Feudal Latent Space Exploration for Coordinated Multi-agent Reinforcement Learning

Xiangyu Liu, Ying Tan
Peking University
Beijing, P.R.China
{xiangyu.liu,ytan}@pku.edu.cn

ABSTRACT

We investigate how multiple agents learn to coordinate to form efficient exploration in reinforcement learning. Though straightforward, independent exploration of the joint action space of multiple agents will become exponentially more difficult as the number of agents increases. To tackle this problem, we propose Feudal Latent-space Exploration (FLE) for Multi-agent Reinforcement Learning. FLE introduces a *feudal commander* to learn a low-dimensional global latent structure that instructs multiple agents to explore coordinately. Under this framework, the multi-agent policy gradient is adopted to optimize both the agent policy and latent structure end-to-end. We demonstrate the effectiveness of this method in two multi-agent environments which need explicit coordination. Experimental results validate that FLE outperforms baseline MARL approaches which use independent exploration strategy in terms of mean rewards, efficiency, as well as the expressiveness of coordination policies.

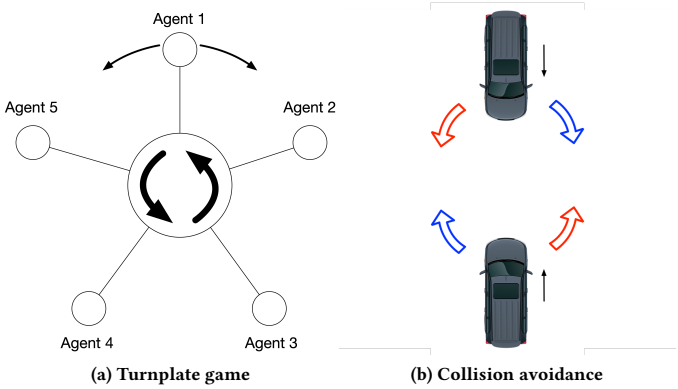
KEYWORDS

Multi-agent Reinforcement Learning; Deep Reinforcement Learning; Exploration; Multi-agent Coordination

1 INTRODUCTION

In recent years, Multi-agent Reinforcement Learning (MARL) has attracted a lot of attention in many application domains, such as multi-player games [9], communication [8], traffic control [25], and display advertising [13]. However, how to discover coordination mode more efficiently remains an open question due to the large exploration space, local optima, or sparse reward signals [5]. This challenge is further exacerbated as the number of agents increases. A common approach for multi-agent exploration is to simply adopt heuristic methods used in single-agent RL, like adding random noise to action space or entropy regularization to encourage abundant exploration. However, pure independent exploration in the original joint state/action space causes the curse of dimensionality in multi-agent systems [5]. For example, consider a multi-agent extension to the *Turnplate* game as illustrated in Figure 1a. In this game, each agent contributes to the joint force of the central turnplate to make it rotate as quickly as possible. The force of each agent should be aligned clockwise or anticlockwise, which is the optimal policy. As the number of agents increases, the exploration space is combinatorially large, making it hard to discover the coordinated motion pattern.

To deal with the issue of complexity, some researchers have attempted to embrace the evolutionary approaches [2], by initializing a population of agent solutions and iterating with evaluation, mutation, and regeneration [23, 34]. Although free to explore all possibilities to obtain a high fitness, evolutionary approaches require large amount of evaluation times. The resulting multi-agent group is also prone to unpredictable behavioral pattern. Besides, it is also difficult to transfer novel exploration strategies in single-agent RL to MARL directly. Typical works include prediction-driven exploration [24], multiple value functions with bootstrapped samples [22], disturbance of model parameters [10], and count-based method [1]. These methods often require auxiliary models which increase the complexity of the training procedure, and are less scalable when extended to multi-agent settings. A more sophisticated research direction is to provide agents with bonus rewards whenever they reach a cooperation state, either by heuristic methods [7] or automatic learning [36]. However, these methods must be given adequate subgoals or extra model-based training, causing additional computational overheads.



(a) Turnplate game (b) Collision avoidance

Figure 1: Two examples of multi-agent tasks demonstrating coordination. (a): To achieve quick rotation, all agents must apply force clockwise or anti-clockwise. (b): To avoid collision between two face-to-face driving cars, an effective coordinated instruction for both agents is to turn left (blue arrows) or turn right (red arrows).

In this paper, we propose a novel approach targeting the exploration challenge in MARL. The key insight behind our approach is that, the coordination mode is often low-dimensional, multi-modal, and with structures shared by the participating agents. To make it clear, another case is illustrated in Figure 1b. Two face-to-face driving cars must make decisions at the same time to avoid collision. The coordination mode in this case is simply dual-modal, either turn left or turn right simultaneously for both cars. Each agent

can share this decision structure and construct the decentralized coordinated motion policy.

Drawing inspiration from these examples, we propose Feudal Latent-space Exploration (FLE) for Multi-agent Reinforcement Learning. We introduce a deep latent model, the *feudal commander*, to learn the low-dimensional and multi-modal coordination structure of multiple agents. The agent policies then use a shared encoding sample from this learned latent space as input, imposing a coordination inductive bias on agents' behavior. The model is fully differentiable and can be trained end-to-end. The size of the exploration space under the proposed approach is constrained by the *feudal commander*, as opposed to the combinatorial large joint action space with vanilla independent exploration strategy. We build the FLE framework upon the popular Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [18]. We have validated our approach on several simulated multi-agent environments [11] that explicitly require coordination. Experimental results show that our method can learn effective diverse coordinated policies, while independent exploration strategies are only able to learn unimodal coordinated policies. We also demonstrate that the proposed FLE is superior to baselines in terms of mean rewards, efficiency, and scalability performance. To summarize, the main contributions in this work are as follows:

- A simple, stochastic latent space exploration structure for expressive multi-agent policy class.
- A promising way to deal with the complex coordination learning problem, especially when the optimal multi-agent policy is combinatorially hard to discover.
- A comprehensive comparison between the proposed method and contemporary approaches through the evaluations on several different multi-agent environments which explicitly require team coordination.

In the reset of the paper, we will firstly present notations and standard methods in Section 2. Our main contribution is introduced in Section 3. Experimental results are in section 4.

2 PRELIMINARIES

2.1 Markov Games

We consider multiple agents operating in a partially-observable stochastic environment, modeled as a partially observable Markov decision process (POMDP). A stochastic game $\mathcal{G}$ is defined by N agents interacting in an environment, with the state $\mathcal{S}$, a set of actions $\{\mathcal{A}_1, ..., \mathcal{A}_N\}$, and a set of observations $\{O_1, ..., O_N\}$. The global state is $s \in \mathcal{S}$. Each agent i has local observation o_i, and uses a policy $\pi_i : O_i \times \mathcal{A}_i \mapsto [0, 1]$ to execute an action a_i to produce next state according to the transition model $\mathcal{P} : \mathcal{S} \times \mathcal{A}_i \times ... \times \mathcal{A}_N \mapsto \mathcal{S}$. Each agent also receives a local reward $r_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$ and a local observation $o_i : \mathcal{S} \mapsto O_i$. Each agent i aims to maximize total expected return of rewards $R_i = \sum_{t=0}^{T} \gamma^t r_{it}$ where γ is discount factor and T is the time horizon.

2.2 Policy Gradient (PG) and Deep Deterministic Policy Gradient (DDPG)

Policy gradient (PG) is a frequently used algorithm for model-free RL [31]. The main idea is to directly optimize the parameter of the policy to maximize the reward objective $J(\theta) = E_{\tau \sim p_\theta(\tau)}[\sum_t r(s_t, a_t)]$, where τ is the trajectory. The gradient of the policy can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)], \tag{1}$$

where $Q^\pi(s, a)$ is the critic to reduce the variance and leads to a series of actor-critic algorithms [28, 30].

Deterministic Policy Gradient (DPG) [29] uses deterministic policies whose parameters θ are adjusted in an off-policy fashion. The exploratory behavior policy is used to perform stochastic gradient ascent. The gradient of $J(\theta)$ is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s, a)|_{a=\pi_\theta(s)}]. \tag{2}$$

Deep Deterministic Policy Gradient (DDPG) [17] is a variant of DPG, where policy π and critic Q are approximated with deep neural networks. Like DQN [21], DDPG uses experience replay and target network. Exploration strategy is treated independently from the learning algorithm, by adding noise sampled from some random process to the actor policy, like Gaussian process and Ornstein-Uhlenbeck process [17].

2.3 Multi-Agent Deep Deterministic Policy Gradients (MADDPG)

Multi-agent deep deterministic policy gradients (MADDPG) [18] is an algorithm for centralised training and decentralized execution of multi-agent systems. It builds deterministic policies as in DDPG for each agent, which are conditioned on each agent's observations. MADDPG alleviates the non-stationary problem associated with the adaption of other agents learning process by introducing a centralised critic for each agent. The gradient of the expected return for agent i with policy π_{θ_i} can be written as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\tau \sim \pi_{\theta_{1,N}}(\tau)}[\nabla_{\theta_i} \pi_{\theta_i}(o_i) \nabla_{a_i} Q_i^\pi(s, a_1, ..., a_N)|_{a_i=\pi_{\theta_i}(o_i)}]. \tag{3}$$

For exploration issue in MADDPG, each agent can only explore locally during the evolving process, which will lead to the curse of dimensionality in exploration space.

3 PROPOSED METHOD

In this section, we propose Feudal Latent-space Exploration (FLE), a framework for MARL to address the major issue outlined in Section 1. As discovering a coordinated policy on joint action space becomes intractable as the number of agents increases, we consider the exploration is driven by the stochastic characteristics of a latent space policy which has a much lower dimension. The latent space structure is shared among all agents. We hypothesize that building a high-level representation will make the exploration easier to master low-dimensional diverse coordination mode. The model computation graph is illustrated in Figure 2.

3.1 Feudal Commander

We begin by introducing a separate *feudal commander* encoder $q_\phi(z|s)$, as illustrated in the left side of Figure 2. The encoder takes in the global state and generates a latent variable $z_g \in \mathbb{R}^n$, where n is the dimension of the latent space and is much smaller than the multi-agent joint action space $\|\mathcal{A}_1 \times ... \times \mathcal{A}_N\|$. In the simplest case, s could be the concatenation of local observations from all

agents, $s = (o_1, ..., o_N)$, but we could also include some global meta information if available. The individual per-agent sub-policies will share this module. The responsibility of coordination relies on this commander, which exerts control over all agents through its provision of the global view during training phase. We will show in Section 4.4 how the commander may coordinate the actions of agents.

We design the latent space to be stochastic. The encoder outputs parameters to $q_\phi(z|s)$, which constructs a Multivariate Gaussian distribution with diagonal covariance:

$$\begin{aligned} \mu_g(s) &= W_\mu \phi(s) + b_\mu \\ log\sigma(s)^2 &= W_\sigma \phi(s) + b_\sigma \end{aligned} \tag{4}$$

The main considerations are as follows: 1) We can sample from this distribution to get stochastic values of the representations, which leads to the exploration of multi-agent policies. 2) Multi-agent tasks may have many coordinated equilibriums (see Figure 1 for examples). Stochastic latent model is beneficial to maintain the expressiveness of the coordination mode. During training, we use the reparameterization trick [15] to learn the latent distribution $q_\phi(z|s)$, by sampling z_g via $\epsilon \sim \mathcal{N}(0, I)$ and network outputs μ, σ:

$$z_g = \mu_g(s) + \sigma_g(s) \odot \epsilon. \tag{5}$$

3.2 Policy Decoder

The actions for each individual agent are independently conditional on both the local observations and the shared latent variable. The policy decoder first queries the z_g and generates a local latent variable z_i by π_{θ_1}, then combines it with the observation o_i to compute the actions by π_{θ_2}:

$$\pi^i_\theta(a_i|o_i, z_g) = \pi_{\theta_2}(a_i|o_i, \pi_{\theta_1}(o_i, z_g)). \tag{6}$$

We emphasize that the exploration in FLE is driven by the stochastic of the policy, which is defined by sampling z_g from the low-dimensional latent space, rather than adding noise process to the joint action space.

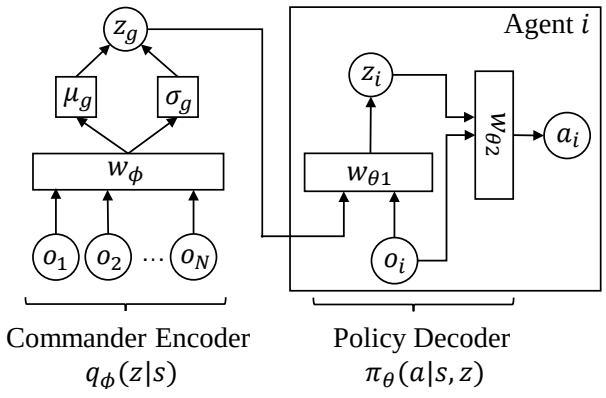


Figure 2: The FLE model computation graph. The global state space is encoded into a latent distribution, from which we sample a latent variable z. Each agent conditions a policy decoder on z and local observation to produce the action.

3.3 Optimization

As our FLE framework requires no additional information, such as the world's dynamics or other prior knowledge of the task, it can be compatible with any MARL algorithm [9, 18, 26]. In this paper, we build the FLE framework on top of MADDPG and try to solve continuous control problem with multi-agent coordination. To highlight the effectiveness of FLE, we disable the exploration strategy (Gaussian or OU process) during training process. In the following, we will show in details how to optimize the whole model. As we do not combine FLE with any other algorithm to conduct the experiments, we frequently refer to FLE-MADDPG simply as FLE.

Actor-Critic Training. We still use the off-policy training by constructing a centralized critic. Note that for each agent i, the introduction of z_g does not influence the evaluation of the critic, as the critic has included all conditional information for generating z_g. Therefore, we can directly apply the off-policy temporal difference to update the Q^π_i function, just as same as MADDPG:

$$\begin{aligned} \mathcal{L}(Q^\pi_i) &= \mathbb{E}_{d\sim\mathcal{D}}[(Q^\pi_i(s, a_1, ..., a_N) - y)^2] \\ y &= r_i + \gamma Q^{\pi'}_i(s', a'_1, ..., a'_N)|_{a'_j=\pi'_j(o'_j,z')} \\ z' &\sim q'_\phi(z|s'), \end{aligned} \tag{7}$$

where $d = (s, a_i, s', r_i)$ is sampled from the experience replay $\mathcal{D}$, and $\{\pi'_j\}, q'_\phi$ are the target policies with delayed parameters $\{\theta'_j\}, \phi'$.

The update of policy actor is also same with MADDPG, except we need to query the value of z_g from the commander module for the sampled data batch:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{d\sim\mathcal{D}} \begin{bmatrix} \nabla_{\theta_i} \pi_{\theta_i}(o_i, z_g) \nabla_{a_i} Q^\pi_i(s, a_1, ..., a_N)| \\ a_i = \pi_{\theta_i}(o_i) \\ z_g = q_\phi(s) \end{bmatrix}. \tag{8}$$

As the commander module can be seen as a part of the actor, we can compute the path derivative from the Q^π_i to the gradients on the commander model's parameters. A regularization item on the stochastic latent variable z_g is added to avoid model collapse, which can positively improve the performance. The prior $p(z)$ is set to be an isotropic unit Gaussian distribution. Thus we construct a regularized KL-divergence, $D_{KL}(q_\phi(z_g|s)||p(z))$, as an additional item of the actor loss:

$$\begin{aligned} &\nabla_\phi J(\phi) = \\ &\frac{1}{N} \sum_i \mathbb{E}_{d\sim\mathcal{D}} \begin{bmatrix} \nabla_\phi q_\phi(z_g|s) \nabla_{z_g} \pi_{\theta_i}(o_i, z_g) \nabla_{a_i} Q^\pi_i(s, a_1, ..., a_N)| \\ a_i = \pi_{\theta_i}(o_i) \\ z_g = q_\phi(s) \end{bmatrix} \\ &- \beta \nabla_\phi D_{KL}(q_\phi(z_g|s)||p(z)), \end{aligned} \tag{9}$$

where we omit time t for abbreviation. During the update, the gradients on ϕ are accumulated and averaged on all local agent's evaluations. To sum up, the FLE-based MADDPG is summarized as Algorithm 1.

3.4 Discussions

Centralized Training and Decentralized Execution. Centralized Training and Decentralized Execution (CTDE) has generated recent interest in Multi-agent Reinforcement Learning community [9, 18, 26], due to the potential for centralized training of ultimately

Algorithm 1 FLE-based MADDPG

for episode = 1 to M **do**
  receive initial state information $x = \{s, o_1, ..., o_N\}$
  for t = 1 to max-episode-length **do**
    sample a_i for each agent based on Eq. (4)(5)(6)
    Executes $a = (a_1, .., a_N)$, observe reward r and next state information $x' = \{s', o'_1, ..., o'_N\}$
    Store (x, a, r, x') to replay buffer $\mathcal{D}$, set $x \leftarrow x'$
    Sample a batch of samples from $\mathcal{D}$
    for agent i to N **do**
      Update critic by minimizing the loss defined in Eq. (7)
      Update actor using the sampled policy gradient defined in Eq. (8)
      Accumulate the gradient on z_g
    end for
    Update commander model with Eq. (9)
    Update target network parameters for each agent i and commander module
  end for
end for

decentralized policies. The centralized module exploits the actions and observations of all agents to aid the training of local policies for each agent.

FLE doesn't quite meet the "decentralized execution", as the *feudal commander* is a function of joint observation or global information, and each local policy must be conditioned on a same code z_g. However, this problem can be partly alleviated by introducing a distributed communication channel for local information exchange (e.g. gossip algorithm [37]) and time-synchronized random seed generator. Similar technique can be found in [27]. But these discussions are beyond the scope of this paper.

Connection to Hierarchical Reinforcement Learning. Our work is partly inspired by the Feudal RL architecture (FRL) [6, 33], and can be viewed as a special case of Hierarchical Reinforcement Learning (HRL). The *feudal commander* imitates the manager and outputs an instruction for each worker to condition their policy on. However in our approach, multiple workers concurrently behavior in an interactive environment, and the "instruction" is dedicated with coordination. Other works in multi-agent RL have also benefited from the HRL ideas [20, 32].

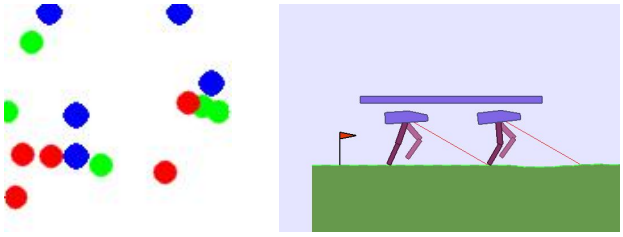
Connection to Variational Autoencoder. The proposed approach has a similar form as the encoder-decoder structure in VAE [15]. However, FLE is trained with policy gradient loss instead of reconstruction loss in standard VAE. So the latent space training does not explicitly encourage summarization of joint observations, but focuses on efficient multi-agent coordination. While during inference phase, FLE samples from the feudal latent posterior. The agents will not be informed of others' observations with the latent sample, but they have learnt how to interpret this sharing code to perform coordination. Some recent works also focus on combining variational methods and multi-agent control, such as cooperative trajectory generation [16].

4 EXPERIMENTS

In our experimental evaluations, we aim to address the following questions: (1) Does the proposed method outperform the baselines which directly add noise to joint action space for exploration? (2) Does the latent space encode meaningful information for coordinated multi-agent decisions? (3) Does it promote the efficiency of exploration in multi-agent learning?

4.1 Environments

We use the *MADRL* environment[1] as a framework for conducting experiments to test the potential of our method. We choose 2 continuous multi-agent control tasks for evaluation: Waterworld and Multi-Walker. We briefly introduce these two environments here. The details of environment settings can be found in [11].



(a) Waterworld (b) Multi-Walker

Figure 3: Two multi-agent environments.

Waterworld. Waterworld is a multi-agent continuous control task, as illustrated in Figure 3a. In this task, multiple agents (in blue) need to coordinate to capture moving food targets (in green) while avoiding poison (in red). Each food target must be collected by all agents, thus an agreement must be achieved between all agents to move towards the same target. In our experiments, we relax the settings by limiting each agent having only 10 range-limited sensors for making distance and velocity measurements of other agents, food targets and poison targets. The food reward was set to 10 and the poison reward was -1.

Multi-Walker. Multi-Walker is a more difficult continuous locomotion task, as shown in Figure 3b. It is based on the BipedalWalker environment from *OpenAI gym* [4]. A package is placed on top of the walkers. The walkers must learn how to move forward and to coordinate with each other to keep the package balanced while navigating a complex terrain. In our experiments, we set the total length of the terrain equal to 20. Dropping the package has a penalty of -100 while moving forward has a reward of 1.

4.2 Baseline Methods and Hyperparameters

All the baseline methods adopted for comparison in this paper are based on DDPG. We test the performance by comparing our method against vanilla MADDPG and parameter-sharing DDPG (PS-DDPG) [11]. In PS-DDPG, all agents share a single policy. But each agent receives different observations, including respective index, which is a completely decentralized learning method.

[1] https://github.com/sisl/MADRL

In all the experiments, we use the Adam optimizer [14] with a learning rate of 0.001 for critics and 0.0001 for actors. The discount factor of reward γ is 0.95. For the soft update of target networks, we use $\tau = 0.01$.

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{10}$$

The networks use RELU for hidden layers. The actor network has two hidden layers $[512, 128]$, and the output layer is the tanh activation function. The critic network has three hidden layers $[1024, 512, 256]$. We initialize all of the model parameters by random normal. The capacity of the replay buffer is $9e5$ and every time we take a minibatch of 1000. For MADDPG and PS-DDPG, we use an OU process with $\theta = 0.15$ and $\sigma = 0.2$ for the exploration noise process. For the commander module in FLE, we set the network structure same with the critic, and the dimension of the latent space is set to 15. We also use the reward normalization to accelerate the training.

4.3 Performance Comparison

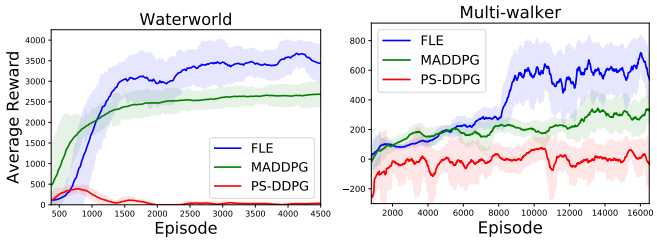


Figure 4: Left: Average rewards on Waterworld with 5 agents. Right: Average rewards on Multi-Walker with 2 agents. Our model (FLE) is competitive in both environments.

We firstly ran experiments with the number of agents set to 5 and 2 for Waterworld and Multi-Walker, respectively. The learning curves are demonstrated in Figure 4, where we measure the average rewards during the training phase. 50 independent statistical runs were used to plot the results and the error bar represents the standard error. It can be seen that our FLE outperforms other baselines by a clear margin. Particularly, we observe in simulation that FLE helps agent groups explore coordinated behaviors more efficiently. For example, in Waterworld, we notice all agents are able to quickly discover the significance of synchronized moving, while the other methods often present unstable oscillation movements, where agents are "pulling" each other. In Multi-Walker, all agents learn to walk forward while keeping consistent paces to steady the package. With the same reward settings, our method can reach the edge of the terrain while MADDPG struggles to carry the package or tends to fall down halfway. The PS-DDPG method performs poorly as each agent's policy changes during training, resulting in a non-stationary environment [5].

Scaling to Many Agents. We next scaled these two tasks to include more agents. We set the number of agent to $[2, 5, 10]$ in Waterworld, while $[2, 3, 4]$ in Multi-Walker. Table 1 shows the performance comparison on the final results on average scores as well as the standard error. With the number of agents increases, FLE scales better and shows lower degradation in average performance as compared to MADDPG and PS-DDPG. Notably, we fixed the dimension of the latent space in FLE across all scaling experiments. This result indicates that when the problem space becomes larger, the behavior of coordination in these tasks exists in a low-dimensional structure. FLE performs exploration in a low-dimensional latent space, therefore scales better and enables more efficient learning.

Not surprisingly, we find FLE shows higher variance than other baselines in Table 1, as well as in Figure 4. On a whole, this high variance mainly comes from the wide variability of coordinated movement choice, which is caused by diverse latent samples generated by FLE. The coordination strategies are often multi-model, and there are many equivalent ways to achieve cooperation (e.g. move towards either target is OK if equal distance). As our method doesn't explicitly deal with the global long-term value, the variable choice may cause the variance of the subsequent scores. Specifically, in Waterworld, we also notice the worst performance of FLE is comparable with MADDPG although high variance. While in Multi-Walker, apart from the variable coordination plans, the high variance also comes from the randomness of the environment. We have also tested the intermediate result during training. Although high variance, the agents expressed high coordination without behaving meaningless.

4.4 Model Inspection

To examine in more details of the model, we now demonstrate empirical evidence suggesting the efficacy and meaningfulness of our approach to coordinated exploration.

Coordinated Exploration vs. Independent Exploration. From the view of exploration structure, our method differs from other MARL baselines, in that the noise is sampled from a latent space and broadcasted to the input of multi-agent policies, rather than adding noise directly to the joint action space. To aid interpretation between coordinated exploration and independent exploration, we compare performance to the same FLE structure but with non-shared noise to each individual policy. Each agent samples z_g concurrently from the commander module. The comparison result is shown in Table 1 (FLE vs. FLE w/o sharing latent).

We can see the FLE without sharing latent variables is inferior to FLE as the number of agents increases, and has a similar performance with MADDPG in Multi-Walker. This highlights the significance of sharing structure for exploration. As the number of agents increases, it becomes harder for multi-agent system to discover an effective action combination if we use independent exploration strategy. For coordinated tasks, the solution space often has low-dimensional structures that can be used to accelerate exploration and training.

Trajectory Embedding during Learning Phase. To visualize the multi-agent exploration during training, we plot the joint multi-agent trajectory embedding in Multi-Walker using T-SNE [19]

# of agents		Final Reward PS-DDPG	MADDPG	FLE	FLE w/o sharing latent
Waterworld	2	508.4 ± 88.9	**3870.6 ± 176.2**	3721.3 ± 187.4	3694.3 ± 180.1
	5	59.2 ± 11.5	2471.9 ± 114.1	**3460.1 ± 205.3**	2671.4 ± 214.6
	10	10.7 ± 8.4	300.2 ± 63.7	**2883.7 ± 146.9**	564.7 ± 53.2
Multi-Walker	2	0.9 ± 38.7	321.6 ± 56.6	**635.4 ± 85.0**	370.9 ± 61.0
	3	-20.1 ± 10.6	-1.4 ± 13.2	**705.9 ± 94.6**	28.4 ± 15.3
	4	-44.3 ± 14.3	-5.8 ± 16.7	**321.0 ± 39.3**	9.6 ± 19.9

Table 1: Scalability Test: Performance comparison for Waterworld and Multi-Walker with different number of agents. Final reward is determined by training until convergence, and evaluating the average reward per episode in the final epoch.

during the training process by running model checkpoints. For clarification, we keep only the first 20 steps of trajectory, as the performance of first 20 steps is crucial for mastering the coordination in Multi-Walker. Naive methods without well-designed coordination strategy often cause an early end of the game (usually around 25 steps), namely the package touches the ground and the game is over. Figure 5 illustrates the results. Each marker represents a $(o_0^t, ..., o_N^t, a_0^t, ..., a_N^t)$ tuple (red markers for FLE, blue markers for MADDPG), and $[\blacktriangledown, \blacksquare, \bullet]$ represents the training episodes, $[0k, 5k, 10k]$. We find that FLE is able to escape from the area of primary stage, and successfully discovers the high reward state space (the upper region). While MADDPG is trapped in a sub-optimal policy space and fails to master coordination, indicating doing exploration in combinatorial action space is very tricky.

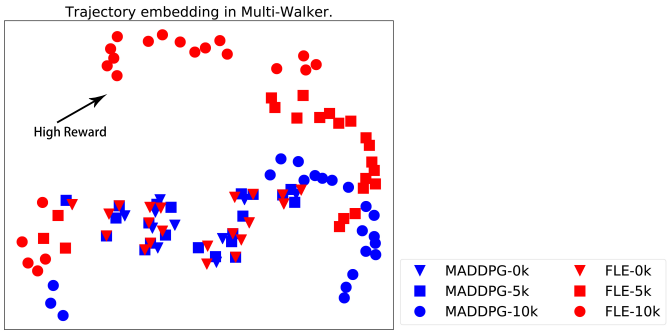
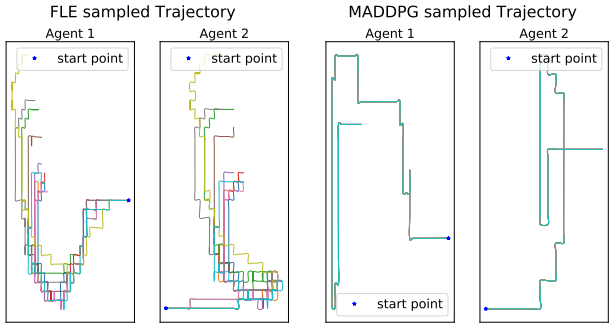


Figure 5: The distribution of first 20 steps multi-agent trajectory in 2-dimensional space using t-sne.

Expressiveness of Coordination. We continue by inspecting the behavior of the latent variable z_g. We conducted an experiment for a simple 2-agents Waterworld task, where the number of food targets is also set to 3. The game ends after all 3 targets are collected. During the test, we fixed the random seed of the environment to verify in a same case. We sample 10 trajectories for FLE and MADDPG respectively by running trained models. The visualization of sampled trajectories is illustrated in Figure 6a and 6b.

We observe that FLE (Figure 6a) is successfully able to capture the wide variability of coordinated movement choice. By sampling from the low-dimensional latent space, two agents share the same coordinated strategy, and the trajectories demonstrate a correlated change between two agents. This is due to the fact that in Waterworld, all agents must move in sync to reach a target. Also, the diverse pattern of trajectories demonstrates that the latent space encodes multi-modal strategies for coordination, as there are many equivalent ways to achieve cooperation (e.g. move towards either target is OK if equal distance). This behavior is also consistent with the high variance performance of FLE demonstrated in Figure 4. On the contrary, MADDPG performs identical trajectories, where 10 samples are overlapped. This is reasonable as the multi-agent policies in MADDPG are deterministic.



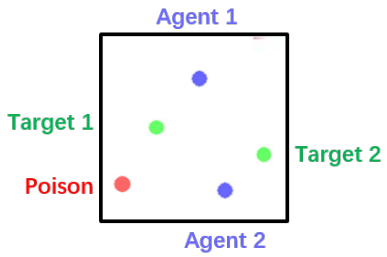
(a) 10 samples of FLE Trajectories for 2 agents.

(b) 10 samples of MADDPG Trajectories for 2 agents.

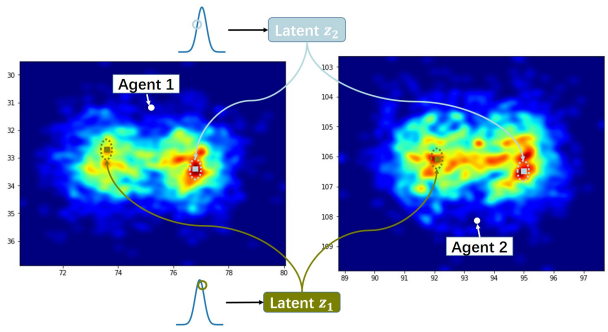
Figure 6: The visualization of sampled trajectories in Waterworld. (In one plot, different colors correspond to different trajectory samples)

To enable a more detailed semantic inspection of the learned coordinated policies, we present a didactic case as shown in Figure 7a. As each food target must be collected by both agents, there are two equivalent solutions, heading towards Target 1 simultaneously, or Target 2. We ran 1000 instances of this fixed environment scene with trained FLE model by repeatedly drawing random samples from feudal commander. Figure 7b shows the heatmap of generated 2-d actions of two agents, namely the agent's position on the next time step. The warm-to-cool color spectrum reflects the dense-to-sparse sample. We notice that the latent sample z_g is exactly significantly correlated with the agents' coordinated actions, with two cliques in the direction of two food targets. Different samples

from the stochastic latent space reflect the variability in coordination mode indicating the latent variables are being utilized on the executed policies.



(a) A Didactic example in Waterworld: two agents are facing almost equivalent cooperation choices.



(b) Heatmap of 1000 sampled actions of two agents in Figure 7a using 1000 shared latent samples generated by feudal commander (left: Agent 1, right: Agent 2). Examples of corresponding real actions of two latent samples z_1, z_2 are also illustrated.

Figure 7: Visualization of how latent samples instruct coordinated behaviors.

5 RELATED WORK

While the field of exploration in single agent RL is popular [1, 10, 22, 24], relatively little work has been done in multi-agent RL. Jaques et al. [12] introduce an intrinsic reward in MARL by giving agents higher reward when its actions lead to relatively higher change in the other agent's behavior. Wang and Wang [35] propose an enhanced prioritized experience replay to help agents explore more efficiently. Böhmer et al. [3] also consider improving the exploration with intrinsic reward and propose a centrally-assisted exploration framework to stabilize learning. These works, although very important, do not address the problem of exploring in a combinatorially large multi-agent action space, and whether these techniques can be improved when scaling to more agents. FLE concentrates on coordinated exploration with a shared latent space, which can be combined with the contributions of these related works.

6 CONCLUSION

In this paper, we propose a Feudal Latent-space Exploration (FLE) method for Multi-agent Reinforcement Learning, by introducing a global *feudal commander* to learn the low-dimensional and multi-modal coordination structure of multiple agents. Each agent shares the same sample from this latent space, and we bias them towards learning to act coordinately. In order to isolate the contribution of our work, we validate our approach on two multi-agent tasks that require explicit coordination, and demonstrate significant improvements over pure independent exploration in MADDPG. Moreover, we also conduct thorough experiments and semantic inspections to demonstrate considerably better scaling, coordination, and expressiveness performance of our method.

ACKNOWLEDGMENTS

This work is supported by Science and Technology Innovation 2030 - "New Generation Artificial Intelligence" Major Project (Grant Nos.: 2018AAA0102301, 2018AAA0100302), and the National Natural Science Foundation of China (Grant Nos.: 61673025, 61375119), and partially supported by National Key Basic Research Development Plan (973 Plan) Project of China (Grant No. 2015CB352302).

REFERENCES

[1] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*. 1471–1479.
[2] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. 2015. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research* 53 (2015), 659–697.
[3] Wendelin Böhmer, Tabish Rashid, and Shimon Whiteson. 2019. Exploration with Unreliable Intrinsic Reward in Multi-Agent Reinforcement Learning. *CoRR* abs/1906.02138 (2019).
[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
[5] Lucian Bu, Robert Babu, Bart De Schutter, et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.
[6] Peter Dayan and Geoffrey E Hinton. 1993. Feudal reinforcement learning. In *Advances in neural information processing systems*. 271–278.
[7] Sam Devlin and Daniel Kudenko. 2016. Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review* 31, 1 (2016), 44–58.
[8] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. 2137–2145.
[9] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[10] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. 2018. Noisy Networks For Exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
[11] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 66–83.
[12] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. 2019. Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*. 3040–3049.
[13] Junqi Jin, Chengru Song, Han Li, Kun Gai, Jun Wang, and Weinan Zhang. 2018. Real-time bidding with multi-agent reinforcement learning in display advertising. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2193–2201.
[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980
[15] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[16] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1995–2003.

[17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. http://arxiv.org/abs/1509.02971

[18] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[19] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[20] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. 2001. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, 246–253.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[22] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*. 4026–4034.

[23] Liviu Panait, R Paul Wiegand, and Sean Luke. 2003. Improving coevolutionary search for optimal multiagent behaviors. In *IJCAI*. Citeseer, 653–660.

[24] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *International Conference on Machine Learning*. 2778–2787.

[25] KJ Prabuchandran, Hemanth Kumar AN, and Shalabh Bhatnagar. 2014. Multi-agent reinforcement learning for traffic signal control. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2529–2534.

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning*. 4292–4301.

[27] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[28] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. http://arxiv.org/abs/1506.02438

[29] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *International Conference on Machine Learning*. 387–395.

[30] Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[31] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[32] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Zhaopeng Meng, Changjie Fan, et al. 2018. Hierarchical Deep Multiagent Reinforcement Learning with Temporal Abstraction. *arXiv preprint arXiv:1809.09332* (2018).

[33] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3540–3549.

[34] Jane X Wang, Edward Hughes, Chrisantha Fernando, Wojciech M Czarnecki, Edgar A Duéñez-Guzmán, and Joel Z Leibo. 2019. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 683–692.

[35] Qisheng Wang and Qichao Wang. 2019. Prioritized Guidance for Efficient Multi-Agent Reinforcement Learning Exploration. *arXiv preprint arXiv:1907.07847* (2019).

[36] Xingyu Wang and Diego Klabjan. 2018. Competitive Multi-agent Inverse Reinforcement Learning with Sub-optimal Demonstrations. In *International Conference on Machine Learning*. 5130–5138.

[37] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. 2007. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing* 67, 1 (2007), 33–46.