PAnDR: Fast Adaptation to New Environments from Offline Experiences via Decoupling Policy and Environment Representations

Anonymous authors
Paper under double-blind review

Abstract

Deep Reinforcement Learning (DRL) has been a promising solution to many complex decision-making problems. Nevertheless, the notorious weakness in generalization among environments prevent widespread application of DRL agents in real-world scenarios. Although advances have been made recently, most prior works assume sufficient online interaction on training environments, which can be costly in practical cases. To this end, we focus on an *offline-training-online-adaptation* setting, in which the agent first learns from offline experiences collected in environments with different dynamics and then performs online policy adaptation in environments with new dynamics. In this paper, we propose Policy Adaptation with Decoupled Representations (PAnDR) for fast policy adaptation. In offline training phase, the environment representation and policy representation are learned through contrastive learning and policy recovery, respectively. The representations are further refined by mutual information optimization to make them more decoupled and complete. With learned representations, a *Policy-Dynamics Value Function* (PDVF) (Raileanu et al., 2020) network is trained to approximate the values for different combinations of policies and environments. In online adaptation phase, with the environment context inferred from few experiences collected in new environments, the policy is optimized by gradient ascent with respect to the PDVF. Our experiments show that PAnDR outperforms existing algorithms in several representative policy adaptation problems.

1 Introduction

Reinforcement learning (RL) has achieved great successes in many fields, e.g., Game Playing (Silver et al., 2016), Robotics (Andrychowicz et al., 2020), Supply Chain (Ni et al., 2021), Medicine Discovery (Schreck et al., 2019) and so on. Despite of the successes, conventional RL is well known to be lack of generalization ability (Packer et al., 2018). To be specific, a policy well optimized in a specific environment can be catastrophically poor even in similar environments. The poor generalization among environments and policys severely prevents more practical applications of RL.

In the literature on RL generalization (Kirk et al., 2021), many approaches like data augmentation (Yarats et al., 2021) and domain randomization (Peng et al., 2018) are proposed to increase the similarity between training environments and the testing environments (i.e., the ones to generalize or adapt to). These methods often require knowing the variations of environment and having the access to environment generation. In contrast, some other works aim to achieve fast adaptation in testing environments without the need for such access and knowledge. Gradient-based Meta RL, e.g., MAML (Finn et al., 2017), aims to learn a meta policy which can adapt to a testing environment within few policy gradient steps. Also in this branch, context-based Meta RL, e.g., PEARL (Rakelly et al., 2019), leverages a context-conditioned policy which allows to achieve adaptation by the generalization among contexts. Useful context representation is learned in training environments to capture the variations; then the context of a testing environment can be inferred from few probed interaction experiences (Fu et al., 2021).

Most prior works consider an *online* training setting where the agent is allowed to interact with training environments arbitrarily within the interaction budget. However, in real-world problems, online interaction is usually expensive while offline experiences are often available and relatively sufficient. For example, in E-commerce, adapting the advertising strategies obtained from massive

offline ads-users interaction data to a new user can effectively save the budgets of advertisers. On this point, a recent work (Raileanu et al., 2020) studies fast adaptation under an *offline-training-online-adaptation* setting, where only offline experiences collected on training environments are available before online policy adaptation in new environments. This is a more practical and general setting which is meanwhile more challenging, since the policy learning and variation capture need to be conducted in an offline manner. To solve this problem, Raileanu et al. (Raileanu et al., 2020) propose an extended value function, called Policy-Dynamic Value Network (PDVF). By definition, a PDVF additionally takes an environment representation and an policy representation as inputs, and evaluates the corresponding values. In principle, the explicit representation allows the generalization of value estimation among environments and policies, which can be utilized to realize policy adaptation. To be specific, given the offline experiences generated from different combinations of training policies and training environments, the representations of environment and policy are learned from transitions and state-action sequences by dynamics prediction and policy recovery respectively. Thereafter, a quadratic form of PDVF is approximated, according to which the policy is optimized during online adaptation with the environment representation inferred from few interactions in the testing environment. The first drawback is that the representations of environment and policy can have redundant and entangled information since they are from the same interaction experiences of the corresponding combination. It negatively affects the generalization conveyed by the representations which is vital to fast adaptation. Another limitation is that the quadratic approximation of PDVF inevitably makes a sacrifice in expressivity for convenient closed-form optimization, in turn this cripples the optimality of policy obtained during adaptation.

In this paper, we focus on the offline-training-online-adaptation setting and follow the paradigm of (Raileanu et al., 2020). We propose **P**olicy **A**daptation with **D**ecoupled **R**epresentations (PAnDR) for more effective fast adaptation of RL. The conceptual illustration of PAnDR is shown in Fig. 1. Compared to transition-level dynamics prediction adopted by (Raileanu et al., 2020), we adopt trajectory-level context contrast for environment representation learning. By this means, the variation is captured in an integral manner and the inter-environment difference is emphasized, thus offering more useful information and better robustness. One can link our comparison here to that between reconstruction and contrastive learning, where the superiority of the latter has been widely demonstrated (He et al., 2020; Laskin et al., 2020; Fu et al., 2021). Moreover, we propose a information-theoretic representation learning method to refine environment and policy representations for information decoupleness and completeness. This is achieved by mutual information minimization and maximization among environment and policy representations and their joint. In addition, distinct to the quadratic form in (Raileanu et al., 2020), PAnDR leverages a typical multi-layer nonlinear neural network for the approximation of the PDVF. Then during online adaptation, the policy is optimized by gradient ascent. In this sense, PAnDR gets rid of such constraints to improve value approximation and policy adaptation. In experiments, we evaluate PAnDR in a series of fast adaptation tasks and analyze the contribution of each part of proposed methods.

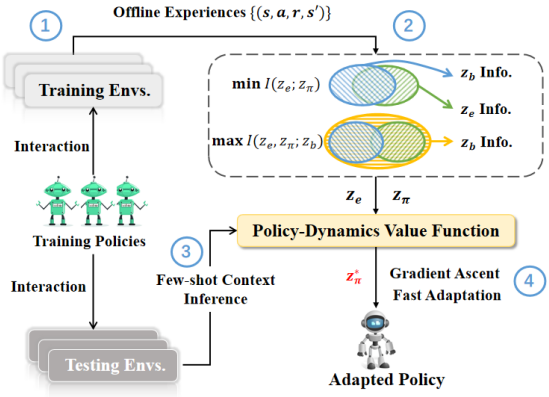


Figure 1: A conceptual illustration of PAnDR for fast adaptation with offline experiences (best view in numbers). With offline experiences collected by training policies and environments (①), the representations for environment (z_e) and policy (z_π) are learned, which are further improved by mutual information optimization with extra joint representation z_b (②). Conditioning on learned representations, a PDVF is trained. For adaptation, the representation of testing environment is inferred with few-shot online context (③), based on which the policy (representation) is adapted according to PDVF (④).

We summarize our main contributions as follows: 1) We leverage trajectory-level contrastive learning and propose a novel mutual information based approach for better environment and policy representations. 2) We get rid of the constraint of prior quadratic approximation of PDVF and demonstrate the effectiveness of nonlinear approximation in improving online policy adaptation. 3) We show that

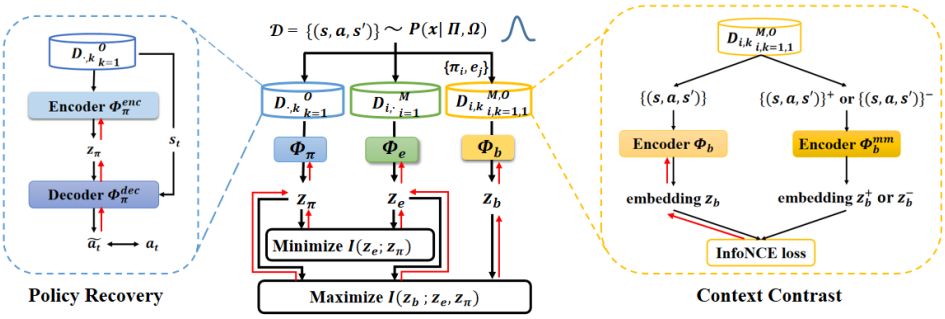


Figure 2: Illustration of learning environment representation z_e and policy representation z_π from the offline experiences $\mathcal{D}$ collected by training environments Ω and policies Π. With different views of $\mathcal{D}$, i.e., $D_{\cdot,k}$, $D_{i,\cdot}$ and $D_{i,k}$, the representations for environment, policy and their joint (z_b) are obtained by encoders ϕ_e, ϕ_π, ϕ_b respectively, which are learned by policy recovery (*left*) and context contrast (*right*). Thereafter, the representations of environment and policy are refined through mutual information based optimization, i.e., $\min I(z_e; z_\pi)$ for representation decoupleness and $\max I(z_b; z_e, z_\pi)$ for representation completeness. Gradient traces of model parameters are plotted in red.

PAnDR consistently outperforms representative existing algorithms in our experiments, sometimes with a large margin.

2 Preliminaries

2.1 Reinforcement Learning

Consider a Markov Decision Process (MDP) defined as $\mathcal{M} = \langle S, A, P, R, \gamma \rangle$, with the state space S, the action space A, the transition probability $P : S \times A \times S \to [0, 1]$, the reward function $R : S \times A \to \mathbb{R}$ and the discount factor $\gamma \in [0, 1)$. An agent interacts with the MDP $\mathcal{M}$ by performing its policy $\pi : S \times A \to [0, 1]$, generating the trajectories as $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and $r_t = R(s_t, a_t)$. The value function of policy π is defined as $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s\right]$. The objective of an RL agent is to optimize its policy to maximize the expected discounted cumulative rewards $J(\pi) = \mathbb{E}_{s_0 \sim \rho_0(\cdot)}[V^\pi(s_0)]$, where ρ_0 is the initial state distribution.

2.2 Mutual Information Optimization

Mutual Information (MI) is a widely used metric that estimates the relationship between pairs of variables in machine learning. Given two random variables X and Y, MI is a measure of dependency among them, defined as $I(X;Y) = \mathbb{E}_{p(x,y)}\left[\frac{p(x,y)}{p(x)p(y)}\right]$, with the joint probability distribution $p(x, y)$ and the marginal distributions $p(x), p(y)$. MI estimation and optimization are fundamental techniques in representation learning, e.g., in Information Bottleneck (Alemi et al., 2017).

In most problems, it is intractable to compute MI directly since the exact distributions of interested variables are not known. Thus, approximate estimation of MI is usually resorted to, which is non-trivial especially in high-dimensional space. For the general purpose of MI optimization, lower-bound and upper-bound approximations of MI are often used as surrogate objectives (Poole et al., 2019). Among the options, variational approximation is a common way to compute MI lower bound:

$$I(X;Y) \geq -\mathbb{E}_{p(x)}[\log\ p(x)] + \mathbb{E}_{p(x,y)}[\log\ q(x \mid y)].$$

The lower bound holds since the KL divergence is non-negative. For the upper bound, a recent work called Contrastive Log-ratio Upper Bound (CLUB) (Cheng et al., 2020) provides the following form:

$$I(X;Y) \leq \mathbb{E}_{p(x,y)}[\log q(x \mid y)] - \mathbb{E}_{p(x)p(y)}[\log q(x \mid y)],$$

where $q(x \mid y)$ is approximated, e.g., by maximum likelihood estimation (MLE), of actual conditional distribution $p(x \mid y)$ when p is not known.

We provide a detailed discussion on related work in Appendix E.

3 METHODOLOGY

In this section, we propose our framework for fast adaptation, named Policy Adaptation with Decoupled Representation (PAnDR). We follow the paradigm of (Raileanu et al., 2020) and consider the offline-training-online-adaptation setting with a set of training environments $\Omega_{\text{train}} = \{\mathcal{M}_i\}_{i=1}^{M}$ and a set of testing environments $\Omega_{\text{test}} = \{\mathcal{M}_j\}_{j=1}^{N}$. Given a set of training policies $\Pi_{\text{train}} = \{\pi_k\}_{k=1}^{O}$, an offline experience buffer is established as $\mathcal{D} = \bigcup_{i=1,k=1}^{M,O} D_{i,k}$, where $D_{i,k}$ contains the s, a, r-trajectories generated by π_k in $\mathcal{M}_i$. M, N, O denote the sizes of corresponding set. In this paper, we call a set of (s, a, r) generated in an environment as the *context* of it, and a set of (s, a) collected by a policy as its *behavior*, denoted by c and b respectively.

With offline experience buffer D, PAnDR learns the representations of environment and policy through context contrast and policy recovery, respectively (Sec. 3.1). To eliminate the redundant information entangled in the two representations while retaining necessary information, we further introduce a novel approach by mutual information minimization and maximization to refine the representations (Sec. 3.2). Thereafter, we train a nonlinear neural network to approximate PDVF conditioning on both environment and policy representations and introduce the online adaptation with gradient-based policy optimization (Sec. 3.3).

3.1 SELF-SUPERVISED REPRESENTATION LEARNING

The first part of PAnDR is to learn effective representations for environment and policy from offline experiences, replying on which the environment generalization and policy adaptation are carried out. We learn these representations in a self-supervised manner via context contrast and policy recovery, to capture the pivotal information of environment variation and policy behavior.

Learning Environment Representation via Context Contrast By following the general idea of self-supervised Contrastive Learning (He et al., 2020), we make use of the contrast of contexts collected in different training environments to extract context embedding as environment representation. For a training environment $\mathcal{M}_i$, we use the contexts sampled from $D_{i,\cdot} = \bigcup_{k=1}^{O} D_{i,k}$ as anchor sample c_i and positive sample c_i^+; in turn, the context sampled from the offline experiences generated in other environments, i.e., $D_{i',\cdot}$ for any $i' \neq i$, is taken as negative sample c_i^-.

With the contrastive context samples described above, we train a context encoder ϕ_e that maps context to embedding z_e by ensuring the representation of similar sample pairs (anchors and their positives) stay close to each other while dissimilar ones (anchors and their negatives) are far apart. Formally, the context encoder is optimized with InfoNCE loss similar to (He et al., 2020; Laskin et al., 2020):

$$\mathcal{L}_{\text{CC}}(\phi_e, W) = -\mathbb{E}_{c,c^+,\{c^-\}\sim\mathcal{D}}\Big[\log \frac{\exp(z_e^\top W z_e^+)}{\exp(z_e^\top W z_e^+) + \sum_{\{c^-\}} \exp(z_e^\top W z_e^-)}\Big], \tag{1}$$

where W denotes the parameter matrix updated along with ϕ_e. Note that z_e is differentiable with respect to ϕ_e; while z_e^+, z_e^- are the embeddings for positive and negative context samples calculated by a momentum averaged version of ϕ_e.

Compared to learning environment representation by transition-level dynamics prediction as adopted in (Raileanu et al., 2020), PAnDR captures environment variations in an integral manner and emphasizes inter-environment differences by context-based contrastive learning. Thus, the environment representation can be more effective and robust, as similarly demonstrated in (Fu et al., 2021).

Learning Policy Representation via Policy Recovery For a training policy π_k, we learn the embedding of the behavior data sampled from $D_{\cdot,k} = \bigcup_{i=1}^{M} D_{i,k}$ as policy representation. As shown in the left of Fig. 2, a behavior b is fed in the behavior encoder ϕ_π^{enc}, producing the embedding z_π. With the purpose of making z_π contain the essential policy information, we utilize a z_π-conditioned policy decoder ϕ_π^{dec} which takes as input a state s and the embedding z_π, then deterministically outputs a prediction of the actual action made by π. We call this *policy recovery* (or Behavioral Cloning (Grover et al., 2018)). With the offline experiences, $\phi_\pi^{\text{enc}}, \phi_\pi^{\text{dec}}$ are updated by minimizing the l_2 prediction loss:

$$\mathcal{L}_{\text{PR}}(\phi_\pi^{\text{enc}}, \phi_\pi^{\text{dec}}) = \mathbb{E}_{s,a,b\sim\mathcal{D}}\big[\|a - \phi_\pi^{\text{dec}}(s, \phi_\pi^{\text{enc}}(b))\|_2^2\big]. \tag{2}$$

3.2 MI-BASED REPRESENTATION REFINEMENT

A consequent concern on the representations obtained above is the redundant and entangled information in them. This is because the representations for both environment and policy are learned from the offline experiences. The entangled information exists in both representations will negatively affect the generalization among environments and policies, thus hinders value approximation and policy adaptation in turn.

To address this problem, we propose a novel information-theoretic approach to refine the learned representations. The key idea is 1) to eliminate the redundant information contained in environment and policy representations for the decoupleness, while 2) to ensure the essential information in their joint representation are completely retained in the two representations. This is achieved by two ways of mutual information (MI) optimization, as illustrated in the middle part of Fig. 2.

MI Minimization for Decoupleness Firstly, for the decoupleness, we are interested in reducing the entangled information in the representations of environment and policy. To be specific, we minimize the MI between the embeddings z_e and z_π that are obtained from context and behavior data sampled from the offline experiences, i.e., $\min I(z_e; z_\pi)$. We then convert the MI minimization into minimizing the CLUB upper bound of MI, as introduced in Sec. 2.2. With the approximation conditional distribution q_{ψ_1} parameterized by ψ_1, our loss function for representation decoupleness (RD) is formulated as:

$$\mathcal{L}_{\text{RD}}(\phi_e, \phi_\pi^{\text{enc}}) = \mathbb{E}_{p(z_e, z_\pi)}\big[\log q_{\psi_1}(z_e \mid z_\pi)\big] - \mathbb{E}_{p(z_e)p(z_\pi)}\big[\log q_{\psi_1}(z_e \mid z_\pi)\big]. \tag{3}$$

Concretely, as to any $D_{i,k}$, we use the embedding pairs of $c_i, b_k \sim D_{i,k}$ for the samples from joint distribution $p(z_e, z_\pi)$; and use those of $c_i \sim D_{i,\cdot}, b_k \sim D_{\cdot,k}$ for the marginal distributions $p(z_e), p(z_\pi)$. Note that ψ_1 is not updated by $\mathcal{L}_{\text{RD}}$, while the approximate conditional distribution is trained by MLE:

$$\mathcal{L}_{\text{MLE}}(\psi_1) = -\,\mathbb{E}_{p(z_e, z_\pi)}\big[\log q_{\psi_1}(z_e \mid z_\pi)\big], \tag{4}$$

where the joint distribution is sampled in the same way as in Eq. 3. The updates defined in Eq. 3 and Eq. 4 are performed iteratively: q_{ψ_1} approximates the conditional distribution with respect to the embeddings given by $\phi_e, \phi_\pi^{\text{enc}}$; and in turn q_{ψ_1} is used in the estimation of CLUB. We refer the readers to the original paper of CLUB (Cheng et al., 2020) for better understanding.

MI Maximization for Completeness Secondly, we expect the information contained in the joint representation of environment and policy can be completely retained in both environment and policy representations. This is also necessary especially when the decoupleness described above is required, in case that the vital information of environment variation or policy behavior is overly reduced during MI minimization. This is achieved by maximizing the MI between the concatenation of the embeddings z_e, z_π and a target environment-policy joint representation z_b which contains complete information, i.e., $\max I(z_e, z_\pi; z_b)$. We resort to extracting the target representation z_b also from the offline experiences. Similar but distinct to the environment representation, z_b is learned by context-behavior joint contrast, i.e., only the context-behavior data sampled from the same $D_{i,k}$ are regarded as mutually positive samples. We omit the similar formulations here and an illustration is shown in the right part of Fig. 2. Thereafter, we maximize the variational lower bound of $I(z_e, z_\pi; z_b)$, i.e., minimize the loss function for representation completeness (RC) below:

$$\mathcal{L}_{\text{RC}}(\phi_e, \phi_\pi^{\text{enc}}, \psi_2) = -\mathbb{E}_{p(z_e, z_\pi, z_b)}\big[\log q_{\psi_2}(z_b | z_e, z_\pi)\big], \tag{5}$$

where q_{ψ_2} is the variational distribution parameterized by ψ_2, and the joint distribution are sampled by $c_i, b_k \sim D_{i,k}$. Note that the negative entropy term $-\mathcal{H}(z_b) = \mathbb{E}_{p(z_b)}[\log p(z_b)]$ irrelevant to the optimization of z_e, z_π is neglected in Eq. 5.

Therefore, combining the self-supervised learning losses and the MI-based refinement losses, the total loss function for the representations of environment and policy is:

$$\mathcal{L}_{\text{Total}}(\phi_e, \phi_\pi^{\text{enc}}) = \mathcal{L}_{\text{CC}} + \mathcal{L}_{\text{PR}} + \alpha\mathcal{L}_{\text{RD}} + \beta\mathcal{L}_{\text{RC}}, \tag{6}$$

where α, β are the balancing weights. The pseudocode of overall representation training is in Alg. 1 in Appendix.

3.3 VALUE APPROXIMATION AND POLICY ADAPTATION

So far, we have introduced the training process of environment and policy representations on the offline experiences. With the trained representations, we are ready to introduce the approximation of PDVF (Raileanu et al., 2020) and the policy optimization during online adaptation.

Value Function Network Approximation As described in Section 1, to evaluate the values of different environment-policy configurations, PDVF additionally conditions on the representations of environment and policy. Distinct from the quadratic approximation adopted in (Raileanu et al., 2020), PAnDR leverages a typical multi-layer nonlinear neural network, denoted by $\mathbb{V}_\theta(s, z_e, z_\pi)$ with parameter θ. This setting frees the function expressivity constrained by quadratic form. We suggest that this is important because the value approximation for multiple environments and policies is much more difficult than the conventional situation (e.g., V^π). Achieving better value approximation is crucial to the following policy adaptation.

For any offline experience $D_{i,k}$, we can obtain the corresponding representations z_e, z_π, conditioning on which $\mathbb{V}_\theta$ is trained by Monte Carlo method (Sutton & Barto, 1998):

$$\mathcal{L}_V(\theta) = \mathbb{E}_{s,G(s),c,b\sim\mathcal{D}}\big[\big(G(s) - \mathbb{V}_\theta(s, z_e, z_\pi)\big)^2\big], \tag{7}$$

where $G(s) = \sum_{t=0}^{T} \gamma^t r_t$ letting $s_0 = s$. The pseudocode is provided in Alg. 2 in Appendix.

Policy Adaptation via Representation Gradient Ascent After training $\mathbb{V}_\theta$ network well, in principle we can evaluate the values for any environment-policy configuration to a certain degree. For online adaptation, we first sample a training policy from Π_{train} and use it to interact with the testing environment for a few steps. Then the embedding of testing environment z_e^{on} can be inferred with the collected context. Next, starting from an arbitrary policy representation z_π^0 (randomly initialized by default), by gradient ascent (GA), we can optimize the policy representation along the direction of increasing values of $\mathbb{V}_\theta$ with learning rate η:

$$z_\pi^{t+1} = z_\pi^t + \eta \nabla_{z_\pi^t} \mathbb{V}_\theta(s, z_e^{\text{on}}, z_\pi^t). \tag{8}$$

After a certain number of GA, the obtained policy representation can be used to interact with the testing environment with the help of the z_π-conditioned policy decoder ϕ_π^{dec} (described in Sec. 3.1), which is trained offline to predict policy's action. We adopt the best-performing policy representation as z_π^* encountered during the GA process for our adapted policy in our experiments.

It is worth noting that the GA optimization itself is zero-shot, i.e., without the overhead of interacting with the testing environment. The pseudocode is in Alg. 3 in Appendix.

4 EXPERIMENT

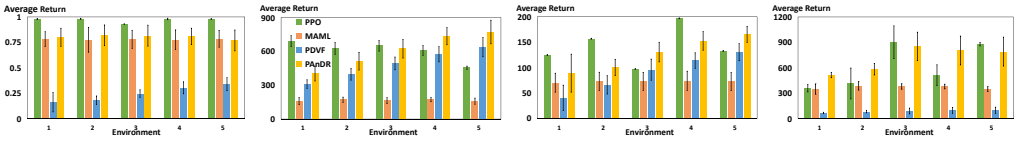
In our experiments, we aim at investigating the following questions: (**Q1**) Whether PAnDR outperforms existing algorithms for fast adaptation? (**Q2**) How do contrastive environment representation and MI-based refinement contribute to the performance of PAnDR? (**Q3**) Whether nonlinear neural approximation of PDVF and gradient-based policy adaptation improve the quadratic counterpart?

4.1 SETUP

We conduct experiments in four continuous control domains. All the experimental details can be found in Appendix C.

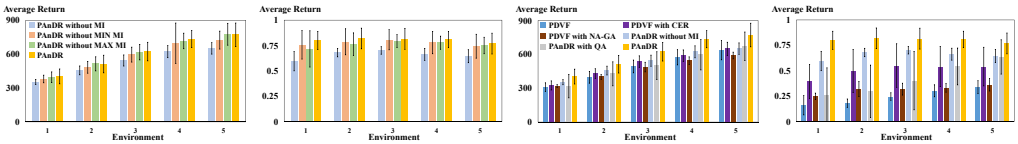
Domains and Variations. The fast adaptation domains used in our experiments is shown in Table 2 in Appendix A, where the Spaceship-Charge, Ant-Wind and Swimmer-Fluid are from (Raileanu et al., 2020) and the HalfCheetah-Mass is from (Lee et al., 2020).

Offline Experiences and Adaptation Budget. We follow the same data setting as (Raileanu et al., 2020). For offline experiences, a policy is optimized by PPO (Schulman et al., 2017) in each training environments. The obtained policies are training policies that serve as collectors. In general, such collector policies can be obtained from other ways. For each domain, 50-episode interaction trajectories are collected by each combination of training environment and policy, e.g., $15 * 15 * 50$ episodes of experience for Ant-Wind. For online adaptation, 50 steps of interaction are collected in the testing environment by one of the training policies. For PAnDR, 100 steps of GA is used by default.



(a) Spaceship-Charge (b) Ant-Wind (c) Swimmer-Fluid (d) HalfCheetah-Mass

Figure 3: Adaptation performance of different algorithms in four domains. Experimental results of algorithm generalization in new environment. Our results are even better than PPO in some environments, which proves that our method has a better generalization ability by learning in the training environment. For the PDVF algorithm, we use the source code provided by the authors.



(a) Ant-Wind (b) Spaceship-Charge (c) Ant-Wind (d) Spaceship-Charge

Figure 4: The two figures on the left are the ablations for MI-based representation refinement. The right two figures are efficacy analysis for contrast environment representation (CER) and neural approximation of PDVF with GA adaptation (NA-GA) in contrast to quadratic approximation (QA).

Evaluation Criteria. The main evaluation criterion is *adaptation performance*, calculated as the undiscounted cumulative rewards of the policy obtained after online adaptation. Moreover, for PAnDR, we analyze the evolvement of adaptation performance against the number of GA adaptation steps (in Sec. 4.3). The results reported in our experiments are means and error bars of a standard deviation calculated with 7 independent trials.

4.2 Performance Comparison

In this section, we compare the adaptation performance between PAnDR and representative existing algorithms to answer **Q1** within the domains introduced above.

Baselines. We consider the following three baselines. 1) PPO: We use the PPO policy which is trained online with $3e6$ steps in testing environments as a *powerful* baseline. 2) MAML (Finn et al., 2017): We use MAML as a representative of gradient-based Meta RL. Note MAML (as well as most Meta RL algorithms) performs online meta training. This means in principle it is *not applicable* to the offline training experiences. 3) PDVF (Raileanu et al., 2020): Since we propose PAnDR based on the same paradigm of PDVF, we use PDVF as a closely related baseline. Moreover, to our knowledge, PDVF is the state-of-the-art fast adaptation algorithm in offline-training-online-adaptation setting. Note that in this section, we use PDVF to denote the method name of (Raileanu et al., 2020) rather than the extended value function. For all above baselines, we use the official source codes provided by PDVF paper (Raileanu et al., 2020). For PAnDR, we implement on these source codes with almost the same configuration of hyperparameters and structures for a valid comparison.

(A1) Results. As shown in Fig. 3, for the simple Spaceship domain, the overall performance of PAnDR is significantly better than PDVF and slightly outperfroms MAML. Further in the three more difficult MuJoCo domains, the leading edge of PAnDR is much more evident and consistent compared with the two baseline methods, while MAML loses its competitiveness. We consider that this is because the preferable representations obtained via self-supervised learning and the MI-based refinement are more effective and robust than those of PDVF. Moreover, with less-constrained approximation, the more accurate value estimation achieved improves the optimality of policy obtained during adaptation.

It is worth noting that, perhaps somewhat surprisingly, PAnDR shows competitive and even better performance when comparing with the powerful PPO baseline trained in the testing environments. Instead of requiring millions of interactions to optimize a policy, PAnDR only needs limited interacting experiences for fast adaptation. This further reveals the superiority and potential of PAnDR.

Table 1: Comparison of the variants with respect to three algorithmic factors (Fig. 4). Abbreviations are used, i.e., Dynamics Prediction (DP), Contrastive Learning (CL).

Alg. / Variant	Env. Repre.	$\mathbb{V}_\theta$ Approx. & Policy Opt.	If Use MI
PDVF	DP	Quadratic & Close-form	✗
PDVF with CER	CL	Quadratic & Close-form	✗
PDVF with NA-GA	DP	NN Nonlinear & GA	✗
PAnDR without MI	CL	NN Nonlinear & GA	✗
PAnDR with QA	CL	Quadratic & Close-form	✓
PAnDR	CL	NN Nonlinear & GA	✓

4.3 Ablation Study and Analysis

In this section, we give the answers to **Q2** and **Q3**. We provide main results below, and the complete results along with detailed analysis are given in Appendix B.

(A2-I) MI-based Representation Refinement. To verify the contribution of the MI minimizing and maximizing in representation refinement of PAnDR, we compare PAnDR with its several variants including: 1) PAnDR without MAX MI: removes MI maximizing refinement; 2) PAnDR without MIN MI: removes MI minimizing refinement; 3) PAnDR without MI: removes both MI maximizing and minimizing refinements. As shown in Fig.4, both MI minimizing and maximizing positively contribute to the improvement of PAnDR's overall performance. It demonstrates that both MI-based refinement components are indispensable in PAnDR.

(A2-II) Context Contrast for Environment Representation. In Fig.4, we present the efficacy of context contrast in comparison with dynamics prediction upon both types of $\mathbb{V}_\theta$ in PDVF and PAnDR. The associated variants are shown in Table 1. As in Fig.4, PDVF with CER has superior performance than original PDVF while PAnDR without MI outperforms PAnDR with NA-GA. We then conclude that context contrast is more favorable in both cases of $\mathbb{V}_\theta$ compared with dynamics prediction.

(A3-I) $\mathbb{V}_\theta$ Approximation & GA Adaptation. Also with the variants in Table 1, we establish the comparsions: PAnDR v.s. PAnDR with QA, and PDVF v.s. PDVF with NA-GA. The results shown in Fig.4 verify that there is a clear advantage of using NN nonlinear approximation of $\mathbb{V}_\theta$ along with GA optimization over the quadratic counterpart.

(A3-II) Adaptation Performance against GA Steps. Since there is no longer a closed-form solution to optimization available for $\mathbb{V}_\theta$ with NN nonlinear approximation, we investigate how consecutive steps of GA affect the adpatation performance of PAnDR. The results for various choices of GA step number are provided in Table 3-8 in Appendix B.3. Moreover, we provide detailed analysis for useful insights.

5 Conclusion and Discussion

In this paper, we focus on fast adaptation problem with only offline experiences on training environments available. Following the paradigm of (Raileanu et al., 2020), we propose a new algorithm called PAnDR that consists of effective environment and policy representations and better policy optimization during online adaptation.

In essence, policy adaptation in PAnDR (and PDVF) is a process of policy representation optimization and decoding. Both the two parts are not well studied at present. Optimization in policy representation space is non-trivial since there is a lack of knowledge on the landscape of the space; while the decoding of such latent representation can be invalid without carefully handling the uncertainty (Notin et al., 2021). We consider that they are significant and there is much room for improvement. In addition, advances in environment and policy representation learning are expected to further improve the performance of PAnDR.

REFERENCES

A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy. Deep variational information bottleneck. In *ICLR*, 2017.

M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *IJRR*, 39(1):3–20, 2020.

P. Cheng, W. Hao, S. Dai, J. Liu, Z. Gan, and L. Carin. Club: A contrastive log-ratio upper bound of mutual information. In *ICML*, pp. 1779–1788, 2020.

Y. Duan, J. Schulman, X. Chen, P. Bartlett, I. Sutskever, and P. Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016.

F. Faccio, L. Kirsch, and J. Schmidhuber. Parameter-based value functions. In *ICLR*, 2021.

R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola. Meta-q-learning. In *ICLR*, 2020.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

H. Fu, H. Tang, J. Hao, C. Chen, X. Feng, D. Li, and W. Liu. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. In *AAAI*, pp. 7457–7465, 2021.

A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In *ICML*, volume 80, pp. 1797–1806, 2018.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *ICML*, pp. 3875–3886, 2020.

J. Harb, T. Schauland D. Precup, and P. Bacon. Policy evaluation networks. *CoRR*, abs/2002.11833, 2020.

K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9726–9735, 2020.

R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. A survey of generalisation in deep reinforcement learning. *CoRR*, abs/2111.09794, 2021.

M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, pp. 5639–5650, 2020.

K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *ICML*, pp. 5757–5766, 2020.

F. Ni, J. Hao, J. Lu, X. Tong, M. Yuan, J. Duan, Y. Ma, and K. He. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *KDD*, pp. 3441–3451, 2021.

P. Notin, J. M. Hernández-Lobato, and Y. Gal. Improving black-box optimization in VAE latent space using decoder uncertainty. *CoRR*, abs/2107.00096, 2021.

C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song. Assessing generalization in deep reinforcement learning. *CoRR*, abs/1810.12282, 2018. URL http://arxiv.org/abs/1810.12282.

X. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, pp. 1–8, 2018.

B. Poole, S. Ozair, A. Oord, A. Alemi, and G. Tucker. On variational bounds of mutual information. In *ICML*, volume 97, pp. 5171–5180, 2019.

R. Raileanu, M. Goldstein, A. Szlam, and R. Fergus. Fast adaptation to new environments via policy-dynamics value functions. In *ICML*, volume 119, pp. 7920–7931, 2020.

K. Rakelly, A. Zhou, C.Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, volume 97, pp. 5331–5340, 2019.

J. S Schreck, C. W Coley, and K. JM Bishop. Learning retrosynthetic planning through simulated experience. *ACS central science*, 5(6):970–981, 2019.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

H. Tang, Z. Meng, J. Hao, C. Chen, D. Graves, D. Li, C. Yu, H. Mao, W. Liu, Y. Yang, and L. Wang. What about inputting policy in value function: Policy representation and policy-extended value function approximator. *CoRR*, abs/2010.09536, 2020.

D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *ICLR*, 2021.

APPENDIX

A ENVRIONMENTS SETTING DETAILS

In each domain, we have a variety of environments controlled by corresponding variations. These variations determine the dynamics and reward functions of environment. Therefore, fast adaptation is expected to be achieved against these changes. The number of training and testing environments are provided in Table 2.

Table 2: Domains in our experiments for fast adaptation.

Domain	Train/Test Env. No.	Variation
Spaceship-Charge	15 / 5	The strength of electric field
Ant-Wind	15 / 5	Direction of wind
Swimmer-Fluid	15 / 5	Direction of the fluid
HalfCheetah-Mass	8 / 5	Joint damping coefficient

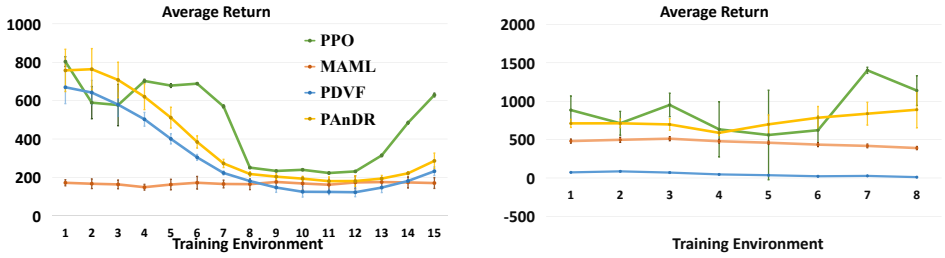
B COMPLETE EXPERIMENTAL RESULTS

B.1 PERFORMANCE ON TRAINING ENVIRONMENTS

Although the aim of meta-RL is the fast adaptation to new testing environments, we also present the results by testing the trained agents on the original training environments, as shown in Fig.5. Obviously, PPO achieves the best performance in most of the environments thanks to re-learning in each environment. Although our method is worse than PPO in most training environments, its overall performance is clearly better than MAML and PDVF. In a few environments, PAnDR even surprisingly outperforms PPO algorithm, which further reveals the superiority and potential of PAnDR.

We find that PAnDR performs poorly in environments 8, 9, 10, 11 and 12. This is because the learning of our algorithm is based on the offline dataset collected by PPO. Due to the poor performance of PPO, the collected offline experience is of low quality, and we cannot extract abundant and effective environment and policy information from it, thus limiting the performance ceiling of our algorithm

The reason for the poor performance of PDVF in the testing environment of HalfCheetah-Mass showed in Fig.3 can also be summarized from Fig.5. PDVF can't get satisfactory performances even in the training environments, so it can't be expected to generalize in the testing environments and perform well.

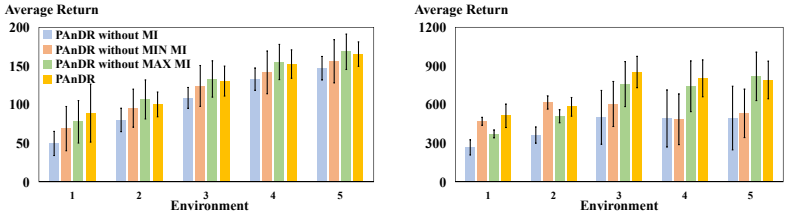


(a) Ant-Wind (b) HalfCheetah-Mass

Figure 5: Experimental results of algorithm generalization in training environment.

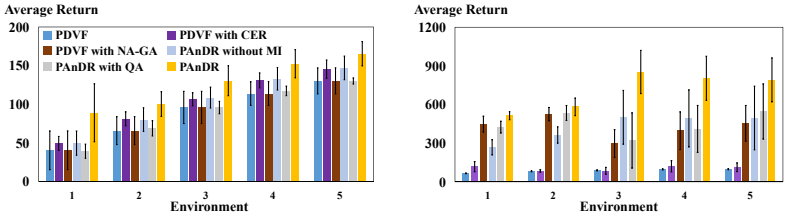
B.2 IN-DEPTH ANALYSIS OF MI-BASED REFINEMENT

Here we give the results on Swimmer and HalfCheetah in Fig.6. As mentioned in Section 4.3, both MI minimizing and maximizing positively contribute to the improvement of PAnDR's overall performance in all environment, which proves that both MI-based refinement components are indispensable in PAnDR. We also observe a phenomenon from the experiments that the results of using



(a) Swimmer-Fluid (b) HalfCheetah-Mass

Figure 6: Ablations for MI-based representation refinement in remain environments.



(a) Swimmer-Fluid (b) HalfCheetah-Mass

Figure 7: Efficacy analysis for contrast environment representation (CER) and neural approximation of PDVF with GA adaptation (NA-GA) in contrast to quadratic approximation (QA) in remain environments.

only MI-maximization refinement are slightly worse than those of using only MI-minimization refinement in most of the environments. This may mainly due to following two factors. On the one hand, additional environment-policy joint representations are introduced into the refinement process compared with that of MI-minimization. An inferior joint representation may result in limited or sometimes even negative improvements to the policy and environment representations during the refinement. On the other hand, in contrast to MI-minimization, the current policy representation space is not further compressed in MI-minimization refinement. As a result, the policy representation may still contains residual redundant information, which can negatively affect the subsequent value estimation and online adaptation.

B.3 Adaptation Performance against GA Steps.

During the online adaptation phase, we use the gradient ascent (GA) to optimize the policy representation. For this purpose, we conduct experiments of different numbers of steps of gradient ascent, the results are shown in Table 3 to Table 6. It can be seen that as the number of steps increases, the performance of the optimized policy also rises gradually.

In Fig.8, we give the distribution of policy performance at each step when the number of $steps = 200$ in the SpaceShip environment, each policy is randomly sampled as described in Alg.3. It can be concluded from the figures that different initial policies have different performance trends as the number of GA steps increases. For example, as showed in Fig.8 Policy 3 Env.1, when the selected initial policy is inferior, a better result can potentially be obtained by gradient ascent. To the opposite, when the selected initial policy is an approximated optimal one, GA with multiple steps may mislead the optimization towards the wrong direction so that the policy's performance is even worse as showed in Fig.8 Policy 3 Env.4. However, the more the number of gradients ascending, the more computation required, and Finally, considering the balance between the additional computation cost of more GA steps and the optimization of policy, we choose to stop the gradient ascend when the step reaches to 100 during the adaptation.

However, evaluating the updated policy at each step and then adopting the best-performing one will lead to heavy computational overhead. To alleviate this problem, we evaluate the updated policy every K steps. We give the results of different K as shown in Table 7 to Table 10. We can see that the results of evaluating policy at different GA frequency are relatively acceptable when $K \leq 5$, which can reduce computational cost by more than 50%.

Table 3: Policy performance against GA steps on Ant

Test Env. / Steps	10	20	50	100	200
1	371.0 ± 43	388.1 ± 45	404.9 ± 43	416.5 ± 45	428.3 ± 71
2	472.3 ± 54	494.9 ± 58	513.5 ± 58	528.2 ± 59	540.6 ± 67
3	555.5 ± 67	589.4 ± 69	612.7 ± 72	622.3 ± 76	648.0 ± 89
4	639.1 ± 70	665.3 ± 79	694.8 ± 86	715.3 ± 88	730.7 ± 94
5	668.9 ± 82	694.5 ± 92	727.2 ± 104	752.1 ± 109	797.2 ± 97

Table 4: Policy performance against GA steps on Spaceship

Test Env. / Steps	10	20	50	100	200
1	0.556 ± 0.17	0.630 ± 0.17	0.661 ± 0.15	0.698 ± 0.14	0.793 ± 0.09
2	0.718 ± 0.12	0.762 ± 0.10	0.819 ± 0.09	0.848 ± 0.08	0.858 ± 0.08
3	0.797 ± 0.08	0.829 ± 0.006	0.861 ± 0.06	0.868 ± 0.06	0.872 ± 0.06
4	0.753 ± 0.09	0.801 ± 0.06	0.837 ± 0.06	0.853 ± 0.06	0.869 ± 0.06
5	0.681 ± 0.11	0.732 ± 0.11	0.755 ± 0.10	0.794 ± 0.10	0.808 ± 0.11

B.4 Remaining Ablation Analysis

The remaining results of **Q2** and **Q3** are showed in Fig.6 and Fig.7. The conclusions are consistent with the ones in Section 4.3: (1) Context contrast is more favorable in designing $\mathbb{V}_\theta$ both in (Raileanu et al., 2020) and PAnDR compared with dynamics prediction; (2) NN nonlinear approximation of $\mathbb{V}_\theta$ along with GA optimization has clear advantage over the quadratic counterpart.

C Additional Experimental Details

C.1 Other Algorithms Experimental settings

For PDVF, we use the source codes from here[1]. In PDVF we follow the parameters settings used in the original paper. We first tried to use the source code directly for training. After running three seeds, we find that except for the Ant-Wind, PDVF cannot achieve the performance neither on Swimmer-Fluid nor Spaceship reported in the original paper. Moreover, the source codes take a lot of time collecting new data during the second stage for data augmentation. On the same device, the training time of PDVF is about three times that of PAnDR, which is obviously unacceptable. Therefore, in the subsequent experiments, we abandon the second stage of PDVF and use the results of the first stage training of 7 seeds.

For MAML, we use the codes from here[2]. In MAML, in the meta training phase, the update number is set to 500 and during each update we sample 20 trajectories from each task; in the meta adaptation phase, the update number is set to 10 and during each update we sample 20 trajectories from each task.

C.2 Equipment and Software

All experiments are performed on a Linux server with Ubuntu 18.04 containing 256G of RAM, Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz CPU processor with 56 logical cores, the server contains 4 Nvidia-2080Ti GPUs. Our codes are implemented with Python 3.7, Torch 1.7.1, gym 0.15.7, and MuJoCo 2.0.2.

C.3 Hyperparameters Setting

Before training the encoder networks, we need first to train the PPO agents on the training environments, and then use these agents to collect data. The hyperparameters setting of the PPO algorithm is shown in Table 11.

Table 12 shows the training parameters of the encoder networks. The large difference between the weights of maximizing mutual information and minimizing mutual information is due to the

[1] https://github.com/rraileanu/policy-dynamics-value-functions

[2] https://github.com/tristandeleu/pytorch-maml-rl

Table 5: Policy performance against GA steps on Swimmer

Test Env. / Steps	10	20	50	100	200
1	52.2 ± 13	54.0 ± 13	62.6 ± 14	71.7 ± 14	80.0 ± 8
2	83.1 ± 12	85.1 ± 12	93.1 ± 13	103.4 ± 14	105.7 ± 14
3	109.4 ± 12	112.6 ± 12	121.2 ± 13	127.8 ± 14	132.8 ± 15
4	132.3 ± 11	135.0 ± 11	141.9 ± 12	149.1 ± 13	154.3 ± 14
5	144.4 ± 10	146.8 ± 10	154.1 ± 11	161.5 ± 13	167.4 ± 14

Table 6: Policy performance against GA steps on HalfCheetah

Test Env. / Steps	10	20	50	100	200
1	385.3 ± 46	423.3 ± 48	478.2 ± 45	508.7 ± 46	529.4 ± 62
2	503.0 ± 58	503.8 ± 47	632.9 ± 42	638.6 ± 37	656.8 ± 60
3	404.4 ± 72	591.9 ± 55	694.5 ± 53	813.9 ± 46	816.19 ± 105
4	536.4 ± 106	631.0 ± 104	711.4 ± 109	724.5 ± 106	807.3 ± 138
5	461.3 ± 107	577.5 ± 89	657.7 ± 99	731.4 ± 108	803.9 ± 147

different ways of calculating the upper and lower bounds of MI. So we give a larger weight to the MI minimizing weight to make the smaller loss work as well.

Table 13 gives the main parameters of the value approximation and policy adaptation stage.

D ALGORITHM PSEUDOCODE

Algorithm 1 describes the training process of encoders. First, we trained the environment representation and policy representation by context contrast and policy recovery, respectively. Then the MI loss is introduced to further refine the encoder networks.

Algorithm 1: PAnDR: Representation Learning and Refinement

input : Environment encoder ϕ_e, policy encoder ϕ_π^{enc}, policy decoder ϕ_π^{dec}, policy-environment encoder ϕ_b, training set D and hyper-parameters α, β.

if *use maximum mutual information* **then**
  Divide D according to the environment and policy to get $D_{i,k}$;
  Sample anchor, positive and negative from $D_{i,k}$;
  Training ϕ_b with contrastive loss.
for *each training step* **do**
  Sample anchor and positive from $D_{i,\cdot}$, sample negative from $D_{j,\cdot}, j \neq i$;
  Calculate environment encoder contrastive loss $\mathcal{L}_{\text{CC}}$ with equation 1;
  Sample data from $D_{\cdot,k} = \bigcup_{i=1}^{M} D_{i,k}$;
  Calculate policy recovery loss $\mathcal{L}_{\text{PR}}$ with equation 2;
  if *use minimize mutual information* **then**
    Sample data from D ;
    Using encoder ϕ_e to get environment embedding z_e;
    Using encoder ϕ_π^{enc} to get policy embedding z_π;
    Calculate minimize mutual information loss $\mathcal{L}_{RD}$ using equation 3;
  if *use maximum mutual information* **then**
    Sample data from D ;
    Using encoder ϕ_e to get environment embedding z_e;
    Using encoder ϕ_π^{enc} to get policy embedding z_π;
    Using encoder ϕ_b to get policy-environment joint embedding z_b;
    Calculate maximize mutual information loss $\mathcal{L}_{RC}$ using equation 5;
Update environment encoder ϕ_e, policy encoder ϕ_π^{enc} and decoder ϕ_π^{dec} with equation 6;

Algorithm 2 and 3 correspond to the training process of the value function and the online adaptation process of the policy representation, respectively. In Algorithm 2, we formulate the training of value function as a prediction problem using Monte Carlo return as the label. In Algorithm 3, we describe

Table 7: Policy evaluation at different GA frequency on Ant

Test Env. / Steps	1	2	5	10	20	Last
1	428.3 ± 71	407.4 ± 41	386.4 ± 37	369.5 ± 34	351.4 ± 34	265.8 ± 36
2	540.6 ± 67	519.1 ± 50	485.0 ± 52	466.2 ± 54	452.7 ± 55	335.7 ± 52
3	648.0 ± 89	626.0 ± 61	604.5 ± 57	587.7 ± 62	565.5 ± 64	402.6 ± 73
4	730.7 ± 94	707.7 ± 66	679.2 ± 71	663.8 ± 75	649.1 ± 76	463.6 ± 89
5	797.2 ± 97	773.3 ± 65	739.7 ± 69	716.9 ± 73	680.1 ± 78	504.8 ± 99

Table 8: Policy evaluation at different GA frequency on Spaceship

Test Env. / Steps	1	2	5	10	20	Last
1	0.793 ± 0.09	0.724 ± 0.14	0.630 ± 0.16	0.567 ± 0.16	0.520 ± 0.16	0.203 ± 0.15
2	0.858 ± 0.08	0.833 ± 0.09	0.729 ± 0.14	0.684 ± 0.14	0.664 ± 0.14	0.187 ± 0.13
3	0.872 ± 0.06	0.858 ± 0.06	0.827 ± 0.07	0.800 ± 0.07	0.770 ± 0.09	0.17 ± 0.11
4	0.869 ± 0.06	0.836 ± 0.06	0.822 ± 0.07	0.809 ± 0.07	0.770 ± 0.09	0.211 ± 0.11
5	0.808 ± 0.11	0.789 ± 0.10	0.767 ± 0.11	0.745 ± 0.12	0.707 ± 0.12	0.21 ± 0.13

the optimization of the policy representation by gradient ascent along the direction of increasing values of $\mathbb{V}_\theta$.

Algorithm 2: PAnDR: Value Function Network Approximation

input : Environment encoder ϕ_e, policy encoder ϕ_π^{enc}, training set D and value estimation network $\mathbb{V}_\theta$.

for *each training step* **do**
- Sample episode $\tau = (s_i, a_i, s_{i+1})_{i=0}^{T}$ and episode reward $G(s_0)$;
- Input τ into environment encoder ϕ_e to get environment embedding z_e;
- Input τ into policy encoder ϕ_π^{enc} to get policy embedding z_π;
- Update $\mathbb{V}_\theta$ with equation 7;

E Related Work

E.1 Meta-RL

Meta-RL is meta-learning in the field of RL that the training and testing tasks are drawn from the same family of problems while having different components such as reward probabilities, environmental dynamics and so on. Common meta-RL methods are mainly divided into three classes. The first class is the model-based meta-RL, Duan et al. (2016) uses a RNN structure to record historical information in the agent model and incoporate task-related knowledge into the model parameters. The second class is the optimization-based meta-RL methods. A representative method is (Finn et al., 2017) that use base-learners to collect data on different subtasks and then use a meta-learner to learn from these data. However, both model-based and optimization-based method requires on-policy learning, of which the sampling efficiency is poor. The third class is the context-based methods which propose to train an encoder in multi different environments to extract the environmental information. The context of new environments is inferred using a small amount of data collected during the initial limited interactions. Then the context is used as the input of the policy and value networks to help the agent quickly adjust its policy and adapt to the new environments. The encoder is expected to capture the different task information. In (Rakelly et al., 2019; Fakoor et al., 2020), the encoder networks are learned based on the reward signal. In (Fu et al., 2021), the encoder is trained on the trajectories through contrastive learning. However, training context encoders using reward signals maybe unstable while directly using trajectories may introduce redundant information as the encoder learned based on trajectory may contains policy information. Recently, Raileanu et al. (2020) uses a predictive approach to try to extract environment context and policy context separately, but they are not completely decoupled. In contrast to these methods, our PAnDR can extract and refine policy and environment embeddings to obtain the decoupled representations.

Table 9: Policy evaluation at different GA frequency on Swimmer

Test Env. / Steps	1	2	5	10	20	Last
textbf1	80.0 ± 8	73.9 ± 13	54.5 ± 13	52.8 ± 13	51.0 ± 12	42.0 ± 12
2	105.7 ± 14	104.7 ± 14	86.0 ± 14	85.2 ± 14	83.0 ± 14	74.0 ± 13
3	132.8 ± 15	129.9 ± 13	110.7 ± 14	109.1 ± 14	108.0 ± 14	98.9 ± 13
4	154.3 ± 14	151.4 ± 13	136.2 ± 14	135.1 ± 13	133.4 ± 12	123.4 ± 13
5	167.4 ± 14	164.0 ± 13	149.6 ± 13	148.1 ± 12	145.8 ± 11	134.8 ± 10

Table 10: Policy evaluation at different GA frequency on HalfCheetah

Test Env. / Steps	1	2	5	10	20	Last
1	529.4 ± 62	524.5 ± 45	521.7 ± 50	494.4 ± 49	486.9 ± 46	314.0 ± 92
2	656.8 ± 60	642.1 ± 67	636.7 ± 62	613.8 ± 62	591.4 ± 70	312.2 ± 96
3	816.19 ± 105	781.5 ± 56	681.6 ± 74	645.7 ± 74	578.8 ± 83	272.2 ± 97
4	807.3 ± 138	742.5 ± 94	577.4 ± 104	519.6 ± 108	410.2 ± 116	93.6 ± 126
5	803.9 ± 147	754.5 ± 101	463.6 ± 152	392.6 ± 147	280.9 ± 129	-91.2 ± 146

E.2 Representation Learning

Due to the high complexity of the policy space in RL, it is difficult to directly optimize the policy in this space. Therefore, a possible way is to map the original high-dimensional policy space to a low-dimensional representation space and optimizing the policy in this representation space. In this way, one can greatly reduce the difficulty of searching for the optimal policy in the policy space. Some recent works focus on the policy representation and use the policy representation as auxiliary input to the value function. For example, PDVF (Raileanu et al., 2020) used this idea in meta-RL. It's trained by using state and policy representations to predict corresponding action. Harb et al. (2020) uses the actions that policy sampled in probing states as policy representations. There are also some articles (Faccio et al., 2021; Tang et al., 2020) that use policy network parameters as policy representations. Then given the policy representation, by gradient ascent, the policy representation can be updated along the direction of increasing the values of value function. In this way, one can find an approximated optimal policy without the overhead of interacting with the environment. However, how to characterize the policy in a reasonable and efficient way and whether the policy representation can improve the meta-RL is still an open challenge. In contrast to policy representation, environment representation is usually applied in context-based meta-RL. There are two common ways to extract envionment representations. One way is to predict future information based on the information of current steps (Raileanu et al., 2020; Guo et al., 2020). The other way is to extract representations based on contrastive learning (Fu et al., 2021). The goal of contrastive learning is to learn an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. A metric is then given to measure the similarity of the feature representations of these samples and a discriminative encoder is trained to group data with similar embedding while distinguish data with dissimilar embedding. Then the learned representations are used for downstream tasks. PAnDR also use contrastive learning to extract representations. We follow the training framework and the infoNCE loss used in CURL (Laskin et al., 2020). But we take a different approach than CURL to construct positive and negative examples. Details are given in Sec. 3.

Algorithm 3: PAnDR: Policy Adaptation via Representation Gradient Ascent

input : Environment encoder ϕ_e, policy encoder ϕ_π^{enc}, policy decoder ϕ_π^{dec}, value estimate network $\mathbb{V}_\theta$ and a initial policy π.

Sample a trajectory $\tau = (s_i, a_i, s_{i+1})_{i=0}^K$ using policy π;
Input τ into environment encoder ϕ_e to get environment embedding z_e^{on};
Input τ into policy encoder ϕ_π^{enc} to get policy embedding z_π^0;
for $t = 0, ..., N$ **do**
  Update policy embedding: $z_\pi^{t+1} = z_\pi^t + \eta \nabla_{z_\pi^t} \mathbb{V}_\theta(s, z_e^{\text{on}}, z_\pi^t)$;
Using policy decoder ϕ_π^{dec} based on z_π^* to output action to interact with new environment.

Table 11: Hyperparameters Setting of PPO

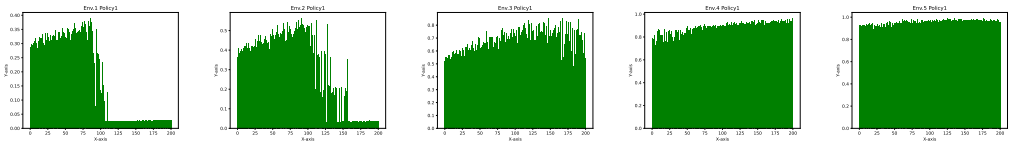
Hyperparameters	**Value**
Batch size	32
Training steps	1e6
ϵ	0.2
Learning rate	3e-4
γ	0.99
Number of trajs. collected under each Env.	50

Table 12: Hyperparameters Setting of Encoder Networks

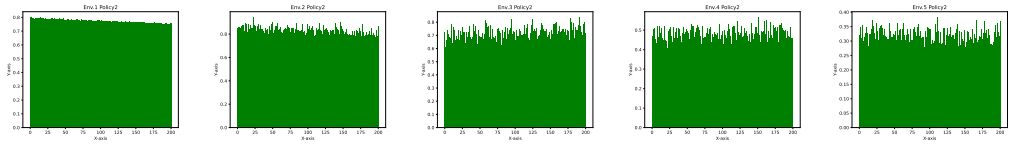
Hyperparameters	**Value**
Training episodes	3000
ϕ_π learning rate	0.01
ϕ_e learning rate	0.001
ψ_1 learning rate	0.005
ψ_2 learning rate	0.005
The weight of MIN MI loss α	1000
The weight of MAX MI loss β	1
The dimension of z_e	8
The dimension of z_π	8
The dimension of z_b	8

Table 13: Hyperparameters Setting of Value Approximation and Policy Adaptation

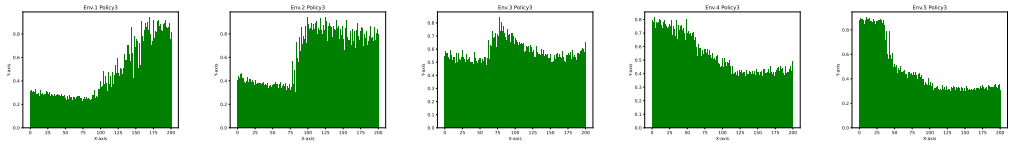
Hyperparameters	**Value**
Training episodes	3000
Learning rate	0.005
Batch size	128
The steps for gradient ascent in adaptation	100



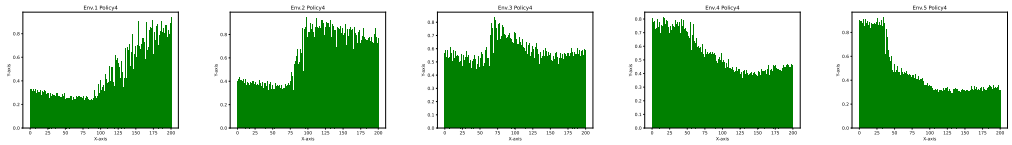
(a) Policy 1 Env. 1 to Env. 5



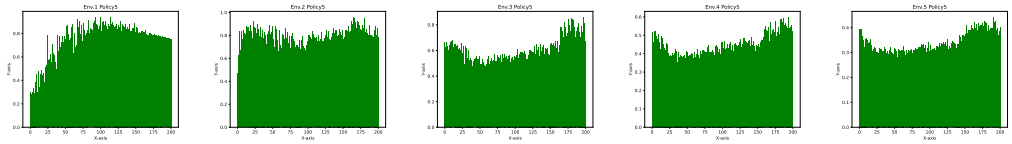
(b) Policy 2 Env. 1 to Env. 5



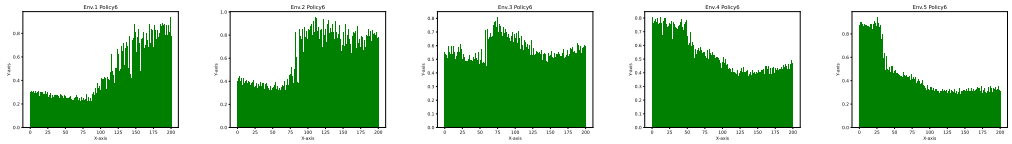
(c) Policy 3 Env. 1 to Env. 5



(d) Policy 4 Env. 1 to Env. 5

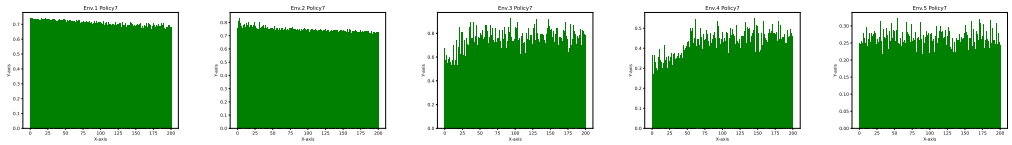


(e) Policy 5 Env. 1 to Env. 5

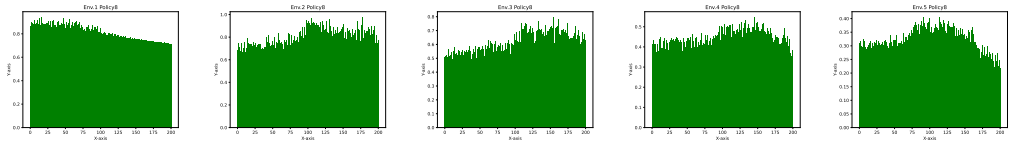


(f) Policy 6 Env. 1 to Env. 5

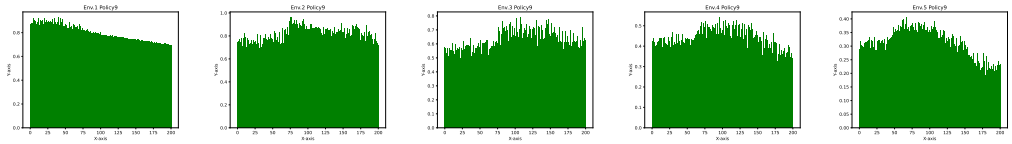
Figure 8: Episodic MC return distribution of adapted policy during GA process.



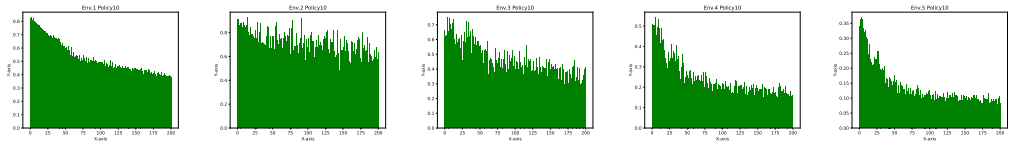
(g) Policy 7 Env. 1 to Env. 5



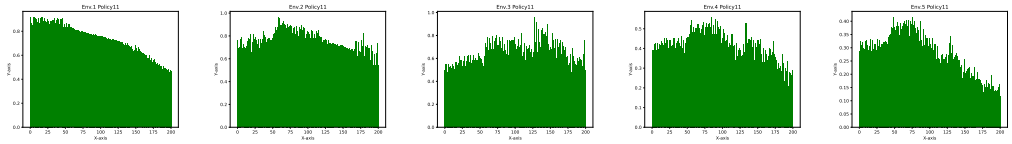
(h) Policy 8 Env. 1 to Env. 5



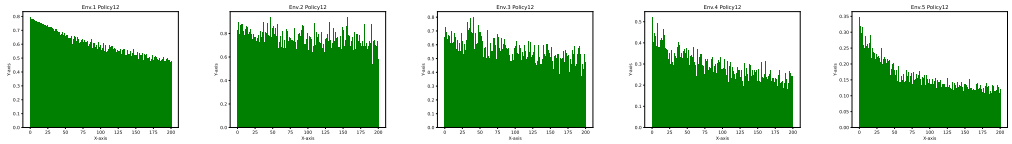
(i) Policy 9 Env. 1 to Env. 5



(j) Policy 10 Env. 1 to Env. 5

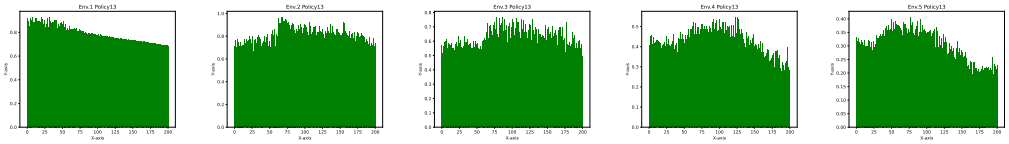


(k) Policy 11 Env. 1 to Env. 5

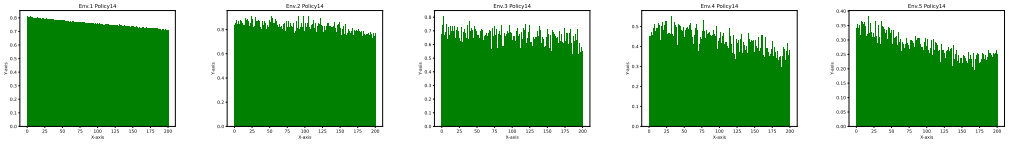


(l) Policy 12 Env. 1 to Env. 5

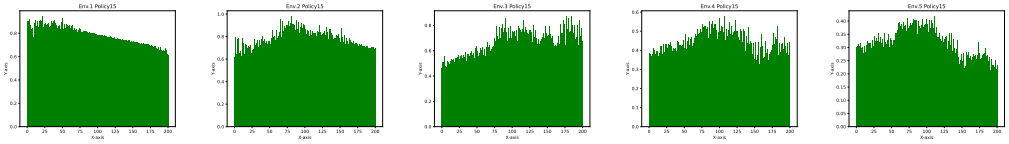
Figure 8: Episodic MC return distribution of adapted policy during GA process.



(m) Policy 13 Env. 1 to Env. 5



(n) Policy 14 Env. 1 to Env. 5



(o) Policy 15 Env. 1 to Env. 5

Figure 8: Episodic MC return distribution of adapted policy during GA process.