Improving Generalization of Multi-agent Reinforcement Learning Through Domain-Invariant Feature Extraction

Yifan Xu[1,2], Zhiqiang Pu[1,2(✉)], Qiang Cai[1,2], Feimo Li[1], and Xinghua Chai[3]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
zhiqiang.pu@ia.ac.cn
[3] The 54th Research Institute of China Electronics Technology Group Corporation, Beijing, China

Abstract. The limited generalization ability of reinforcement learning constrains its potential applications, particularly in complex scenarios such as multi-agent systems. To overcome this limitation and enhance the generalization capability of MARL algorithms, this paper proposes a three-stage method that integrates domain randomization and domain adaptation to extract effective features for policy learning. Specifically, the first stage samples environments provided for training and testing in the following stages using domain randomization. The second stage pretrains a domain-invariant feature extractor (DIFE) which employs cycle consistency to disentangle domain-invariant and domain-specific features. The third stage utilizes DIFE for policy learning. Experimental results in MPE tasks demonstrate that our approach yields better performance and generalization ability. Meanwhile, the features captured by DIFE are more interpretable for subsequent policy learning in visualization analysis.

Keywords: multi-agent reinforcement learning · domain adaptation · domain randomization

1 Introduction

As reinforcement learning (RL) continues to achieve success in games [13,18,24], researchers are currently working on broader applications [14,16,20]. Implementation of RL in complex tasks like applications in real world demands algorithms that can adapt to variations, unlike tasks with static environments, for example, board games [3] and video games [24]. However, traditional RL methods struggle to maintain performance when faced with even slight differences between training and testing environments, particularly in complex and dynamic multi-agent settings. To improve agents' generalization ability, domain randomization (DR) has emerged as a common method [15,17,21]. DR uses a distribution,

L. Iliadis et al. (Eds.): ICANN 2023, LNCS 14259, pp. 49–62, 2023.
https://doi.org/10.1007/978-3-031-44223-0_5

rather than fixed parameters, to model environment dynamics, inducing uncertainty during training processes. Ranging from image-based observations [17] to internal dynamics [14], all accessible parameters are available for randomization. Tobin [21] and Sadeghi [17] first implement DR in robotics by randomizing visual variants in input images under simulation where various training environments improve object detection accuracy when transferring the policy to reality. Peng [15] randomizes dynamic variations of a robotics arm in push tasks and shows that memory-based models achieve high performance than memoryless models through experiments. Chen [2] fills the blank of theoretical analysis in DR by deriving the upper bound of the gap between optimal policies in simulation and reality. In the assumptions of [2], obtaining the optimal policy in DR is formulated as solving a latent Markov decision process (LMDP), where an LDMP contains a series of MDPs. The sample complexity of policy optimization in LMDP grows exponentially with the number of MDPs in LMDP [9]. Therefore, sample-efficient algorithms for training policies in DR are in great demand. OpenAI [14] proposes automatic curriculum training to improve sample efficiency. However, a delicately crafted curriculum evaluation protocol is needed to guarantee that algorithms learn progressively according to task difficulty. Mandlerkar [12] creates an adversary policy to sample environments where current policies fail. However, the adversarial approach leads to learning pessimistic samples and the training process is likely to be unstable. In general, optimizing RL policy in domain randomization remains intractable.

In the context of multi-agent reinforcement learning (MARL), the inherent uncertainty of the environment is exacerbated due to dynamic multi-agent systems. The number of agents and agents' configurations can vary greatly among different environments, leading to variations in agent interactions and making policy transfer even more challenging.

In order to improve generalization performance of RL, it is essential that the optimal policy in the LMDP and the optimal policies in individual MDPs exhibit similarities [6]. To effectively learn policies in domain randomization, we introduce domain adaptation (DA) into our methodology, thereby leveraging this similarity. Initially used for style migration in image classification tasks, DA helps to discover shared feature spaces across different image classes and filter out irrelevant features [10]. Methods in DA contain discrepancy-based approaches [11,22], adversarial-based approaches [4,7], and reconstruction-based approaches [1] [5]. The discrepancy-based approaches focus on reducing the distributional discrepancy between the source and target domains. This discrepancy is often measured using distance metrics such as the maximum mean discrepancy (MMD) [22], second-order moment, or k-oder moment [19]. However, these metrics only accurately capture differences under specific assumptions, which can lead to suboptimal adaptation performance. Besides, labeled target domain data is required to compute the distance, which is costly in some scenarios. The adversarial-based approaches learn domain-invariant representations by training a domain discriminator to distinguish between the source and target domains [4]. The features are extracted in an unsupervised manner, without requiring labeled

target domain data [23]. However, training the discriminator to accurately distinguish between the source and target domains requires lots of data and may cause unstable training [26]. The reconstruction-based approaches learn domain-invariant representations by reconstructing the input data from the learned features. These methods leverage the idea that a good domain-invariant representation should preserve the essential information needed to reconstruct the input data while removing the domain-specific information. Bousmalis [1] learns disentangled domain-invariant and domain-specific features through shared and private encoders, respectively. Xing [25] uses cycle consistency to learn feature disentanglement in autonomous driving. In RL simulations, it is convenient to collect a vast amount of unlabeled data, which makes the use of reconstruction models a promising option for feature extraction.

Therefore, our work proposes a method that uses DA to learn a latent representation across a series of randomized environments in multi-agent scenarios. We train a module called domain-invariant feature extractor (DIFE) in domains with randomized parameters and use a cycle-consistent variational autoencoder (cycle-consistent VAE) [8] to disentangle general and specific feature embeddings. This allows agents to utilize domain information in downstream policy learning.

Experiments are conducted in a series of multi-agent coverage tasks with different parameter configurations. Results show that our approach yields better performance and generalization ability. Meanwhile, the features captured by DIFE are more interpretable for subsequent policy learning in visualization analysis.

2 Preliminaries

2.1 Multiagent Reinforcement Learning

Multi-agent reinforcement learning is a sub-field of reinforcement learning that focuses on learning how multiple agents can interact with each other in a dynamic environment to achieve common goals. This process can be described by a Markov game (N, S, A, T, r, γ), where N represents the number of agents, S is the joint state space of agents $S = S_1 \times \cdots \times S_i \times \cdots \times S_N$, where S_i is the state space of agent i, A is the joint action space of agents $A = A_1 \times \cdots \times A_i \cdots \times A_N$, where A_i is the action space of agent i, T is the transition function, $T : S \times a_1 \times \cdots \times a_i \times \cdots \times a_N \times S \to [0, 1]$, where a_i is the action of agent i. $r_i(s_t, a_1, \ldots a_i, \ldots, a_N, s'_{t+1})$ is the reward function for agent i, where s_t is the state of agent i at time step t and s'_{t+1} is the state of agent i at time step $t + 1$. In MARL, each agent perceives an observation of their surrounding environment o_t at time step t. Based on this information, the agents select joint actions according to policy π. After carrying out the actions, states of the agents and the environment change based on transition function T. Meanwhile, each agent receives a reward signal from the environment, which is used to update the policy.

2.2 Domain Randomization and LMDPs

In this section, we introduce domain randomization (DR) under the framework of latent Markov decision process (LMDP). An LMDP is composed of a set of MDPs $\mathcal{M}$ and a distribution μ over $\mathcal{M}$. An MDP M_l in $\mathcal{M}$ can be sampled from the distribution μ, and can be represented by (S, A, T_l, R_l, ρ_l), where S and A are shared state space and action space in $\mathcal{M}$, T_l is the sampled transition function, R_l is the sampled reward function, ρ_l is the sampled initial states in each episode. The optimal policy in LMDPs is defined as the policy π with the best expectation performance over μ:

$$V^*_{\mathcal{M}} := \max_{\pi \in \Pi} \sum_{l=1}^{|\mathcal{M}|} w_l \mathbb{E}\pi \left[\sum_{t=1}^{H} r_t \right]$$

where $V^*_{\mathcal{M}}$ is the optimal total value function in $\mathcal{M}$ of the optimal policy π^* in a policy set Π, ω_l is the weight coefficient for M_l, H is the episode horizon for each MDP.

In DR, similar MDPs of the same task form the MDP set $\mathcal{M}$ in LMDP. Considering two MDPs M_i and M_j in $\mathcal{M}$, M_i and M_j are represented by $(S_i, A_i, T_i, R_i, \gamma)$ and $(S_j, A_j, T_j, R_j, \gamma)$, respectively. To guarantee the performance of the optimal policy, it is essential that the optimal policy in the LMDP and the optimal policies in individual MDPs exhibit similarities. Therefore, M_i and M_j must have similar state space, action space, transition functions and reward functions, i.e. $S_i \approx S_j, A_i \approx A_j, T_i \approx T_j, R_i \approx R_j$. The values of S_i, A_i, T_i, R_i for MDP M_i are indirectly determined by the parameters of environment in M_i. Therefore, the distribution μ of MDPs in $\mathcal{M}$ can be transformed to a distribution over tunable environmental parameters in $\mathcal{M}$. During training, an MDP M_l in $\mathcal{M}$ is sampled at the start of each episode and remains fixed throughout the whole episode. Agents interact with M_l without knowing the identity.

2.3 Domain Adaptation

In this subsection, we introduce domain adaptation (DA) in the framework of transfer learning. In transfer learning, data is categorized into source domains and target domains. In reinforcement transfer learning, a domain is viewed as an MDP. A source domain is defined as M_s and a target domain is defined as M_t. Transfer learning methods utilize knowledge learned in M_s to facilitate learning in M_t.

For policy optimization in a single static MDP, the underlying trajectory distributions of training and testing are the same, i.e. $M_s = M_t$. However, in MDPs with dynamic environments, traditional RL algorithms fail due to domain shift, i.e. $M_s \neq M_t$. DA addresses this issue by extracting shared latent state representations among the source and target domains, which can be utilized in multi-domain policy optimization. However, DA does not explicitly consider action space, dynamics transitions and reward functions. Therefore, to apply

DA, the targeted MDPs should have the same action space, similar transitions, similar reward functions and distinct state spaces, i.e. $A_s = A_t, T_s \approx T_t, R_s \approx R_t, S_s \neq S_t$. In our method, we use DA to optimize policies in DR.

3 Method

In this section, we adopt the formulations and expressions presented in Sect. 2. To improve the generalization ability of MARL algorithms, we first employ domain randomization to generate an LMDP with various environmental parameter configurations in our method. To optimize policies in this LMDP, we leverage domain adaptation to discover shared feature space among MDPs and design curricula for policy training. Finally, our policy is evaluated on a set of MDPs sampled from distributions in domain randomization. Our method comprises three stages: 1) Environment sampling. 2) Pretraining. 3) Policy learning. Figure 1 shows the framework of our method.

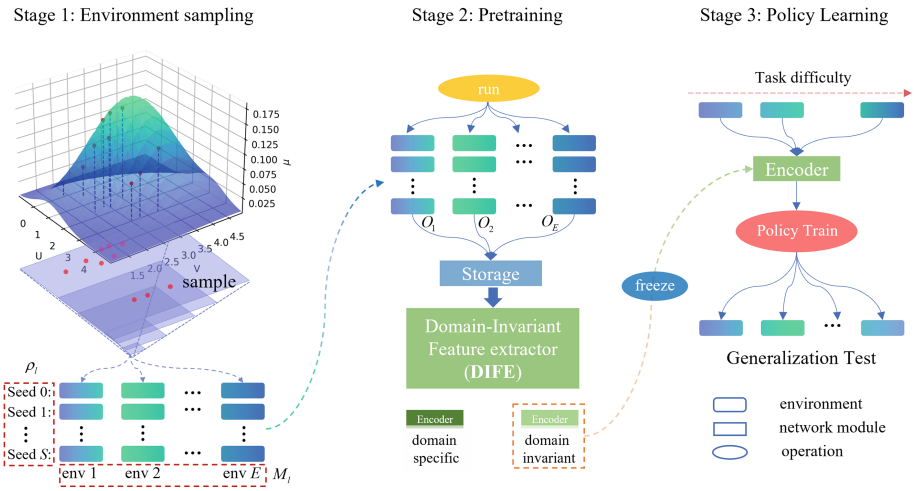


Fig. 1. The framework of our method

3.1 Environment Sampling

For a given task, the complexity of the task is primarily determined by some decisive parameters. To create a smooth task space for generalized policy training, we first reveal the relations of these decisive parameters and build reasonable distributions for them. Then, we sample environments based on the distributions to form an environment set.

Consider an LMDP $\mathcal{M}$ with K environmental variable parameters $v_i, i = 1, 2, \ldots, K$, each parameter has a predefined value range. Firstly, a base policy

π_b is trained in a base environment E_b and evaluated in environment i with variations in v_i compared to E_b. To assess the importance of each parameter, a performance decrease threshold P_{th} is defined. After K groups of evaluation, parameters with performance decrease above P_{th} are identified as decisive parameters. To build a sampling distribution μ over (u, v), regression analysis is performed based on the performance of π_b in different environments with varying values of (u, v). Once we obtain distribution μ through experiments, we use it to randomly generate E environments and sample S seeds for the initial states ρ_l of agents in each environment.

3.2 Pretrain

After collecting $E \times S$ environments, we can exploit the mutual information present within these environments through pretraining, as depicted in Stage 2 of Fig. 1. Specifically, we execute a scripted searching policy in these E environments (with each environment running S seeds), collect and store E batches of observations denoted as $O_1, O_2, \cdots, O_E$. Next, we train our domain-invariant feature extractor (DIFE) module by sampling observations from the stored batches.

The DIFE module of our method is based on cycle-consistent VAE, a generative model designed to learn disentangled latent representations of data. The cycle-consistent VAE is grounded on the concept of cycle consistency, which asserts that the composition of well-trained forward and reverse transformations, in any order, should closely approximate an identity function. Cycle consistency comprises of forward consistency and reverse consistency. In the VAE model, the encoder is a forward transformation that converts an input image into a latent feature vector, while the decoder is the reverse transformation that converts the latent vector back to a reconstructed image. Forward consistency implies that the newly reconstructed observation should be similar to the original observation after encoding and decoding the original observation. Reverse consistency implies that the newly obtained features should be similar to the original features after decoding and encoding them. Therefore, to obtain cycle consistency, we use two reconstruction losses for the forward and backward transformations. To provide a comprehensive overview of the training process of the DIFE module, we discuss a simple scenario involving two domains, where in our work each domain represents an MDP with a distinct set of environmental parameters.

As depicted in Fig. 2, assuming that two domains of data are provided, domain i and domain j. O_i and O_j represent the observation sets in domain i and domain j. o_1^i and o_2^i are two different observations in O_i. o_3^i and o_4^j are random observations in O_i and O_j, respectively. The encoder function $\mathrm{Enc}_\theta(\cdot)$ is parameterized by θ and the decoder function $\mathrm{Dec}_\phi(\cdot)$ is parameterized by ϕ . Processed by the encoder, the observations are mapped into latent feature embeddings $o_1^i \to z_1^i = \left\langle \bar{z}_1^i, \hat{z}_1^i \right\rangle, o_2^i \to z_2^i = \left\langle \bar{z}_2^i, \hat{z}_2^i \right\rangle, o_3^i \to z_3^i = \left\langle \bar{z}_3^i, \hat{z}_3^i \right\rangle, o_4^j \to z_4^j = \left\langle \bar{z}_4^j, \hat{z}_4^j \right\rangle$, where $z_1^i, z_2^i, z_3^i, z_4^j$ are latent features of $o_1^i, o_2^i, o_3^i, o_4^j$, $\bar{z}_1^i, \bar{z}_2^i, \bar{z}_3^i, \bar{z}_4^j$ are domain-invariant features in $z_1^i, z_2^i, z_3^i, z_4^j$ and $\hat{z}_1^i, \hat{z}_2^i, \hat{z}_3^i, \hat{z}_4^j$ are domain-specific features in $z_1^i, z_2^i, z_3^i, z_4^j$. The loss function of DIFE contains two parts.

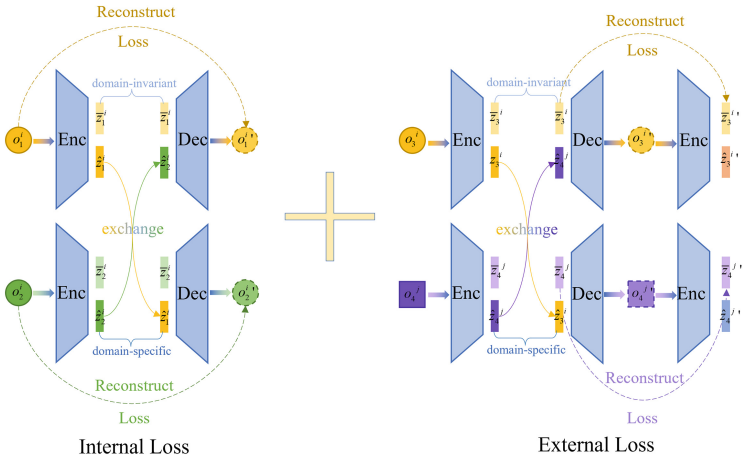


Fig. 2. Cycle-consistent VAE

Internal Loss. This loss corresponds to forward transformation in cycle consistency and contains both the self-reconstruction loss for VAE and the reconstruction loss for domain-specific feature extraction. Domain-specific features exhibit variation across domains but remain constant within a given domain. Therefore, if two features within the same domain have their domain-specific components swapped, the reconstructed observations should restore the original observations, which is $o_1^i \neq o_2^i, \bar{z}_1^i \neq \bar{z}_2^i, \hat{z}_1^i = \hat{z}_2^i$. The internal loss contains both VAE reconstruction loss and domain-specific feature extraction loss.

$$\mathcal{L}_{\text{internal}} = \mathcal{L}_{\text{VAE}} + k_1 \mathcal{L}_{\text{specific}}$$

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{\text{Enc}_\theta\left(\bar{z}_1^i, \hat{z}_1^i | o_1^i\right)} \left[\log \text{Dec}_\phi \left(o_1^i \mid \bar{z}_1^i, \hat{z}_1^i\right)\right]$$

$$\mathcal{L}_{\text{specific}} = \mathbb{E}_{\text{Enc}_\theta\left(\bar{z}_1^i, \hat{z}_1^i | o_1^i\right) \cdot \text{Enc}_\theta\left(\bar{z}_2^i, \hat{z}_2^i | o_2^i\right)} \left[\log \text{Dec}_\phi \left(o_1^i \mid \bar{z}_1^i, \hat{z}_2^i\right)\right]$$

where $\mathcal{L}_{\text{internal}}$ represents the internal loss of DIFE, $\mathcal{L}_{\text{VAE}}$ represents the VAE loss, $\mathcal{L}_{\text{specific}}$ represents the loss for domain-specific feature extraction, and k_1 is the weight coefficient for $\mathcal{L}_{\text{VAE}}$ and $\mathcal{L}_{\text{specific}}$.

External Loss. This loss corresponds to reverse transformation in cycle consistency. Domain-invariant features contain little domain-specific information but can be reconstructed across all domains. Therefore, if the domain-specific feature $\hat{z}_3^i$ of the observation o_3^i is replaced by $\hat{z}_4^j$, the embedding of the reconstructed observation $o_3^{i'} = \text{Dec}_\phi\left(\bar{z}_3^i, \hat{z}_4^j\right)$ can still restore the domain invariant feature $\bar{z}_3^i$.

$$\mathcal{L}_{\text{external}} = \mathbb{E}_{\bar{z}^i} \left[\left\| \text{Enc}_\theta \left(\text{Dec}_\phi \left(\bar{z}_3^i, \hat{z}_4^j\right)\right) - \bar{z}_3^i \right\|_1\right]$$

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{internal}} + k_2 \mathcal{L}_{\text{external}}$$

where $\mathcal{L}_{\text{external}}$ represents the external loss of DIFE, $\mathcal{L}_{\text{total}}$ represents the overall loss of DIFE, and k_2 is the weight coefficient for $\mathcal{L}_{\text{internal}}$ and $\mathcal{L}_{\text{external}}$.

3.3 Policy Learning

Upon completing the pretraining stage, the parameters in the DIFE module are frozen to aid policy learning, as depicted in Stage 3 of Fig. 1. During this stage, a batch of environments is randomly sampled from the distribution μ. Initially, the policy is optimized in a random order of these environments to evaluate its performance and efficiency. Subsequently, the policy is optimized across these environments, arranged in ascending order of task difficulty, to enhance its performance in more challenging tasks. Finally, a new batch of environments is sampled from μ to assess the policy's generalization ability. Any MARL algorithm can be used as the policy training algorithm.

Our experiments demonstrate that the DIFE module improves policy performance in various scenarios and enhances generalization ability. Furthermore, this module can be used as a plugin in all MARL algorithms to aid policy learning.

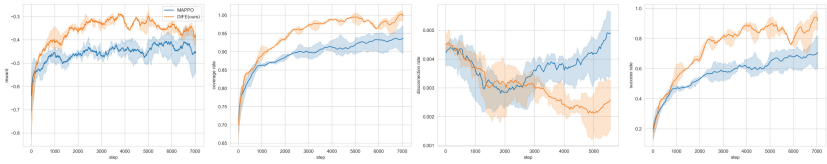
4 Experiment

We selected the multi-agent particle environment (MPE) as our test platform due to its diverse configurations of multi-agent environments. In our previous work [27], we addressed the challenging problem of cooperative coverage and connectivity maintenance in MPE by utilizing graph attention networks. This task encompasses a broader range of environmental configurations compared to other MPE tasks. The primary experiments are conducted in this task, and more details can be found in [27]. Agents in this task need to cooperate in a 2D space to cover targets within their coverage range while maintaining communication links with other agents within their communication range. The goal of this task is to cover as many targets as possible while ensuring a connected communication topology among the agents. The evaluation metrics for this task include training reward, coverage rate, disconnection rate, and success rate. The coverage rate is the percentage of covered targets, the disconnection rate is the percentage of stpng where agents are disconnected, and the success rate is the percentage of episodes where agents cover 90% of the targets while maintaining connectivity.

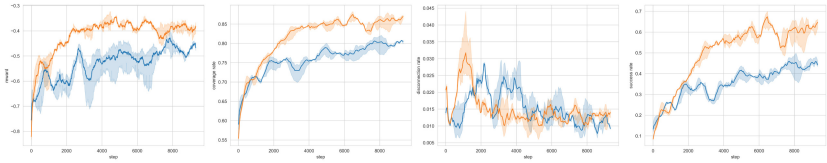
To identify the most influential parameters of this task, we first conduct experiments on a range of parameters. We use MAPPO as our base algorithm. Our testing results indicate that the size of the arena h and the number of agents N are the primary factors. Intuitively, if we represent the joint state space as $S = S_1 \times \cdots \times S_i \times \cdots \times S_N$, where S_i is the state space of an individual agent i and N is the number of agents, the arena size h determines the range of individual state spaces, while the range of joint state space increases exponentially with N.

During the pretraining process, we employ a scripted searching policy to gather observations from sampled environments. By analyzing the testing performance of our base policy, we establish correlations between h and N, and determine a sampling distribution of the two parameters. We then randomly sample 100 environments in this distribution to gather observations.

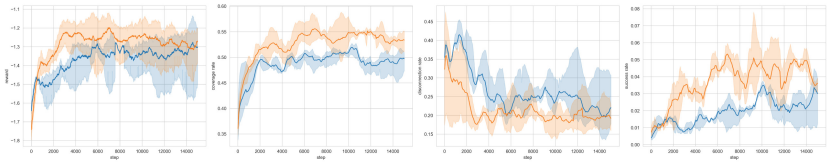
After pretraining, we freeze the encoder layers to generate domain-invariant features in downstream policy training. We select three representative configurations with different N and h to illustrate the performance of our method in Fig 3.



(a) $N = 3, h = 1.0$



(b) $N = 5, h = 1.2$



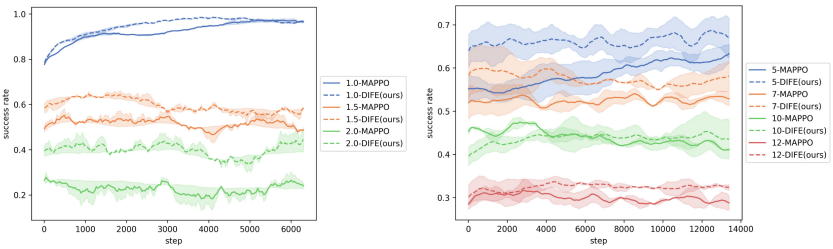
(c) $N = 10, h = 2.0$

Fig. 3. Training curves of MAPPO and DIFE (ours) in 3 representative scenarios. (Three rows represent three scenarios, respectively. Four columns, from left to right, represent training reward, coverage rate, disconnection rate, and success rate)

The results of our experiments illustrate that, with similar disconnection rates, our method consistently outperforms MAPPO in terms of success and coverage rates, as well as reward. This indicates that our DIFE module is a highly effective means of improving policy performance.

To improve the policy performance in more complex scenarios, we arrange the sampled environments in the sequence of increasing values for h and N. The results indicate that our feature extractor can effectively enhance learning performance as h increases while showing less improvement as N increases. This is due to the fact that in MARL, the optimization complexity increases exponentially with the number of agents. Learning unified latent feature embeddings have limited impacts on optimizing policies in such complex scenarios.

We also performed generalization tests by randomly sampling environments using domain randomization. The policies trained in the base environments were then evaluated in these sampled environments. We selected 12 representative environments to demonstrate the generalization ability of our method (Fig. 4).



(a) arena size (b) number of agents

Fig. 4. Curriculum training. (In fig (a), $N = 3$ remains constant, and the arena size h is 1.0, 1.5, 2.0. In fig (b), the number of agents N is 5, 7, 10, 12)

Table 1. Coverage rates in generalization tests

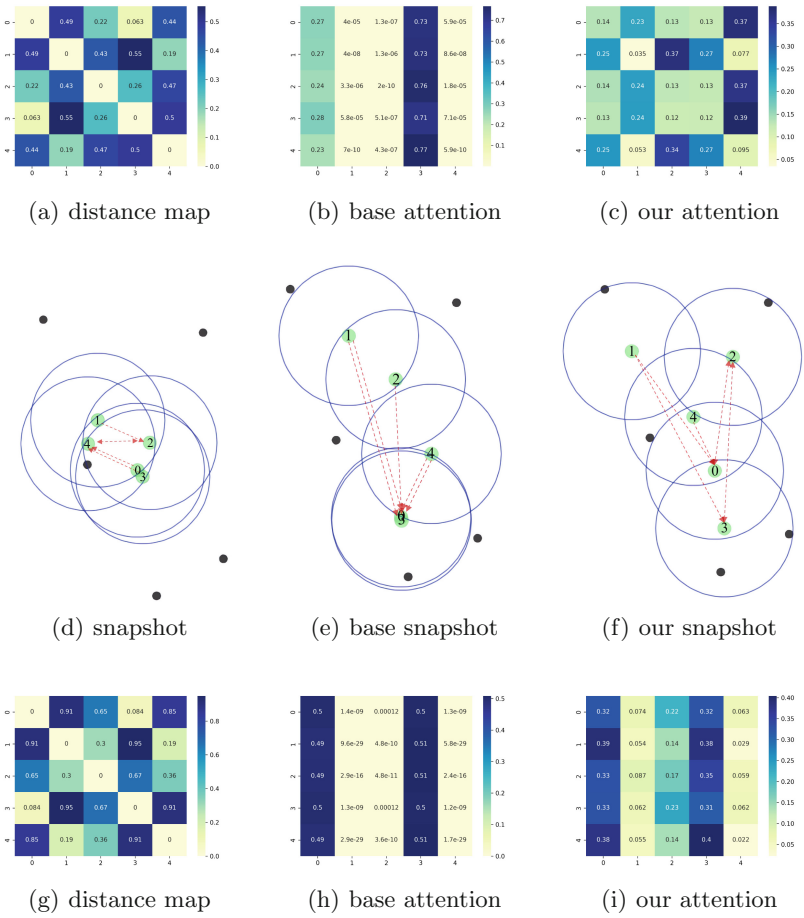
3			5		
size	base	**ours**	size	base	**ours**
1.0	0.87±0.01	**0.92±0.00**	1.2	0.68±0.05	**0.76±0.03**
1.2	0.67±0.02	**0.73±0.02**	1.5	0.44±0.05	**0.46±00.06**
1.5	0.40±0.02	**0.47±0.01**	1.7	**0.30±0.02**	0.29±0.02
7			10		
size	base	**ours**	size	base	**ours**
1.5	0.60±0.05	**0.67±0.03**	1.5	0.62±0.03	**0.67±0.02**
1.7	0.43±0.02	**0.49±0.02**	1.7	0.41±0.04	**0.45±0.03**
1.9	0.24±0.04	**0.29±0.01**	1.9	0.24±0.03	**0.31±0.04**

Table 2. Disconnection rates in generalization tests

3			5		
size	base	**ours**	size	base	**ours**
1.0	0.67±0.05	**0.33±0.05**	1.2	2.33±0.12	**0.00±0.00**
1.2	4.00±0.08	**0.33±0.05**	1.5	4.00±0.08	**0.00±0.00**
1.5	5.00±0.08	**0.33±0.05**	1.7	3.33±0.09	**0.00±0.00**
7			10		
size	base	**ours**	size	base	**ours**
1.5	1.67±0.12	**1.00±0.00**	1.5	**0.67±0.05**	1.0±0.08
1.7	0.67±0.05	**0.67±0.05**	1.7	0.67±0.09	**0.33±0.05**
1.9	0.00±0.00	**0.00±0.00**	1.9	0.33±0.05	**0.33±0.05**

As illustrated in Tables 1 and Table 2, policy performance tends to deteriorate as the arena size or the number of agents increases. Nonetheless, our method exhibits better adaptability with higher coverage rates and lower disconnection rates than the base policy.

To assess the effectiveness of our approach, we visualize the output of both the attention module in our method and the base policy in Fig. 5. In the first row, it is evident that the attention map computed by the base policy focuses heavily on agents 0 and 3, likely due to a preference for maintaining connectivity, resulting in a conservative and overfitting policy. In contrast, the attention computed by our method corresponds more closely with the distance map, offering a more flexible and interpretable approach to deploying agents.



(a) distance map (b) base attention (c) our attention

(d) snapshot (e) base snapshot (f) our snapshot

(g) distance map (h) base attention (i) our attention

Fig. 5. Attention Visualization for $N = 5, h = 1.2$ in the same episode. (The first row shows visualization maps at the beginning of the episode. The second row shows the rendering snapshots. In the snapshots, the red dotted lines with arrows are the highest attention for each agent. fig (d) shows the snapshot at the beginning of the episode, fig (e) shows the snapshot of MAPPO at the end of the episode, fig (f) shows the snapshot of our method at the end of the episode. The third row shows visualization maps at the end of the episode) (Color figure online)

In the second and third rows, the base policy concentrates its attention almost exclusively on agents 0 and 3, resulting in overlap between the two agents. Our method, on the other hand, attends not only to these two agents but also to agent 2, which is responsible for covering the upper right target and is at risk of disconnecting from the system. Moreover, it is observed that our method distributes attention more uniformly across all agents at the end of the episode, in contrast to the base policy. These results suggest that the observations collected across domains assist in learning a more effective and explainable feature extractor that captures latent domain-invariant features.

5 Conclusion

In this work, we propose a method that leverages domain randomization and domain adaptation to improve the training efficiency and generalization performance of MARL algorithms. Our approach disentangles the domain-invariant and domain-specific features across domains by employing cycle consistency and utilizes the domain-invariant features to facilitate policy learning. Our experiments demonstrate the effectiveness of this approach, and our analysis shows that DIFE captures more interpretable latent features than the compared method.

Acknowledgments. This work was supported by the Beijing Nova Program under Grant 20220484077, the National Natural Science Foundation of China under Grant 62073323, the External cooperation key project of Chinese Academy Sciences No. 173211KYSB20200002.

References

1. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Dumitru Erhan, D.: Domain Separation Networks. In: Proceedings of NeurIPS, pp. 343–351. Curran Associates Inc (2016). ISBN 978-1-5108-3881-9
2. Chen, X., Hu, J., Jin, C., Li, L., Wang, L.: Understanding domain randomization for sim-to-real transfer. In: ICLR (2022)
3. Hassabis, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419), 1140–1144 (2018). ISSN 0036–8075. https://doi.org/10.1126/science.aar6404. Publisher: American Association for the Advancement of Science
4. Ganin, Y., et al.: Domain-Adversarial Training of Neural Networks. In: Csurka, G. (ed.) Domain Adaptation in Computer Vision Applications. ACVPR, pp. 189–209. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58347-1_10
5. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 597–613. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_36
6. Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R.P., Levine, S.: Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. In: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, Advances in NeurIPS, vol. 34, pp. 25502–25515. Curran Associates Inc (2021)

7. Hoffman, J., Tzeng, E., Darrell, T., Saenko, K.: Simultaneous Deep Transfer Across Domains and Tasks. In: Csurka, G. (ed.) Domain Adaptation in Computer Vision Applications. ACVPR, Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58347-1_9
8. Jha, A.H., Anand, S., Singh, M., Veeravasarapu, V.S.R.: Disentangling Factors of Variation with Cycle-Consistent Variational Auto-encoders. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 829–845. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_49
9. Kwon, J., Efroni, Y., Caramanis, C., Mannor, S.: RL for Latent MDPs: Regret Guarantees and a Lower Bound. In: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, eds, Advances in NeurIPS, vol. 34, pp. 24523–24534. Curran Associates Inc (2021)
10. Liu, A.H., et al.: A unified feature disentangler for multi-domain image translation and manipulation. In: Proceedings of NeurIPS, pp. 2595–2604 (2018). https://papers.nips.cc/paper/7525-a-unified-feature-disentangler-for-multi-domain-image-translation-and-manipulation
11. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Proceedings of ICML, ICML'15, pp. 97–105. JMLR.org (2015)
12. Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., Savarese, S.: Adversarially robust policy learning: active construction of physically-plausible perturbations. In: Proceedings of IROS, pp. 3932–3939 (2017). https://doi.org/10.1109/IROS.2017.8206245. ISSN: 2153-0866
13. OpenAI, Christopher Berner, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680 (2019)
14. OpenAI, Akkaya, I., et al.: Solving Rubik's Cube with a Robot Hand. arXiv:1910.07113 (2019)
15. Peng, X.B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Sim-to-real transfer of robotic control with dynamics randomization. In: Proceedings of ICRA, pp. 3803–3810 (2018). https://doi.org/10.1109/ICRA.2018.8460528. arXiv:1710.06537
16. Reed, S., et al.: A Generalist Agent. arXiv:2205.06175 (2022)
17. Sadeghi F., Levine, S.: CAD2RL: real single-image flight without a single real image. arxiv.org/abs/1611.04201arXiv:1611.04201 (2017)
18. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature 529(7587), 484–489 (2016). ISSN 1476–4687. https://doi.org/10.1038/nature16961. Number: 7587 Publisher: Nature Publishing Group
19. Sun, B., Saenko, K.: Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 443–450. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_35
20. Adaptive Agent Team, et al.: Human-timescale adaptation in an open-ended task space. arXiv:2301.07608 (2023)
21. Tobin, J., et al.: Domain randomization for transferring deep neural networks from simulation to the real world, arXiv:1703.06907 (2017)
22. Tzeng, E., J., H., Zhang, N., Saenko, K., Trevor Darrell, T.: Deep domain confusion: maximizing for domain invariance. arXiv:1412.3474 (2014)
23. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of CVPR, pp. 2962–2971 (2017). https://doi.org/10.1109/CVPR.2017.316. ISSN: 1063–6919

24. Vinyals, O, et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), 350–354 (2019). ISSN 1476–4687. https://doi.org/10.1038/s41586-019-1724-z. Number: 7782 Publisher: Nature Publishing Group
25. Xing, J., Nagata, T., Chen, K., Zou, X., Neftci, E., Krichmar, J.L.: Domain adaptation in reinforcement learning via latent unified state representation. In: Proceedings of AAAI, vol. 35, pp. 10452–10459 (2021). https://doi.org/10.1609/aaai.v35i12.17251
26. Xing, Y., Song, O., Cheng, G.: On the algorithmic stability of adversarial training. In: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan, editors, Advances in NeurIPS, vol. 34, pp. 26523–26535. Curran Associates Inc (2021)
27. Yifan Xu, Y., et al.: A double-observation policy learning framework for multi-target coverage with connectivity maintenance. In: Ren, Z., Wang, M., Hua, Y., eds, Proceedings of CCSICC, pp. 1279–1290. Springer Nature Singapore (2023). https://doi.org/10.1007/978-981-19-3998-3_120