REMAX: Relational Representation for Multi-Agent Exploration

Heechang Ryu
Samsung Research
Seoul, Republic of Korea
heechang.ryu21@gmail.com

Hayong Shin
KAIST
Daejeon, Republic of Korea
hyshin@kaist.ac.kr

Jinkyoo Park*
KAIST
Daejeon, Republic of Korea
jinkyoo.park@kaist.ac.kr

ABSTRACT

Training a multi-agent reinforcement learning (MARL) model with a sparse reward is generally difficult because numerous combinations of interactions among agents induce a certain outcome (*i.e.,* success or failure). Earlier studies have tried to resolve this issue by employing an intrinsic reward to induce interactions that are helpful for learning an effective policy. However, this approach requires extensive prior knowledge for designing an intrinsic reward. To train the MARL model effectively without designing the intrinsic reward, we propose a learning-based exploration strategy to generate the initial states of a game. The proposed method adopts a variational graph autoencoder to represent a game state such that (1) the state can be compactly encoded to a latent representation by considering relationships among agents, and (2) the latent representation can be used as an effective input for a coupled surrogate model to predict an exploration score. The proposed method then finds new latent representations that maximize the exploration scores and decodes these representations to generate initial states from which the MARL model starts training in the game and thus experiences novel and rewardable states. We demonstrate that our method improves the training and performance of the MARL model more than the existing exploration methods.

KEYWORDS

Multi-Agent Reinforcement Learning; Multi-Agent Exploration

ACM Reference Format:
Heechang Ryu, Hayong Shin, and Jinkyoo Park*. 2022. REMAX: Relational Representation for Multi-Agent Exploration. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Along with deep neural networks, reinforcement learning (RL) has dramatically improved in the past decades, exceeding human-level performance in challenging games [23, 30, 31]. Learning with the games or physical simulators helps to simulate and solve real-world problems, such as smart grids [9], logistics [6, 36], and distributed vehicles/robots [8, 11, 22]. The advances in RL have naturally generated a significant interest in MARL as the achievements in RL can be extended to more complex problems involving multiple interacting agents. However, the training of a MARL model is difficult even for a simple multi-agent task because it needs to learn implicitly how multiple agents interact with each other along with the environment and how these interactions induce certain task outcomes from only a reward signal. Furthermore, training the MARL model using typical exploration methods that are effective for RL has been unsuccessful because of a large number of possible interactions that the MARL model needs to explore.

Exploration methods employed by researchers for RL can be categorized into three categories: exploration bonus, goal conditioning, and initial state generation. The exploration bonus methods provide an intrinsic reward to an agent if the agent visits a state that has not been visited. Count-based exploration [2, 25, 32], and curiosity-driven exploration [4, 5, 26] are some examples of this method. Goal-conditioning methods manipulate a task's goal in the simulator to ensure that the agent keeps receiving a reward to achieve different goals. These methods virtually endorse the agent the experience of "success" to make training faster using virtual reward signals of success [1, 12]. Finally, initial state generation methods adjust the initial state distribution in the simulator to generate initial states where the agent can easily complete a task and receive a reward [13, 14, 24, 27]. These methods assume that they can arbitrarily reset the agent into any initial state at the beginning of the training episodes in the simulator. The common purpose of these three types of exploration methods is to expose the agent to as many new and rewardable experiences as possible. In particular, goal-conditioning and initial state generation methods are mainly used for goal-oriented tasks where the reward signal is typically sparsely given and depends on whether a goal is achieved or not.

On the contrary, MARL exploration methods are limited in comparison to RL. Most MARL exploration methods focus on designing an intrinsic reward to induce certain collective behaviors of agents that are believed to help solve a multi-agent task or game. For example, intrinsic rewards are provided when one agent's action affects other agents' state transition [3, 35], and when all agents explore only different (novel) or same (rewardable) state space [15]. However, designing a good intrinsic reward is difficult because it requires the prior knowledge of interaction types that can help solve a multi-agent task or game. In addition, designing an effective intrinsic reward often requires an iterative reward shaping procedure until satisfactory performance is achieved.

We herein propose a learning-based exploration method called **RE**lational representation for **M**ulti-**A**gent e**X**ploration (REMAX) to effectively generate novel and rewardable initial states for improving the training of a MARL model without designing an intrinsic reward. As an initial state generation method, REMAX has additional networks that interact with the MARL model to generate initial states in the simulator. These networks consist of two parts: state representation and generation. For state representation, REMAX employs a variational graph autoencoder (VGAE) [18] to extract latent vectors from the game states such that they can be used as an effective input for a coupled surrogate model to predict exploration scores. The exploration score quantifies the balance

*Corresponding author

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

between exploitation and exploration to generate states that are useful for MARL training. In particular, REMAX effectively encodes the states by representing the relationships among agents using an encoder constructed using a graph attention network (GAT) [34] in VGAE. For state generation, REMAX optimizes the surrogate model with respect to a latent vector to find new latent vectors with high exploration scores and decodes them through a VGAE decoder to generate new states. The generated states are then used as the initial states of the training episodes in the simulator to train the MARL model. To summarize, REMAX learns how to optimally generate initial states that are helpful in boosting the MARL model training through the coupled VGAE and surrogate model.

2 RELATED WORK

A representative task with sparse rewards is a goal-oriented task where a binary reward is given only when agents achieve goals (*i.e.,* complete a task). Goal-conditioning and initial state generation methods are mainly used as RL exploration methods for goal-oriented tasks. As one of the goal-conditioning methods, HER [1] virtually exposes an agent to a task's goal by setting the final state of a training episode as a virtual goal in a simulator. Similarly, Goal GAN [12] generates goals with an increasing difficulty using a generator network. Meanwhile, RCG [13], one of the initial state generation methods, adjusts the initial state of a training episode in the simulator to ensure that the probability of an agent reaching a task's goal is within a specific bound. While adjusting the initial state, RCG needs to know at least one state that can reach the goal unconditionally, such as an exact goal state. Some methods perform imitation learning on the predicted state trajectory to obtain a reward [14], or determine the initial state using expert demonstrations on the task [24, 27].

MARL exploration methods for a multi-agent task with sparse rewards, such as a multi-agent goal-oriented task, have not been extensively studied. As one of the possible exploration strategies, GENE [16] has been proposed to generate initial states in the simulator for boosting the exploration of MARL models. This method represents the state of a game as a latent vector using a variational autoencoder (VAE) [17] and estimates the density of the latent vectors using a kernel density estimation (KDE) [10]. By sampling latent vectors from KDE and decoding them, GENE generates new initial states that are believed to be novel and rewardable. However, because GENE trains the state representation module (VAE) and the state density estimation module (KDE) separately, the latent representations may not be the best for generating novel and rewardable initial states. Another critical limitation of the method is that it lacks of considering the relationships among agents. A simple example illustrating this issue is when the relative positions of n homogeneous agents can be represented as $n!$ different states unless a permutation invariance is imposed on the state representation. When focusing on the relative configuration among agents, $n!$ different states can be considered as the same state. It can reduce the state space to be explored and increase the efficiency of sampling states during exploration.

We consider GENE as the main comparison method because GENE and REMAX are both initial state generation methods that can be used for multi-agent goal-oriented tasks without shaping a reward. We also consider additional baseline methods for goal-oriented tasks, such as HER and RCG. However, for a non-goal-oriented task, where there is no goal to be reached, we exclude HER and RCG. Instead, we consider EDTI [35], which uses an intrinsic reward to quantify the influence of an agent's action on the expected returns of other agents. Note that we exclude the MARL exploration methods using the intrinsic reward because they require a user-designed or a domain-specific reward signal. Our study assumes that we can arbitrarily reset agents into any initial state at the beginning of training episodes in the simulator, which can be easily satisfied for the current study that is focusing on offline training using simulation environments.

3 BACKGROUND

3.1 Multi-Agent Reinforcement Learning

We consider a partially observable Markov game [20], which is an extension of the partially observable Markov decision process for a game with multiple agents. A partially observable Markov game for N agents is defined as follows: $s \in \mathcal{S}$ denotes the global state of the game; $o_i \in \mathcal{S} \mapsto O_i$ denotes a local observation correlated with the state that agent i can acquire; and $a_i \in \mathcal{A}_i$ is an action of agent i. The reward for agent i is obtained as a function of state s and joint action $\mathbf{a}$ as $r_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \mapsto \mathbb{R}$. The state evolves to the next state according to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \mapsto \mathcal{S}$. Agent i aims to maximize its discounted return, $R_i = \sum_{t=0}^{T} \gamma^t r_{i,t}$, where $\gamma \in [0,1]$ is a discount factor. In MARL, each agent learns an individual policy that maps the observation to its action to maximize its expected return, which is approximated by a Q-function. While the policy can be deterministic ($a_i = \mu_i(o_i)$) or stochastic ($a_i \sim \pi_i(\cdot|o_i)$), the multi-agent deep deterministic policy gradient (MADDPG) [21] adopts a deterministic policy. MADDPG comprises individual Q-networks and policy networks for each agent. MADDPG lets the Q-network of agent i be trained by minimizing the loss (TD error) as follows: $\mathcal{L}(\varphi_i) = \mathbb{E}_{\mathbf{o},\mathbf{a},r,\mathbf{o}'\sim\mathcal{D}}[(Q_i^{\mu}(\mathbf{o},\mathbf{a};\varphi_i) - y_i)^2], y_i = r_i + \gamma Q_i^{\mu'}(\mathbf{o}',\mathbf{a}';\varphi_i')|_{a_j'=\mu_j'(o_j';\vartheta_j')}$, where $\mathbf{o} = (o_1,\ldots,o_N)$ and $\mathbf{a} = (a_1,\ldots,a_N)$ represent the joint observation and joint action of all agents, respectively. $\mathcal{D}$ is an experience replay buffer that stores $(\mathbf{o},\mathbf{a},r,\mathbf{o}')$ samples obtained from the training episodes. $Q^{\mu'}$ and μ' are target networks for the stable learning of Q and policy networks, respectively. The policy network, $\mu_i(o_i;\vartheta_i)$, of agent i is optimized using the gradient computed as $\mathbb{E}_{\mathbf{o},\mathbf{a}\sim\mathcal{D}}[\nabla_{\vartheta_i} Q_i^{\mu}(\mathbf{o},\mathbf{a};\varphi_i)] = \mathbb{E}_{\mathbf{o},\mathbf{a}\sim\mathcal{D}}[\nabla_{\vartheta_i}\mu_i(o_i;\vartheta_i)\nabla_{a_i} Q_i^{\mu}(\mathbf{o},\mathbf{a};\varphi_i)|_{a_i=\mu_i(o_i;\vartheta_i)}]$.

3.2 Variational Graph Autoencoder

VGAE is an unsupervised representation learning model for graph-structured data. It consists of a graph convolutional network (GCN) [19] encoder and a simple inner product decoder. The GCN encoder performs a posterior inference for latent representations using a node feature matrix and an adjacency matrix of graph-structured data. The decoder performs an inner product between the encoded latent variables to reconstruct the adjacency matrix.

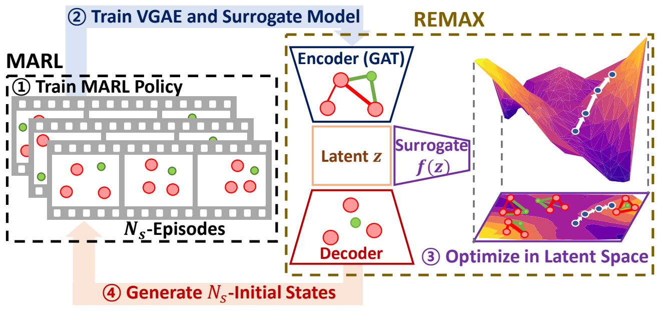

② Train VGAE and Surrogate Model
REMAX
MARL
① Train MARL Policy
Encoder (GAT)
Latent z
Surrogate $f(z)$
N_s-Episodes
Decoder
③ Optimize in Latent Space
④ Generate N_s-Initial States

Figure 1: Overview of REMAX.

3.3 Graph Attention Network

GAT is an effective model for processing graph-structured data. It proposes to compute new node embeddings h_i' for target node i in a graph by aggregating previous node embeddings h_j from neighboring nodes $\{j \in \mathcal{N}_i\}$ that are connected to target node i as $h_i' = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} h_j)$. The attention coefficient $\alpha_{ij} = \mathrm{softmax}_j(e_{ij})$, where $e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j)$, quantifies the importance of node j to node i in computing h_i'. The attention mechanism effectively differentiates the importance of different nodes in updating the embeddings of the target node. In addition, the attention mechanism can be extended to a multi-head attention [33].

4 METHODS

Figure 1 shows how REMAX runs with a coupled MARL model. First, MARL starts training by playing a game in a simulator. When a certain number of states and associated rewards are collected from the game during MARL training, REMAX trains (1) a VGAE to represent the states as latent vectors by considering the relationships among agents and (2) a surrogate model to map the latent vectors into the exploration scores. The VGAE and surrogate model undergo an end-to-end training. Once the training is finished, REMAX gathers a set of optimized latent vectors by maximizing the surrogate model with respect to a latent vector. Finally, REMAX decodes these optimized latent vectors using the trained VGAE decoder to generate new states. MARL then starts the next training episodes from the new initial states generated by REMAX in the simulator and collects states again during training. The newly collected states by MARL are used to retrain REMAX. In other words, the policy training phase of MARL and the surrogate model learning are alternated during training.

4.1 State Representation Using VGAE and Surrogate Model

4.1.1 Encoder of VGAE. We propose GAT-based VGAE to transform states and their latent vectors in both directions (forward and backward) while considering the relationships among agents. Unlike the original VGAE using GCN, we use GAT in the encoder part of VGAE to effectively consider the relationships among agents with the learnable and adjustable relational attentions [21, 28].

First, the GAT encoder reshapes state $s \in \mathbb{R}^{NF}$ as a set of node features $\mathbf{h} = \{h_1, ..., h_N\}$, where N is the number of agents (nodes), and $h_i \in \mathbb{R}^F$ is the local state or observation (node feature) of agent i. The importance of node j to node i is then computed as the attention coefficient $\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{m=1}^{N}\exp(e_{im})}$, where $e_{ij} = \mathrm{LeakyReLU}(\mathbf{v}^T[\mathbf{W}h_i \parallel \mathbf{W}h_j])$. The trainable parameters $\mathbf{W} \in \mathbb{R}^{F' \times F}$ and $\mathbf{v} \in \mathbb{R}^{2F'}$ are shared for all nodes. The computed attention coefficients for every node serve as "soft" edges in a graph representing a state, which means the coefficients softly quantify the level of interactions among agents to be between 0 and 1 in the state.

Employing a multi-head attention with K heads, we compute the updated node embedding $h_i' \in \mathbb{R}^{KF'}$ as $h_i' = \parallel_{k=1}^{K} \mathrm{ReLU}(\sum_{j=1}^{N} \alpha_{ij}^k \mathbf{W}^k h_j)$, where α_{ij}^k and $\mathbf{W}^k$ are the attention coefficients and parameters corresponding to the k-th head attention. In multi-agent settings, using a multi-head attention is beneficial for extracting the distinctly different interactions as there are various types of interactions among agents, especially for non-homogeneous agents.

Once $\mathbf{h}' = \{h_1', ..., h_N'\}$ is computed, it is concatenated as $\parallel_{i=1}^{N} h_i'$. The concatenated node embedding is then used to output the mean μ and standard deviation σ for constructing the distribution for a latent vector z as $z \sim q_\phi(z|s) = \mathcal{N}(\mu, \sigma^2 \mathbf{I})$ with parameters ϕ.

4.1.2 Decoder of VGAE. A multi-layer perceptron (MLP) decoder g_θ is adopted to reconstruct the states $\tilde{s}$ from the latent vector z, *i.e.*, $\tilde{s} \sim p_\theta(s|z)$ or simply $\tilde{s} = g_\theta(z)$ with parameters θ.

4.1.3 Surrogate Model. While VGAE is trained to represent states as latent vectors, an MLP surrogate model $f_\psi(z)$, where z is a latent vector in VGAE, is simultaneously learned to map z to an exploration score $y_s = f_\psi(z)$. The exploration score can be flexibly determined depending on the MARL model. For example, if the MARL model is MADDPG, then the Q and policy networks can be used to define the exploration scores y_s as

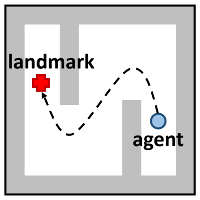
$$\frac{1}{N}\sum_{j=1}^{N}\Big[Q_j(\mathbf{s},\mathbf{a}) + \lambda|Q_j(\mathbf{s},\mathbf{a}) - (r_j + \gamma Q_j'(\mathbf{s}',\mathbf{a}'))|\Big], \tag{1}$$

where $\mathbf{s}$ and $\mathbf{a}$ are, respectively, the joint state and the joint action. In MADDPG, observations and rewards corresponding to the states are obtained from the experience replay buffer. In Equation 1, the first term $\frac{1}{N}\sum_{j=1}^{N} Q_j(\mathbf{s},\mathbf{a})$, which is an empirical mean of Q, quantifies how valuable the corresponding $(\mathbf{s}',\mathbf{a}')$ is for all agents. This term corresponds to the objective of policy networks to maximize their Q-values. Meanwhile, the second term $\frac{1}{N}\sum_{j=1}^{N}|Q_j(\mathbf{s},\mathbf{a}) - (r_j + \gamma Q_j'(\mathbf{s}',\mathbf{a}'))|$, which is an empirical mean of TD error, quantifies how novel the corresponding $(\mathbf{s}',\mathbf{a}')$ is for all agents. This term corresponds to the objective of Q-networks to minimize their TD errors as losses. In other words, the first and second terms respectively evaluate how rewardable (exploitable) and novel (explorable) the state is. It is a combination of exploration using the mean of Q [7] and TD error [29]. The hyper-parameter $\lambda \geq 0$ adjusts the balance between exploitation and exploration.

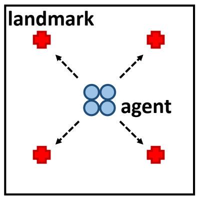
The parameters (θ, ϕ, ψ) of VGAE and the surrogate model are optimized by minimizing the total loss:

$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{VGAE}} + \beta\mathcal{L}^{\text{surrogate}}, \tag{2}$$

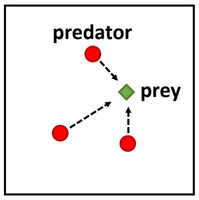
where $\mathcal{L}^{\text{VGAE}}$ is the negative variational lower bound expressed as $-\mathbb{E}_{z \sim q_\phi(z|s)}[\log p_\theta(s|z)] + \mathrm{KL}(q_\phi(z|s) \| p(z))$, where KL is the KL divergence, and $p(z)$ is a prior expressed as $\mathcal{N}(0, \mathbf{I})$. $\mathcal{L}^{\text{surrogate}}$ is



a) Maze



b) Cooperative Navigation



c) Predator-Prey

d) SMAC - 3 marines

Figure 2: Illustrations of the experimental environments.

the mean squared error loss expressed as $\mathbb{E}_{z\sim q_\phi(z|s)}[(f_\psi(z)-y_s)^2]$, where $z \sim q_\phi(z|s)$ is the latent vector encoded from states s. $\beta > 0$ is a hyper-parameter.

4.2 State Generation by Maximizing Surrogate Model

4.2.1 Optimization in Latent Space. After training the surrogate model $f_\psi(z)$ to predict the exploration scores of states based on their latent vectors z, we maximize $f_\psi(z)$ with respect to z to find new latent vectors with high exploration scores.

First, the N_s number of latent vectors $\mathbf{z^0} = \{z_1^0, ..., z_{N_s}^0\}$ are sampled from $\mathcal{N}(0, \mathbf{I})$. Every vector z_i^0 for $i = 1, ..., N_s$ is then optimized using the stochastic gradient ascent.

$$z_i^{t+1} = z_i^t + \delta(\nabla_{z_i^t} f(z_i^t) + \eta_t), \tag{3}$$

where δ is a step size, and η_t is a Gaussian noise with zero mean and decreasing variance. The noise is injected to prevent the latent vectors from being stuck in local optima over a latent space. After L iterations of Equation 3, a set of optimized latent vectors $\mathbf{z}^* = \{z_1^*, ..., z_{N_s}^*\}$ is obtained.

4.2.2 Decoding Latent Vectors. Once $\mathbf{z}^*$ is obtained, it is decoded to a set of new states $\mathbf{s}^* = \{s_1^*, ..., s_{N_s}^*\}$ through the decoder $s_i^* = g_\theta(z_i^*)$ of VGAE. MARL then uses $\mathbf{s}^*$ as the new initial states for the next training episodes in the game.

4.3 Training MARL Policy

MARL policies are then trained while playing the game with the training episodes having the initial states $\mathbf{s}^*$ generated by REMAX. To maintain a certain level of default initial states, MARL uses the initial states generated from REMAX and the default initial states provided by the game with a certain probability, such as 0.8:0.2. After training MARL using N_s training episodes, REMAX is trained again using newly collected states from MARL and generates new initial states for training MARL in the next round. In other words, for every N_s number of training episodes, REMAX is retrained and generates new initial states for training MARL. The parameters (θ, ϕ, ψ) of REMAX are reinitialized and retrained using only newly collected states from just previous training episodes to reflect the evolution of MARL. This is similar to employing curriculum learning in that the game condition is dynamically being adjusted depending on the current performance of the intermediate policy that is being trained.

Table 1: Number of training episodes in maze.

	Latent space dim.	Episode(×10)
Random	-	640± 223
HER	-	601± 227
RCG	-	574± 198
GENE	1	559±255
REMAX	1	**456**±202
REMAX	2	495±139

5 EXPERIMENTS

Figure 2 shows the environments (games) used to evaluate the performances of the proposed and comparison methods. The environments are designed to render more sparse rewards than existing environments [21, 28]. We assume that the agents in the environments have the positions and velocities (or health) of all agents as a state. In the environments, we consider random exploration and GENE as main comparison methods.

- **Goal-oriented tasks (a) and (b)**: In single-and multi-agent goal-oriented tasks, we additionally consider HER and RCG for goal-oriented tasks.
- **Mixed cooperative-competitive games (c) and (d):** We exclude HER and RCG because they are not suitable for non-goal-oriented tasks. Instead, we additionally consider EDTI, which uses intrinsic rewards that quantify the influence of one agent on others.

We use MADDPG as the MARL model. Note that any MARL model can be used; however, only MADDPG, one of the most general MARL models, is used as a representative in this study because the focus of this study is on investigating the effectiveness of exploration methods, irrespective of the MARL model used. In addition, all results in this study were averaged across five different seeds.

5.1 Maze

Maze, shown in Figure 2 (a), is a single-agent environment which requires an agent (blue circle) to search for a landmark (red cross). The agent obtains +1 as a reward when it reaches the landmark, and 0 otherwise. Because there is only one agent in this environment, considering the relationships among agents is unnecessary. Thus, REMAX has VGAE without GAT, which is similar to VAE. Although both REMAX and GENE use VAE, they are different in that REMAX uses the surrogate model while GENE uses KDE.

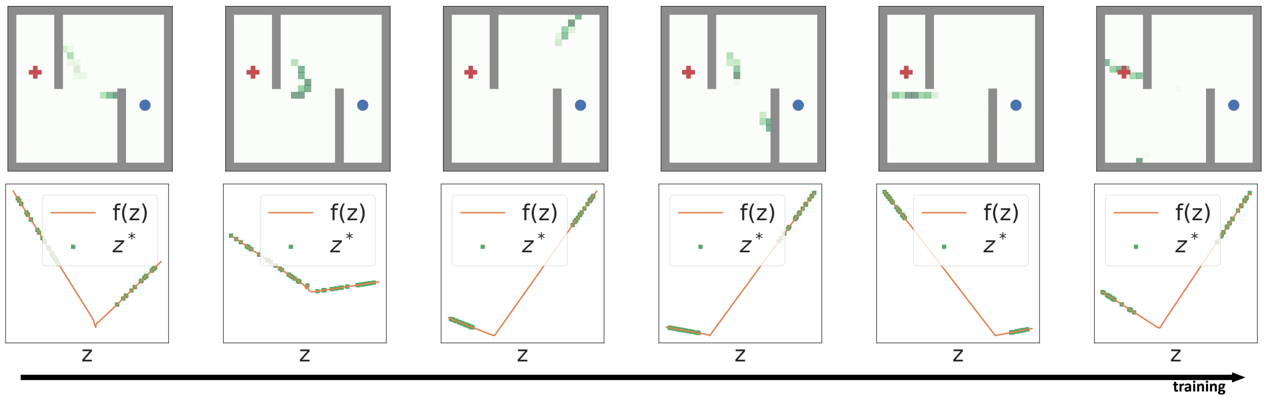


Figure 3: Generated states and surrogate models with one-dimensional latent space in maze.

Table 2: Number of training episodes (×10) in cooperative navigation. In the table, ">2000" indicates that the policy cannot be learned to complete the task by the corresponding method within the given number of training episodes.

	Additional architecture				Number of agents				
	VAE	KDE	GAT	Surrogate	2	3	4	6	8
Random					583± 271	834± 168	1178± 542	>2000	>2000
HER					489± 140	790± 217	1071± 618	>2000	>2000
RCG					353± 133	674± 309	877± 679	>2000	>2000
GENE	✓	✓			342± 145	632± 265	742± 540	1401± 625	>2000
GENE with GAT	✓	✓	✓		177± 64	400±256	454±310	920± 551	1204± 698
REMAX	✓		✓	✓	**161**± 69	**272**±80	**331**±114	**602** ± 287	**908** ± 601

Table 1 presents the number of training episodes required for training a policy to complete the task. Here, completing a task is defined as achieving ten consecutive successes, starting from the initial states provided by the environment. Thus, a smaller number in the table means that fewer training episodes are needed to train the successful policy. The latent space dimension denotes the dimension of the latent vector z. GENE is reported as having the best performance with a one-dimensional latent space. The table shows that REMAX requires fewer episodes than HER (a goal-conditioning method) and RCG, although it does not need to specify the goal state during training. Note that to train and run RCG, one needs to specify the goal state. We believe that REMAX performs better than GENE because it trains the VAE and the surrogate model together in an end-to-end learning, while GENE separately trains VAE and KDE.

5.1.1 Analysis of Generating States. The first-row plots in Figure 3 show the distribution of generated states $\mathbf{s}^*$ as green dots, and the second-row plots show how the surrogate model $f_\psi(z)$ and the optimized latent vectors $\mathbf{z}^*$ change with the training's progress. These states $\mathbf{s}^*$ are generated by decoding $\mathbf{z}^*$. As MARL and REMAX are trained with more samples, as shown in the figures, REMAX tends to generate states near the landmark because it learns that these states are easily rewardable and helpful for MARL training.

5.2 Cooperative Navigation

Cooperative navigation shown in Figure 2 (b) is a multi-agent environment where homogeneous agents (blue circles) are required to be positioned at landmarks (red crosses) at the four corners. There are as many landmarks as there are agents. Each agent obtains a reward +1 when every landmark is occupied by one agent, and 0 otherwise. Thus, all agents should be coordinated to occupy a distinct landmark and thus receive the reward.

5.2.1 Effectiveness of the REMAX Structure. We conducted ablation experiments to evaluate the synergistic effects of VGAE and the surrogate model in REMAX, and the results are summarized in Table 2. The table presents the number of training episodes required for training a policy to complete the task. In the table, GENE with GAT indicates GENE whose VAE module is replaced with VGAE with the GAT encoder. Compared with GENE, adding GAT to GENE reduces the number of training episodes, as indicated in the table. Because REMAX requires fewer episodes than the other methods, regardless of the number of agents, the use of both VGAE and the surrogate model is validated as being effective for training the MARL model, especially when the number of agents is large. In addition, the number of episodes for REMAX increases more slowly with the number of agents, which implies that REMAX enables MARL to train the policy scalably even when the number of agents increases. Meanwhile, HER and RCG require a rapidly increasing number of episodes for training the policy with an increasing number of

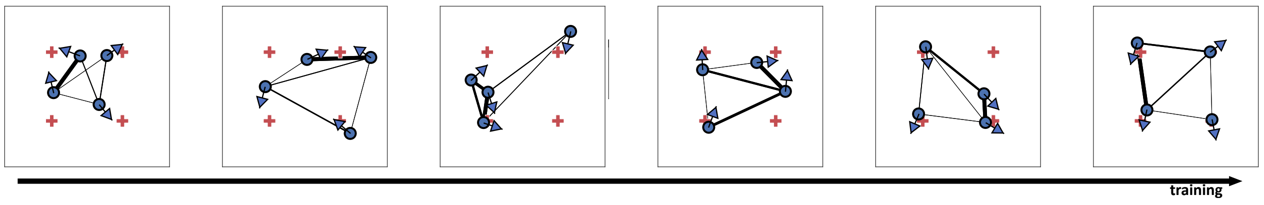


Figure 4: Attention of GAT in REMAX in cooperative navigation with 4 agents.

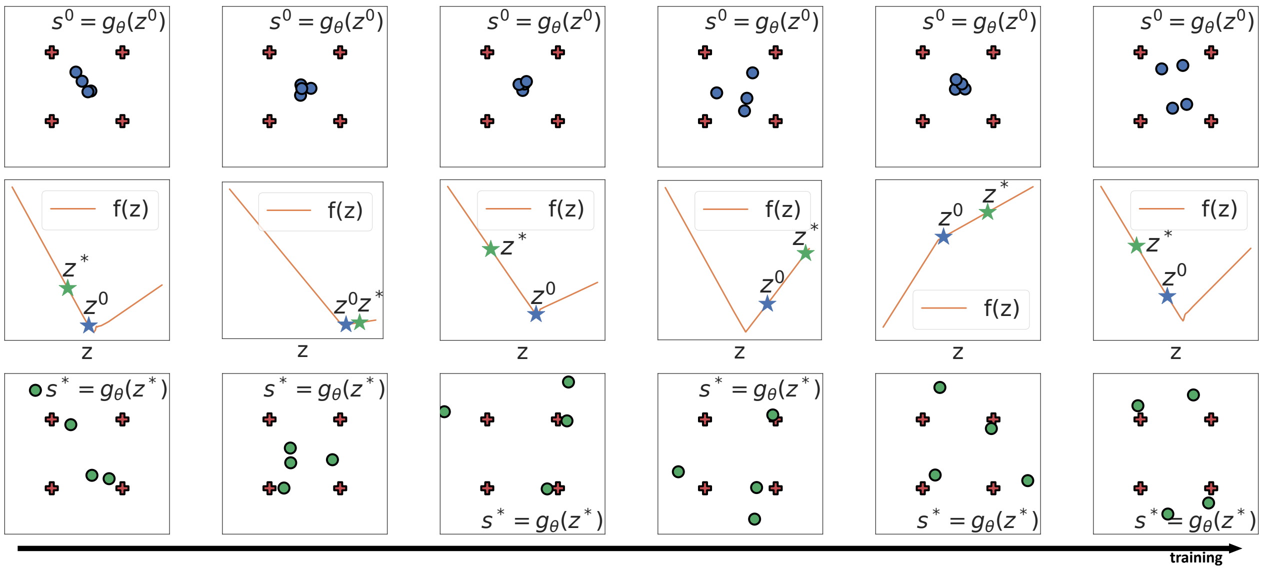


Figure 5: Surrogate models and generated states in cooperative navigation with 4 agents.

Table 3: Number of training episodes (×10) according to types of exploration scores for the surrogate model of REMAX. $f_i(s)$ is the inverse of state visit count of agent i and $m(s) = \frac{1}{N}\sum_{j=1}^{N} f_j(s)$.

Naive reward r	Intrinsic reward: Burrowing $\sum_{i=1}^{N} f_i(s)\mathbf{1}[f_i(s) > m(s)]$	Intrinsic reward: Covering $\sum_{i=1}^{N} f_i(s)\mathbf{1}[f_i(s) < m(s)]$	Proposed Equation 1
471± 193	392± 159	338± 126	**161**± 69

agents, and can no longer train the policy when the number of agents exceeds 6.

5.2.2 Analysis of Representing/Generating States. Figure 4 shows the normalized attention coefficients of GAT in the VGAE encoder in REMAX; the thicker the line is, the larger the coefficient is. The coefficients between agents increase as multiple agents with their actions (blue arrows) approach a common landmark. This trend can be observed in the second figure of Figure 4, which shows the two agents approaching the upper-right landmark, and the fifth figure of Figure 4, which shows the two agents approaching the lower-right landmark.

Figure 5 compares the two types of states: s^0 decoded from an initial latent vector z^0 and s^* decoded from an optimized latent vector z^*. The second-row figures show z^0 and z^* over $f_\psi(z)$. In addition, the first-and third-row figures show the decoded states, s^0 and s^*, from z^0 and z^* using g_θ. As MARL and REMAX are trained with more samples, as shown in the figure, REMAX tends to generate states near the landmarks, enabling the agents to easily complete the task.

5.2.3 Effectiveness of the REMAX Exploration Score. Table 3 presents the number of training episodes according to the types of exploration scores for the surrogate model of REMAX in a cooperative navigation with two agents. In the table, r is the case where the score is replaced with agents' naive rewards corresponding to states. Burrowing and Covering are reward signals designed empirically to minimize and maximize the inverses of state visit counts of agents, which are similar to intrinsic rewards [15], thus boosting exploitation and exploration, respectively. In the table, Equation 1 requires fewer episodes than the other types of scores. This may be because the naive reward is too sparse to be used as a score, and

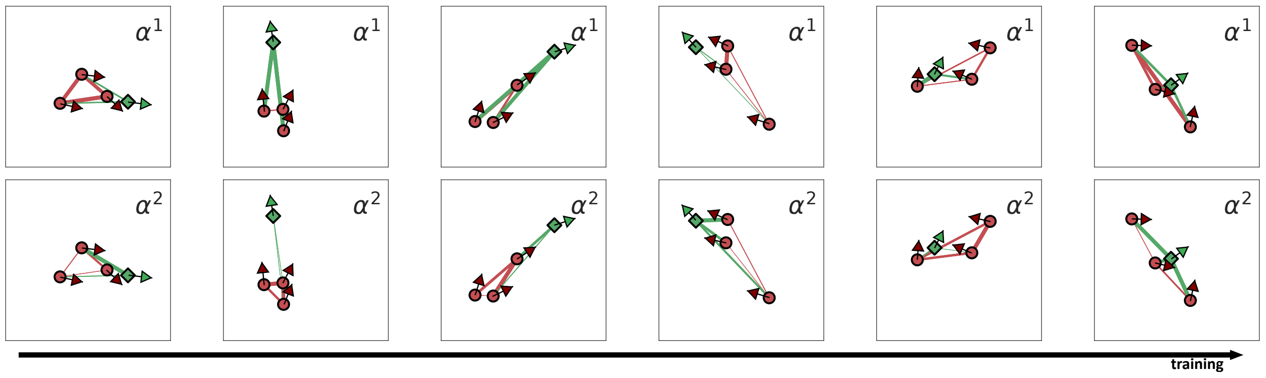
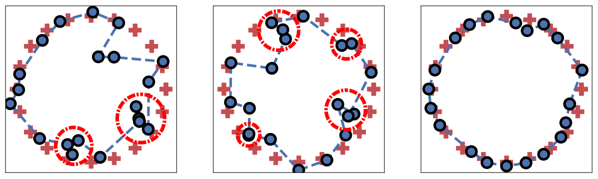


Figure 7: Multi-head attention of GAT in REMAX in predator-prey.



a) Random b) GENE c) REMAX

Figure 6: Occupation in cooperative navigation with 20 agents.

Table 4: Occupation rate in cooperative navigation.

	Random	GENE	REMAX
10 agents	0.71± 0.23	0.82± 0.21	**0.96**± 0.05
20 agents	0.55± 0.19	0.63± 0.23	**0.88**± 0.12
30 agents	0.42± 0.17	0.51± 0.21	**0.81**± 0.14

the others are designed only to induce specific behaviors of agents, while Equation 1 balances exploitation and exploration and boosts MARL training. The balance of Equation 1 changes according to the hyper-parameter λ, and it is discussed in the appendix.

5.2.4 Scalability Test. Table 4 presents the rate of occupied landmarks in cooperative navigation with more agents. In the table, REMAX has higher occupation rates than the other methods, which implies that the agents trained by REMAX cooperate effectively to occupy distinct landmarks, even when their numbers are large. Figure 6 shows how 20 agents occupy landmarks positioned in a circular shape after trained by each method. In the figure, the agents in Random and GENE tend to cluster (dotted red circles) at some landmarks, while the agents in REMAX occupy distinct landmarks.

5.3 Predator-Prey

As shown in Figure 2 (c), predator-prey is a multi-agent environment where three homogeneous predator agents (red circles) aim to capture one prey agent (green diamond), which is called 3 vs. 1 predator-prey. Because the prey has a faster speed and acceleration than predators, the predators are required to cooperate for capturing the prey. In the 3 vs. 1 game, each predator obtains a reward +10 when two or more predators capture the prey simultaneously, and 0 otherwise. Meanwhile, the prey obtains a reward -10 for getting captured. If there are 6 predators and 2 preys, it is denoted as 6 vs. 2 predator-prey, and the rewards are obtained when three or more predators capture the prey simultaneously. 9 vs. 3 predator-prey is also similar to 6 vs. 2 predator-prey. In predator-prey, we used the GAT encoder with a $K = 2$ multi-head attention for REMAX because there are at least two types of relationships, predator-predator and predator-prey. We exclude HER and RCG as they are not suitable for a non-goal-oriented task. Instead, we consider EDTI, which uses intrinsic rewards quantifying the influences of one agent's action on other agents' rewards and transitions.

Table 5: Returns of predators in predator-prey.

	Random	EDTI	GENE	REMAX
3 vs. 1	1.6± 0.4	3.7± 1.0	14.3± 4.9	**18.7**± 4.6
6 vs. 2	1.8± 0.6	2.8± 0.8	8.3± 1.8	**13.5**± 4.4
9 vs. 3	2.6± 0.7	3.2± 1.1	9.1± 2.7	**14.4**± 4.1

Table 5 compares the returns of the predators. After the predators and prey are trained together by each method, the predators are validated for 200 test episodes against the prey trained by a random exploration. In the table, REMAX has higher returns than the other methods, which implies that the predators trained by REMAX cooperate effectively to capture the prey. Meanwhile, EDTI ends up achieving significantly lower returns than GENE and REMAX. These lower returns may be because EDTI cannot accurately predict the influences of intrinsic rewards. Accurate prediction requires many samples, especially when there are many agents in a continuous state-action space.

5.3.1 Analysis of Representing States with Multi-Head Attention. Figure 7 shows two sets of the normalized attention coefficients, α^1 and α^2, of the GAT encoder with a $K = 2$ multi-head attention in REMAX. The two sets of coefficients can quantify the levels of cooperation among predators and the competition between predators and prey, respectively (upper and lower figures). In the figures, the coefficients (the thickness of the line) increase when two predators

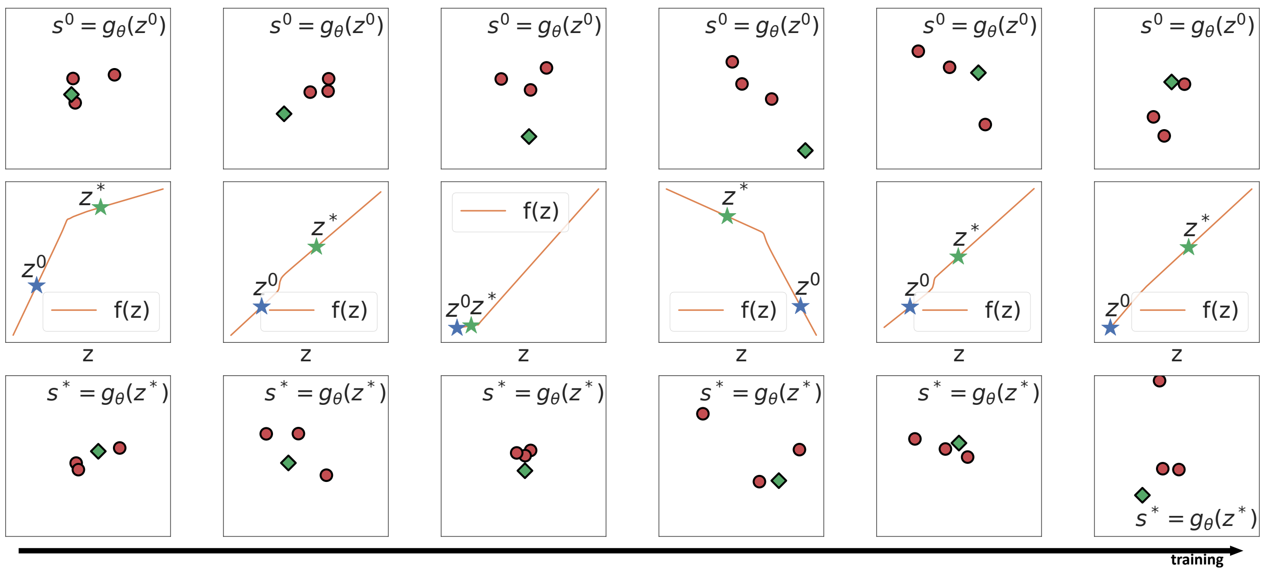


Figure 8: Surrogate models and generated states in predator-prey.

Table 6: Winning rates of controlled marines in SMAC.

	Random	EDTI	GENE	REMAX
3 vs. 3	0.86± 0.05	0.88± 0.06	0.94± 0.08	**1.00**± 0.00
5 vs. 6	0.00± 0.00	0.00± 0.00	0.10± 0.04	**0.43**± 0.11

with their actions (red arrows) approach the prey with its action (green arrows) simultaneously. For example, in the fourth figure of Figure 7, two predators relate to a strong attention (upper figure), and, at the same time, both predators are connected strongly with the prey they are about to capture together (lower figure). We identified that the two predators connected by larger attention coefficients tend to cooperate to capture the prey.

5.3.2 Analysis of Generating States. Figure 8 compares the two types of states: s^0 decoded from an initial latent vector z^0 and s^* decoded from an optimized latent vector z^*. The second-row figures show z^0 and z^* over $f_\psi(z)$. In addition, the first-and third-row figures show the decoded states, s^0 and s^*, from z^0 and z^* using g_θ. As MARL and REMAX are trained with more samples, as shown in the figure, REMAX tends to generate states where the predators are located more closely to the prey or surround the prey. This allows the predators to easily capture the prey and thus receive a reward, which expedites the MARL training.

5.4 Starcraft Multi-Agent Challenge (SMAC)

SMAC [28] is a more realistic multi-agent environment, as shown in Figure 2 (d), where three marines (agents) aim to fight against equivalent marines of game AI. SMAC originally has dense reward signals. However, we simply modify the SMAC's rewards to be more sparse at the final timestep of each episode: a victory reward 1, while a defeat -1, with an attack reward 1 to encourage attacking the enemies, rather than running away.

Table 6 presents winning rates of controlled marines against game-AI marines in SMAC. In the table, REMAX outperforms other methods with higher winning rates.

6 CONCLUSIONS

We proposed REMAX, an exploration method that generates initial states for accelerating the training of a MARL model. Empirically, we demonstrated that REMAX generates states by representing relationships among agents, and the generated states improve the training and performance of the MARL model more than existing exploration methods.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in neural information processing systems*. 5048–5058.

[2] 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.

[3] Wendelin Böhmer, Tabish Rashid, and Shimon Whiteson. 2019. Exploration with unreliable intrinsic reward in multi-agent reinforcement learning. *arXiv preprint arXiv:1906.02138* (2019).

[4] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. 2019. Large-Scale Study of Curiosity-Driven Learning. *International Conference on Learning Representations* (2019).

[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. *International Conference on Learning Representations* (2019).

[6] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2013), 427–438.

[7] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. 2017. UCB exploration via Q-ensembles. *arXiv preprint arXiv:1706.01502* (2017).

[8] Peter Corke, Ron Peterson, and Daniela Rus. 2005. Networked robots: Flying robot navigation using a sensor net. In *Robotics research. The eleventh international symposium*. Springer, 234–243.

[9] Emiliano Dall'Anese, Hao Zhu, and Georgios B Giannakis. 2013. Distributed Optimal Power Flow for Smart Microgrids. *IEEE Trans. Smart Grid* 4, 3 (2013), 1464–1475.

[10] Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. 2011. Remarks on some nonparametric estimates of a density function. In *Selected Works of Murray Rosenblatt*. Springer, 95–100.

[11] J Alexander Fax and Richard M Murray. 2004. Information flow and cooperative control of vehicle formations. *IEEE transactions on automatic control* 49, 9 (2004), 1465–1476.

[12] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. 2018. Automatic Goal Generation for Reinforcement Learning Agents. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 1515–1528.

[13] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse Curriculum Generation for Reinforcement Learning. In *Proceedings of the 1st Annual Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 78)*. PMLR, 482–495.

[14] Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. 2019. Recall traces: Backtracking models for efficient reinforcement learning. *International Conference on Learning Representations* (2019).

[15] Shariq Iqbal and Fei Sha. 2019. Coordinated Exploration via Intrinsic Rewards for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1905.12127* (2019).

[16] Jiechuan Jiang and Zongqing Lu. 2020. Generative Exploration and Exploitation. In *Thirty-Fourth AAAI conference on artificial intelligence*.

[17] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *International Conference on Learning Representations* (2014).

[18] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[19] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations* (2017).

[20] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.

[21] Ryan Lowe, Yi I 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.

[22] Laëtitia Matignon, Laurent Jeanpierre, Abdel-Illah Mouaddib, et al. 2012. Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Processes.. In *AAAI*.

[23] 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.

[24] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6292–6299.

[25] Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2721–2730.

[26] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 16–17.

[27] Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. 2018. Backplay:" Man muss immer umkehren". *arXiv preprint arXiv:1807.06919* (2018).

[28] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, 2186–2188.

[29] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. *International Conference on Learning Representations* (2016).

[30] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

[32] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*. 2753–2762.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[35] Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. 2020. Influence-Based Multi-Agent Exploration. *International Conference on Learning Representations* (2020).

[36] Wang Ying and Sang Dayong. 2005. Multi-agent framework for third party logistics in E-commerce. *Expert Systems with Applications* 29, 2 (2005), 431–436.