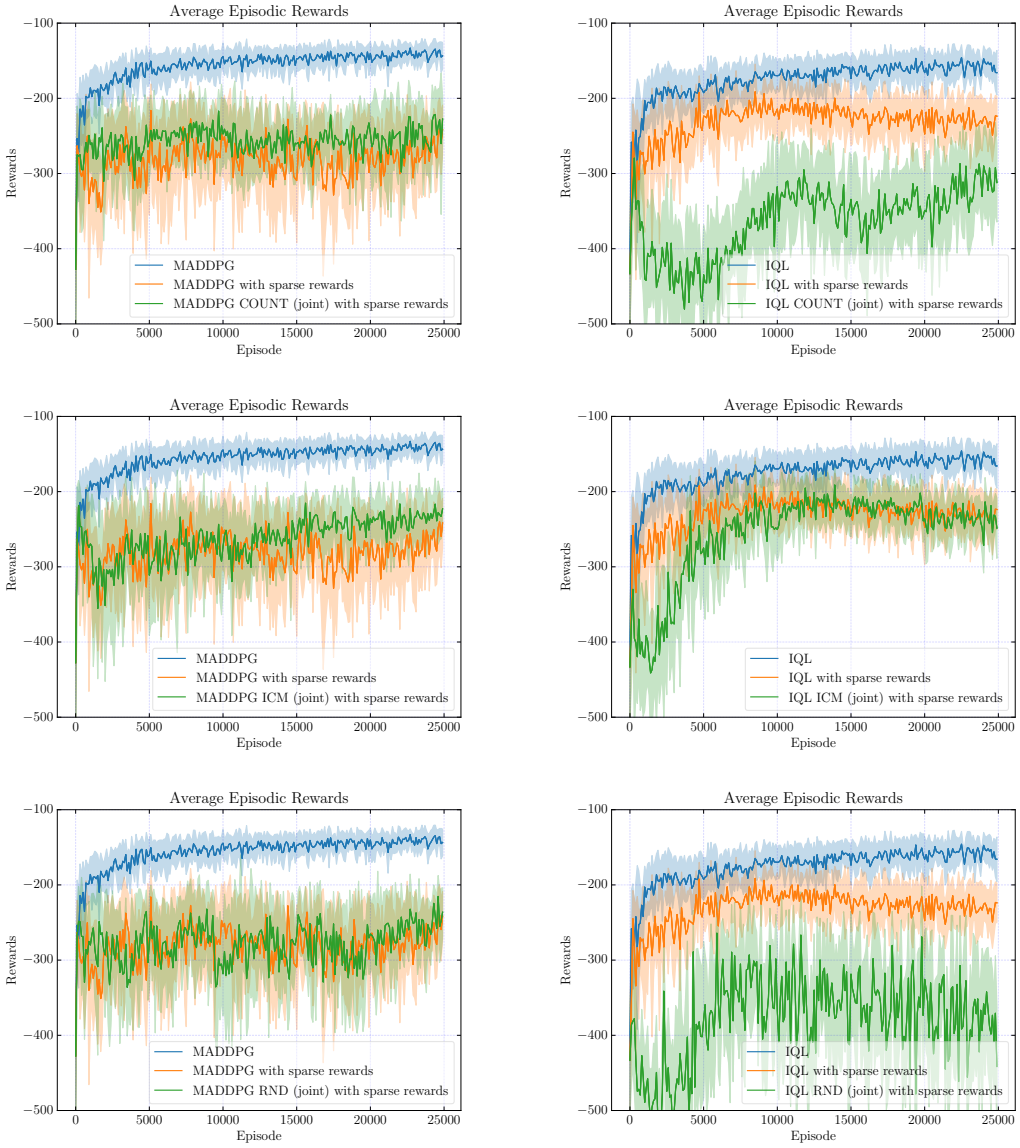

Figure C.16: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for cooperative navigation task with and without sparse rewards.

C.1.3.3 Physical Deception

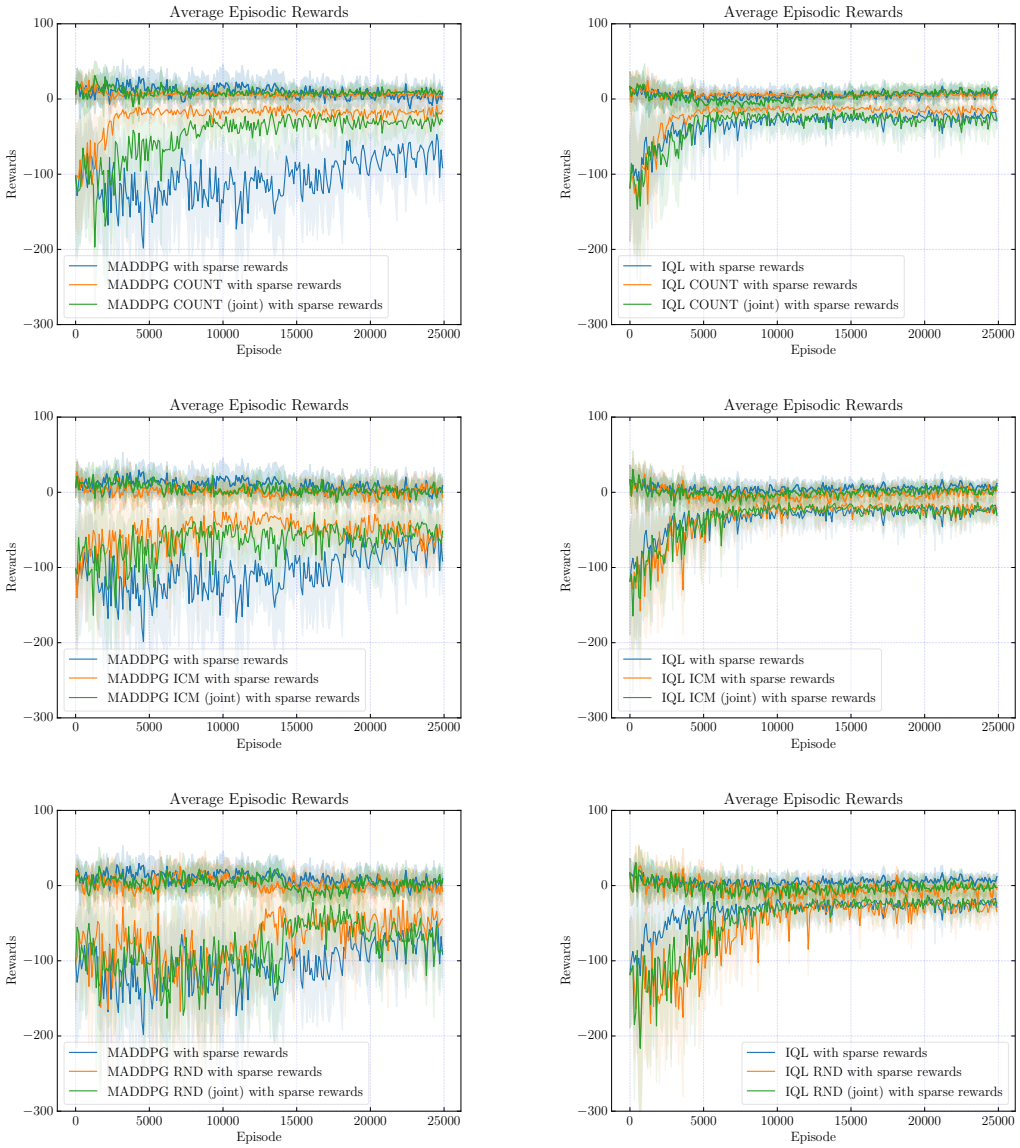

Figure C.17: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for physical deception task with sparse rewards.

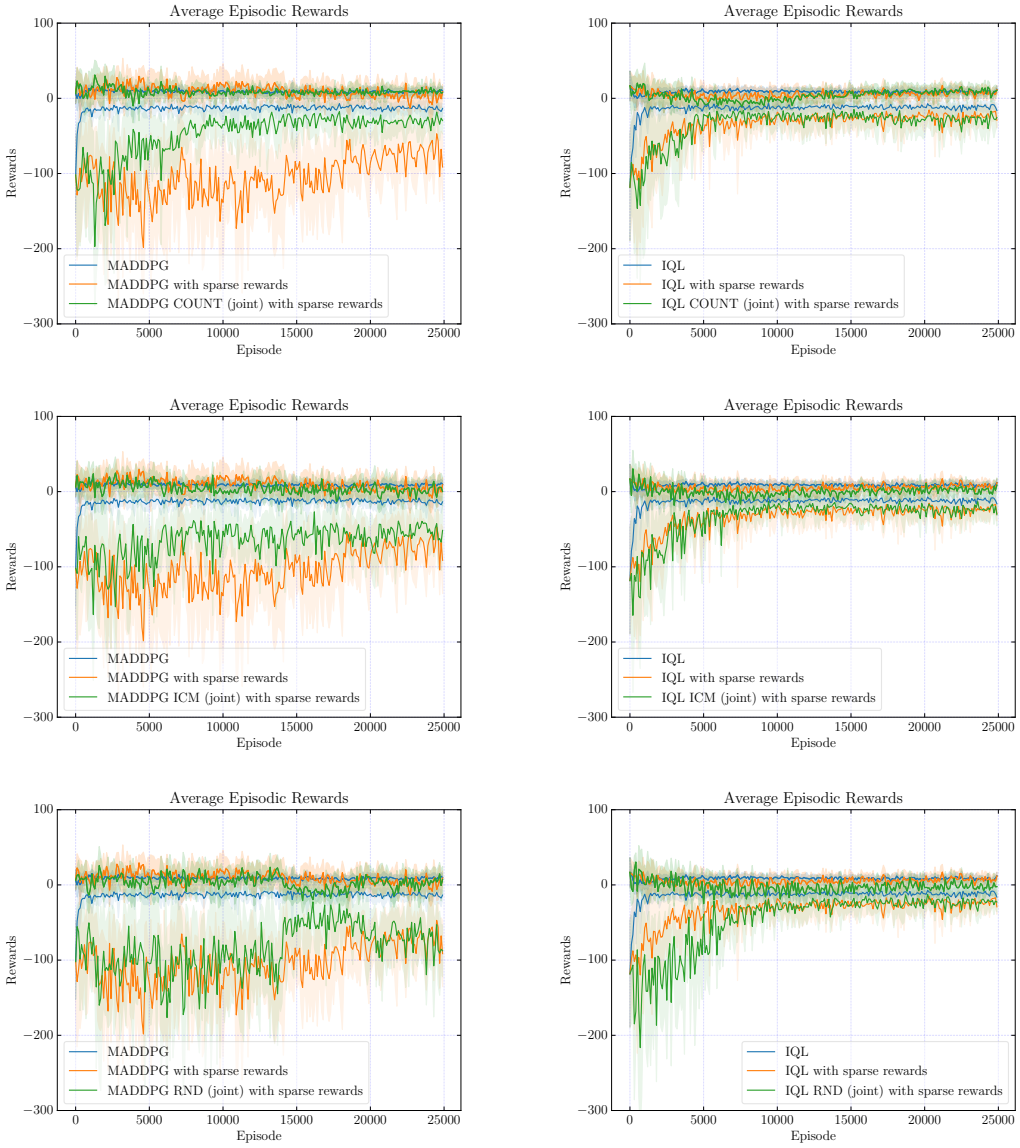

Figure C.18: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for physical deception task with and without sparse rewards.

C.1.3.4 Predator Prey

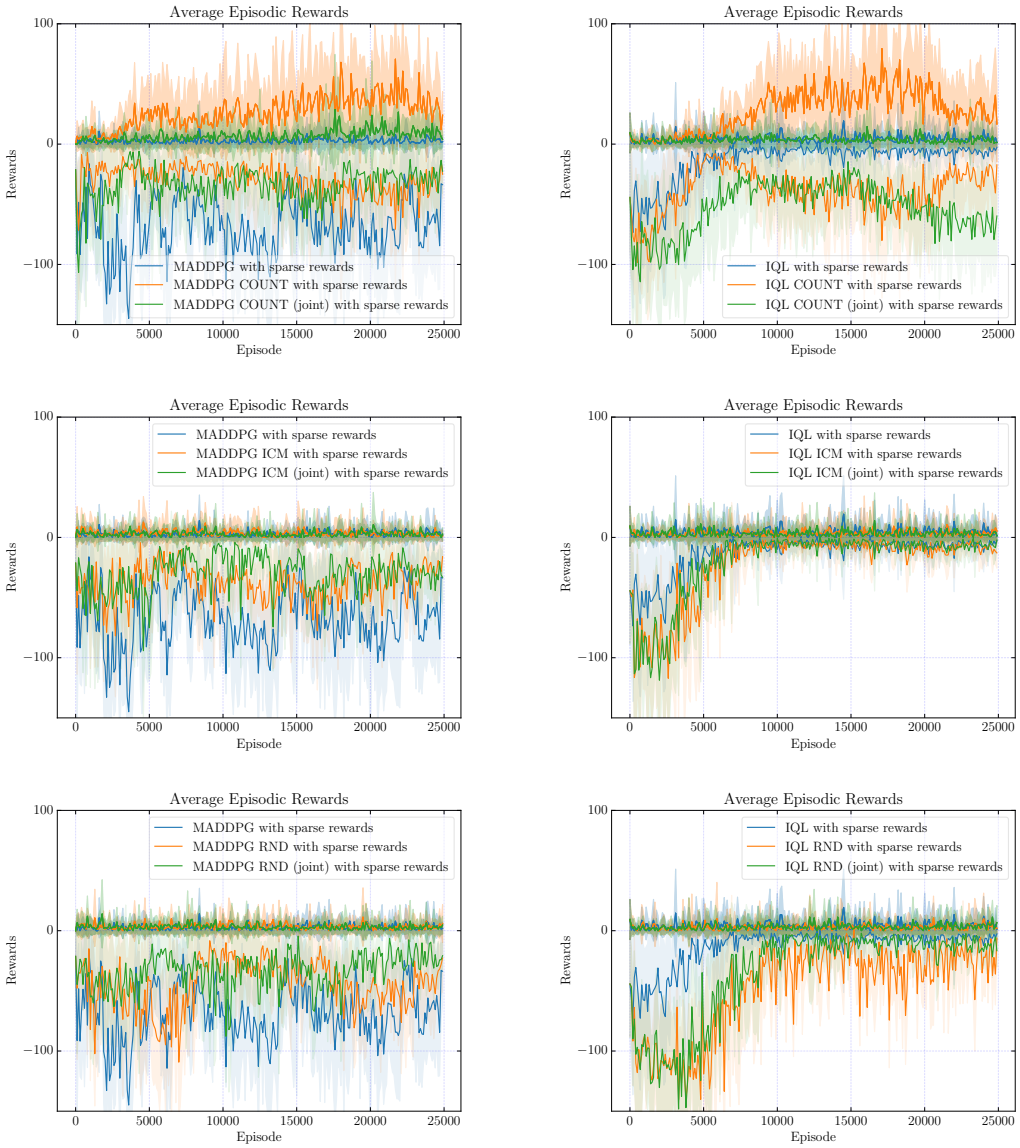

Figure C.19: Episodic rewards for MADDPG (left column) and IQL (right column) with individual and joint curiosities for predator prey task with sparse rewards.

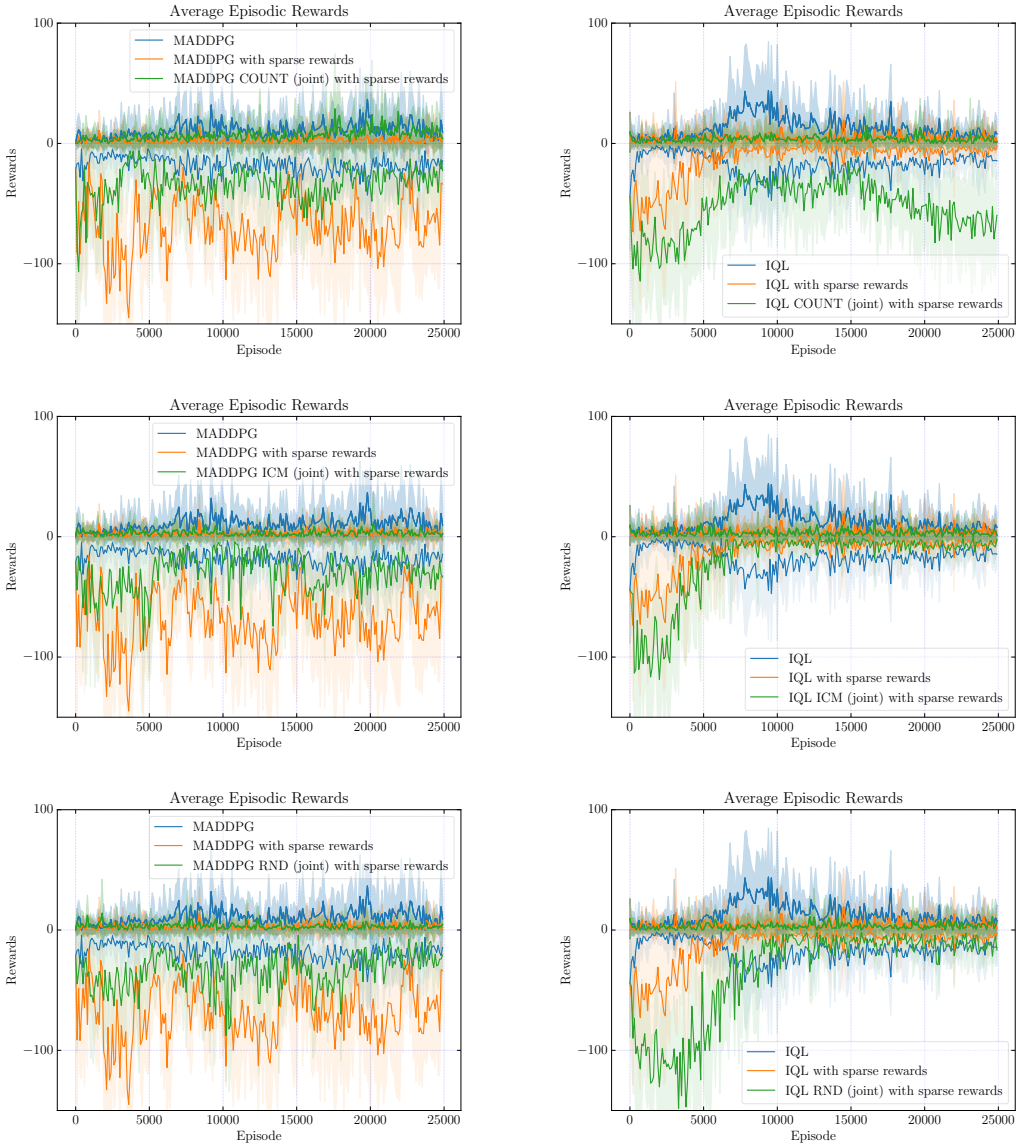

Figure C.20: Episodic rewards for MADDPG (left column) and IQL (right column) with joint curiosities for predator prey task with and without sparse rewards.

C.2 Intrinsic Rewards

C.2.1 Multi-Agent Particle Environment

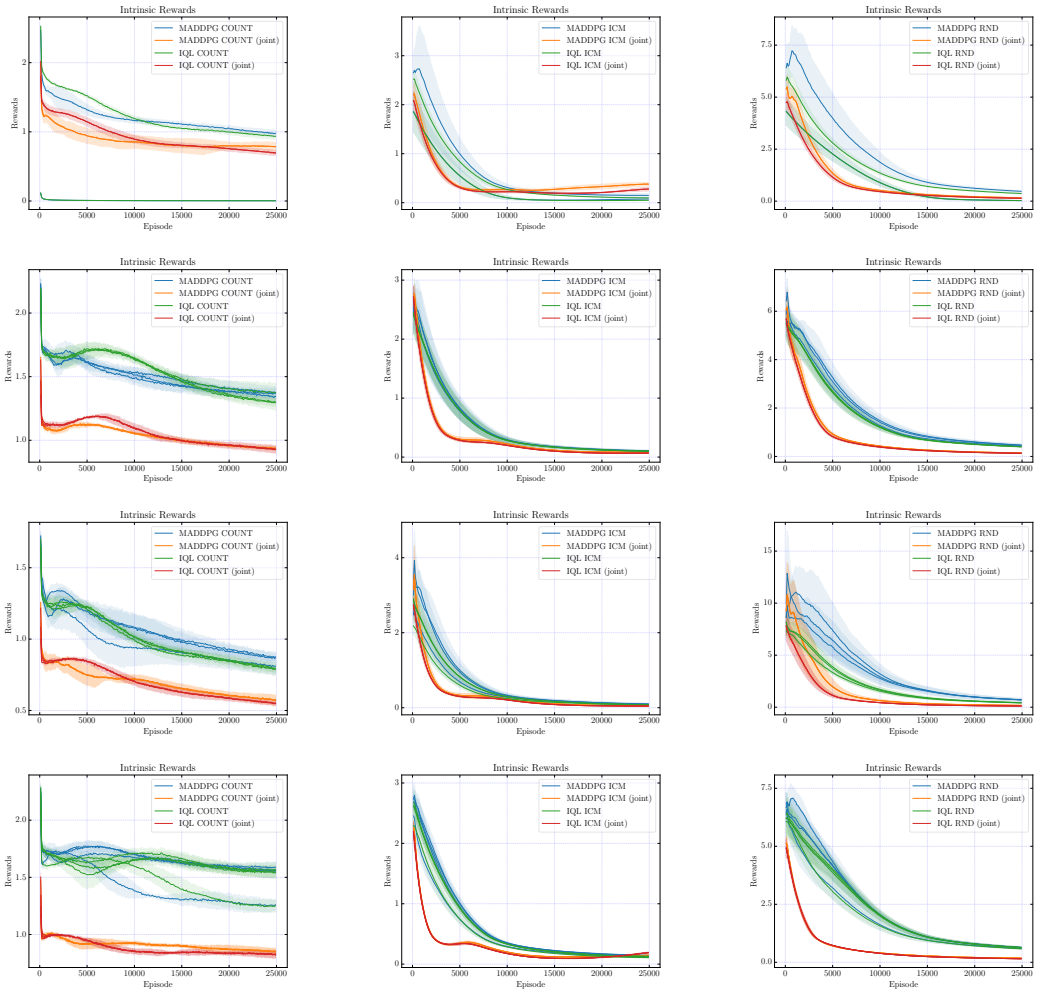

Figure C.21: Intrinsic rewards for MADDPG and IQL with individual and joint curiosities for cooperative communication (1st row), cooperative navigation (2nd row), physical deception (3rd row) and predator prey (4th row) task.

C.2.2 Multi-Agent Particle Environment under Partial Observability

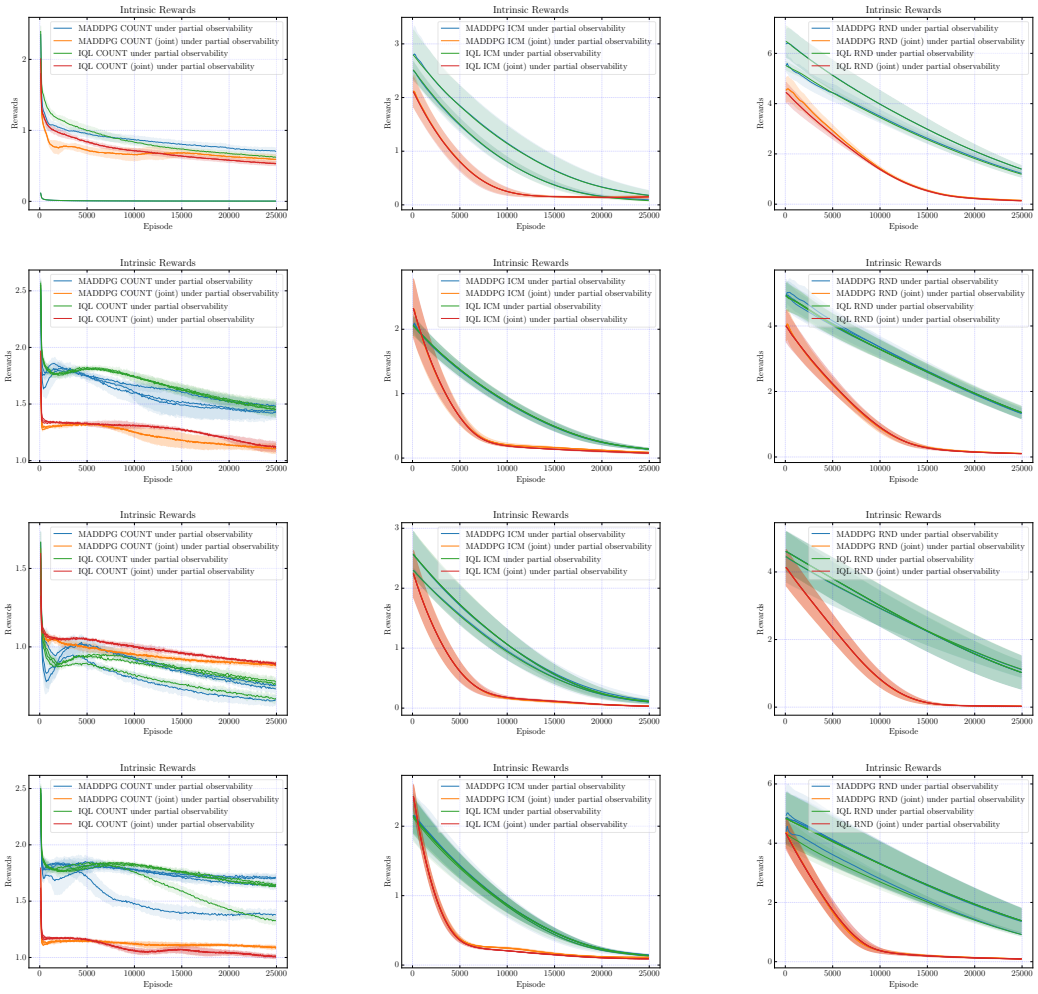

Figure C.22: Intrinsic rewards for MADDPG and IQL with individual and joint curiosities for cooperative communication (1st row), cooperative navigation (2nd row), physical deception (3rd row) and predator prey (4th row) task under partial observability.

C.2.3 Multi-Agent Particle Environment with Sparse Rewards

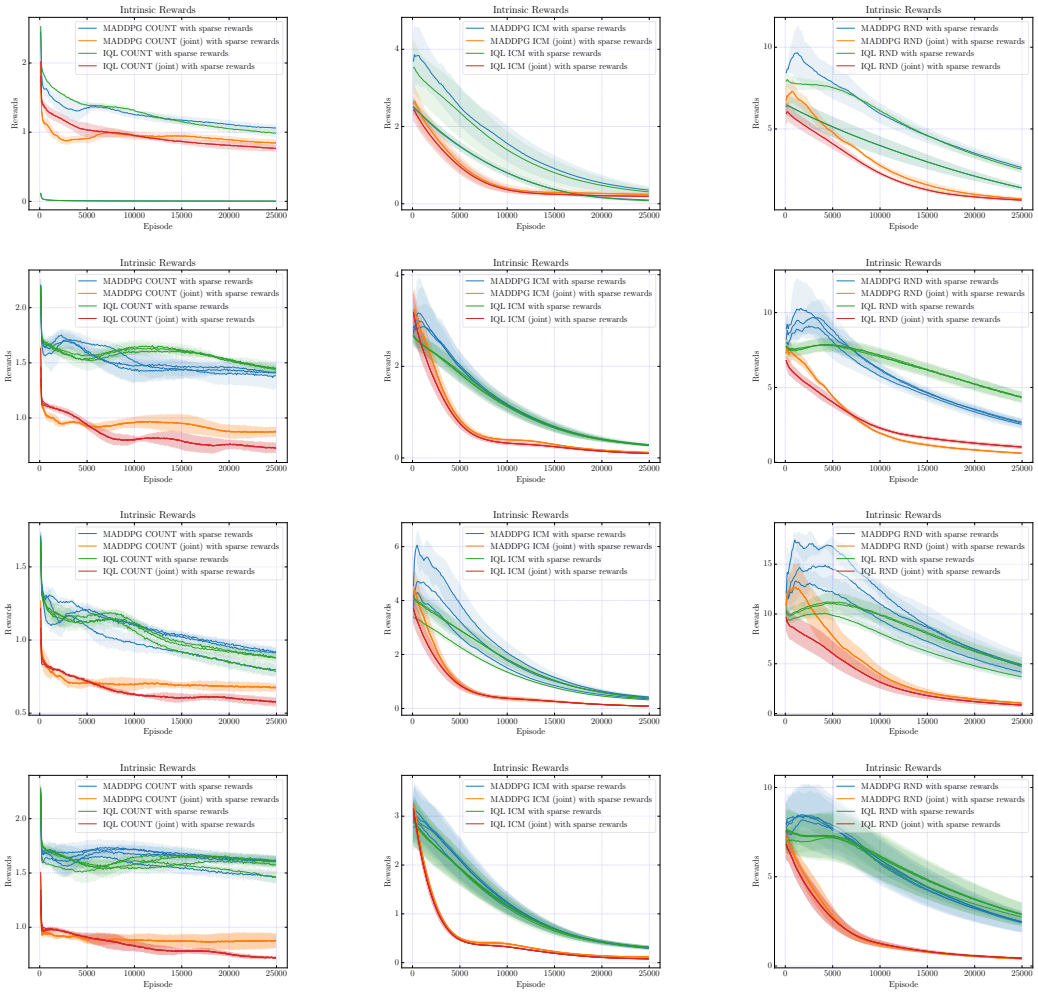

Figure C.23: Intrinsic rewards for MADDPG and IQL with individual and joint curiosities for cooperative communication (1st row), cooperative navigation (2nd row), physical deception (3rd row) and predator prey (4th row) task with sparse rewards.

C.3 Training Time

C.3.1 Multi-Agent Particle Environment

Training time (in s)	MADDPG	IQL
Baseline	1181.35 ± 6.70	1061.44 ± 7.83
COUNT	1236.65 ± 20.41	1131.53 ± 11.71
COUNT (joint)	1236.98 ± 7.39	1156.89 ± 4.51
ICM	1307.24 ± 2.94	1196.75 ± 10.97
ICM (joint)	1311.74 ± 4.32	1207.72 ± 5.70
RND	1251.21 ± 14.22	1150.68 ± 4.96
RND (joint)	1236.57 ± 4.09	1151.78 ± 9.52

Training time (in s)	MADDPG	IQL
Baseline	2035.47 ± 4.13	1909.75 ± 22.34
COUNT	2172.93 ± 21.78	1964.71 ± 16.52
COUNT (joint)	2179.26 ± 20.34	1963.43 ± 19.51
ICM	2251.91 ± 29.74	2073.12 ± 37.87
ICM (joint)	2267.04 ± 23.62	2100.06 ± 16.47
RND	2160.37 ± 20.87	2002.03 ± 26.07
RND (joint)	2172.50 ± 22.52	1986.23 ± 32.44

Training time (in s)	MADDPG	IQL
Baseline	1796.15 ± 18.79	1602.39 ± 12.84
COUNT	1881.22 ± 20.89	1718.93 ± 8.23
COUNT (joint)	1874.70 ± 9.69	1739.01 ± 8.75
ICM	2004.54 ± 20.41	1835.24 ± 8.21
ICM (joint)	1988.79 ± 13.43	1805.95 ± 10.15
RND	1880.30 ± 4.01	1742.24 ± 18.48
RND (joint)	1878.81 ± 15.08	1739.74 ± 16.20

Training time (in s)	MADDPG	IQL
Baseline	2689.98 ± 16.77	2467.60 ± 9.16
COUNT	2839.22 ± 18.36	2562.10 ± 10.55
COUNT (joint)	2829.41 ± 35.68	2563.47 ± 6.70
ICM	2986.11 ± 18.22	2711.24 ± 43.49
ICM (joint)	2929.34 ± 17.64	2683.16 ± 39.37
RND	2869.65 ± 54.09	2524.62 ± 23.56
RND (joint)	2821.58 ± 5.83	2533.52 ± 24.25

Table C.1: Training times for MADDPG and IQL with individual and joint curiosities for cooperative communication (top left), cooperative navigation (top right), physical deception (bottom left) and predator prey (bottom right) task.

C.3.2 Multi-Agent Particle Environment under Partial Observability

Training time (in s)	MADDPG	IQL
Baseline	1250.11 ± 9.87	1149.04 ± 7.73
COUNT	1308.24 ± 23.29	1193.47 ± 6.60
COUNT (joint)	1315.00 ± 8.93	1197.76 ± 11.26
ICM	1366.13 ± 11.60	1257.67 ± 12.09
ICM (joint)	1356.91 ± 11.99	1271.92 ± 19.31
RND	1267.29 ± 25.74	1178.45 ± 6.21
RND (joint)	1273.75 ± 10.09	1176.02 ± 11.22

Training time (in s)	MADDPG	IQL
Baseline	2346.98 ± 41.40	2110.79 ± 21.82
COUNT	2395.66 ± 35.10	2190.82 ± 28.06
COUNT (joint)	2352.71 ± 37.66	2227.06 ± 37.17
ICM	2439.84 ± 15.77	2262.98 ± 30.60
ICM (joint)	2433.60 ± 26.70	2318.65 ± 78.85
RND	2258.50 ± 15.83	2173.27 ± 66.13
RND (joint)	2299.67 ± 57.24	2101.19 ± 15.14

Training time (in s)	MADDPG	IQL
Baseline	1979.58 ± 8.88	1830.87 ± 31.25
COUNT	2058.01 ± 13.36	1926.44 ± 22.44
COUNT (joint)	2076.44 ± 4.51	1897.19 ± 8.60
ICM	2146.44 ± 13.85	1980.83 ± 15.10
ICM (joint)	2138.27 ± 28.79	1972.59 ± 32.82
RND	2049.32 ± 30.45	1868.68 ± 22.24
RND (joint)	2013.28 ± 28.45	1867.21 ± 7.92

Training time (in s)	MADDPG	IQL
Baseline	2918.95 ± 7.45	2679.24 ± 20.32
COUNT	3002.49 ± 30.11	2767.72 ± 37.60
COUNT (joint)	3050.88 ± 25.44	2816.98 ± 24.72
ICM	3150.09 ± 38.47	2907.24 ± 26.08
ICM (joint)	3182.70 ± 50.44	2935.51 ± 24.78
RND	2956.11 ± 27.67	2660.36 ± 0.78
RND (joint)	2901.15 ± 6.21	2669.79 ± 36.82

Table C.2: Training times for MADDPG and IQL with individual and joint curiosities for cooperative communication (top left), cooperative navigation (top right), physical deception (bottom left) and predator prey (bottom right) task under partial observability.

C.3.3 Multi-Agent Particle Environment with Sparse Rewards

Training time (in s)	MADDPG	IQL
Baseline	1222.39 ± 3.97	1114.19 ± 4.19
COUNT	1274.78 ± 20.65	1162.51 ± 20.79
COUNT (joint)	1269.88 ± 8.25	1200.06 ± 9.37
ICM	1318.55 ± 11.67	1214.21 ± 8.46
ICM (joint)	1324.71 ± 4.27	1213.64 ± 6.70
RND	1236.81 ± 3.66	1128.33 ± 8.99
RND (joint)	1239.75 ± 6.67	1146.20 ± 3.45

Training time (in s)	MADDPG	IQL
Baseline	2116.46 ± 22.69	1907.01 ± 25.59
COUNT	2195.79 ± 24.88	2024.31 ± 7.26
COUNT (joint)	2230.98 ± 2.34	2073.29 ± 2.33
ICM	2274.98 ± 4.79	2097.70 ± 36.47
ICM (joint)	2274.54 ± 14.32	2109.50 ± 18.55
RND	2152.70 ± 1.87	2019.10 ± 15.94
RND (joint)	2160.27 ± 32.65	1979.45 ± 48.92

Training time (in s)	MADDPG	IQL
Baseline	1827.27 ± 27.31	1676.32 ± 9.47
COUNT	1915.92 ± 2.89	1749.47 ± 16.57
COUNT (joint)	1950.27 ± 31.87	1747.61 ± 9.03
ICM	1993.05 ± 5.54	1809.85 ± 10.31
ICM (joint)	2020.55 ± 16.29	1830.45 ± 0.87
RND	1907.22 ± 14.95	1736.41 ± 6.78
RND (joint)	1916.91 ± 9.48	1711.12 ± 12.75

Training time (in s)	MADDPG	IQL
Baseline	2788.26 ± 19.32	2546.15 ± 33.27
COUNT	2878.25 ± 10.60	2633.62 ± 52.29
COUNT (joint)	2914.54 ± 44.25	2659.92 ± 40.89
ICM	2965.98 ± 6.59	2764.67 ± 23.20
ICM (joint)	3026.98 ± 20.95	2713.50 ± 9.05
RND	2818.00 ± 13.45	2563.77 ± 11.25
RND (joint)	2863.79 ± 53.19	2587.64 ± 19.80

Table C.3: Training times for MADDPG and IQL with individual and joint curiosities for cooperative communication (top left), cooperative navigation (top right), physical deception (bottom left) and predator prey (bottom right) task with sparse rewards.

Bibliography

Albrecht, S. V. and Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95.

Andoni, A. and Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117.

Babes, M., De Cote, E. M., and Littman, M. L. (2008). Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1389–1392. International Foundation for Autonomous Agents and Multiagent Systems.

Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279.

Berlyne, D. E. (1965). Structure and direction in thinking.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018a). Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018b). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

Böhmer, W., Rashid, T., and Whiteson, S. (2019). Exploration with unreliable intrinsic reward in multi-agent reinforcement learning.

Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM.

Chentanez, N., Barto, A. G., and Singh, S. P. (2005). Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288.

Devlin, S. M. and Kudenko, D. (2012). Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. IFAAMAS.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338.

Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc.

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*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Howard, R. A. (1964). Dynamic programming and markov processes.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Iqbal, S. and Sha, F. (2019). Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning. *arXiv preprint arXiv:1905.12127*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

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

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

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

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*.

Matignon, L., Laurent, G. J., and Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Oliphant, T. (2006–). NumPy: A guide to NumPy. USA: Trelgol Publishing. [Online; accessed August 16, 2019].

OpenAI (2018a). Openai five. `https://blog.openai.com/openai-five/`.

OpenAI (2018b). Reinforcement learning with prediction-based rewards. `https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/`.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. (2017). Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org.

Oudeyer, P.-Y. and Kaplan, F. (2009). What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6.

Oudeyer, P.-Y., Kaplan, F., et al. (2008). How can we define intrinsic motivation. In *Proc. of the 8th Conf. on Epigenetic Robotics*, volume 5, pages 29–31.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017.

Python Core Team (2019). *Python: A dynamic, open source programming language*. Python Software Foundation. Python version 3.7.

Ryan, R. M. and Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67.

Schmidhuber, J. (1991a). Curious model-building control systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pages 1458–1463. IEEE.

Schmidhuber, J. (1991b). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550:354–.

Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

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

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. (2017). # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., Ewalds, T., Horgan, D., Kroiss, M., Danihelka, I., Agapiou, J., Oh, J., Dalibard, V., Choi, D., Sifre, L., Sulsky, Y., Vezhnevets, S., Molloy, J., Cai, T., Budden, D., Paine, T., Gulcehre, C., Wang, Z., Pfaff, T., Pohlen, T., Wu, Y., Yogatama, D., Cohen, J., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Apps, C., Kavukcuoglu, K., Hassabis, D., and Silver, D. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. In *Machine Learning*, pages 279–292.

White, R. W. (1959). Motivation reconsidered: The concept of competence. *Psychological review*, 66(5):297.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.