Adapt to Environment Sudden Changes by Learning a Context Sensitive Policy

Fan-Ming Luo,[1] **Shengyi Jiang,**[1] **Yang Yu,**[1, 2*] **Zongzhang Zhang,**[1, 3] **Yi-Feng Zhang** [1]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]Polixir Technologies, Nanjing 210038, China
[3]Alibaba Group, Hangzhou 310052, China
{luofm, jiangsy}@lamda.nju.edu.cn, {yuy, zzzhang}@nju.edu.cn, zhangyf@lamda.nju.edu.cn

Abstract

Dealing with real-world reinforcement learning (RL) tasks, we shall be aware that the environment may have sudden changes. We expect that a robust policy is able to handle such changes and adapt to the new environment rapidly. Context-based meta reinforcement learning aims at learning environment adaptable policies. These methods adopt a context encoder to perceive the environment on-the-fly, following which a contextual policy makes environment adaptive decisions according to the context. However, previous methods show lagged and unstable context extraction, which are hard to handle sudden changes well. This paper proposes an environment sensitive contextual policy learning (ESCP) approach, in order to improve both the sensitivity and the robustness of context encoding. ESCP is composed of three key components: *variance minimization* that forces a rapid and stable encoding of the environment context, *relational matrix determinant maximization* that avoids trivial solutions, and a *history-truncated recurrent neural network* model that avoids old memory interference. We use a grid-world task and 5 locomotion controlling tasks with changing parameters to empirically assess our algorithm. Experiment results show that in environments with both in-distribution and out-of-distribution parameter changes, ESCP can not only better recover the environment encoding, but also adapt more rapidly to the post-change environment ($10\times$ faster in the grid-world) while the return performance is kept or improved, compared with state-of-the-art meta RL methods.

Introduction

Reinforcement learning (RL) (Sutton and Barto 1998) is powerful for sequential decision-making (Mnih et al. 2015; Schrittwieser et al. 2020). It learns an optimal policy via trial-and-errors in a given environment. Meta RL learns adaptive policies and is able to adapt to unseen environments within few steps of updates (Duan et al. 2016). In recent years, meta RL has achieved significant success in the tasks that have various scenarios (Kumar et al. 2021; Zhang, Yu, and Zhou 2018).

Despite the fact that meta RL is able to adapt to diverse environments, existing methods implicitly require intra-episode stationarity of the environments, i.e., the environments are stationary within an episode. Specifically, transition function of such environments never changes within an episode. However, this property is unlikely to hold in real-world environments where the transition function can switch from one to another at an arbitrary time step in an episode due to external disturbances. For brevity, we call such two kinds of environments *stationary* and *changing* environments, respectively. Figure 1 illustrates a simple example of changing environment where the agent runs into a puddle of water unexpectedly in an autonomous driving task. To alleviate this risk, quick identification as well as adaptation are essential. A delayed response to the change will lead to severe results (as shown in Figure 1 Left). On the contrary, a fast adaptation ability can help the agent handle this emergency. In this paper, we aim at improving the fast adaptation ability of the agents in the changing environment.

Figure 1: An illustration of sudden changes in an environment. When a car is driving through a water puddle, the water reduces the friction of the road, and the dynamics of the environment suddenly changes. Without the ability of fast adaptation for the new dynamics, an RL trained driving policy will lose control (left). In contrast, we expect the policy can adapt to the environment change rapidly and handle this emergency (right).

Context-based meta RL methods (Rakelly et al. 2019; Yu et al. 2017) have the potential to solve this task. They are usually composed of (1) a context encoder that extracts environment-related information from the environment interactions and (2) a contextual policy makes decisions based on the environment context. Since these two components are functionally decoupled, the adaptation makes fast as long as the context encoder identifies the changed context fast. However, previous context-based methods need many interactions with the changed environment to recognize its changes, which results in a slow adaptation speed. Here, the *adapta-*

*Yang Yu is the corresponding author.

tion speed is quantified by the number of interactions the policy needs to choose the optimal action in the post-change environment. The slow adaptation speed of existing methods finally leads to an overall poor performance of them in changing environments.

In this paper, we propose Environment Sensitive Contextual Policy learning (ESCP) to enable the agents to recognize and adapt to sudden environment changes rapidly. ESCP follows the paradigm of context-based meta RL and consists of three key components: (1) *variance minimization* that forces the environment context encoder to make a fast and robust encoding of the environment; (2) *relational matrix determinant maximization* that prevents the encoder from generating trivial encodings and makes the encodings separable; (3) a *history-truncated recurrent neural network* (RNN) model that forces the context encoder to focus on the most recent experiences, which are closely correlated to the environment changes. With these components, ESCP is able to fast recognize a sudden environment change based on the most recent interactions by the context encoder. The contextual policy can then adapt to the change rapidly. Empirically, we test ESCP in a set of environments that have sudden changes: a grid-world and 5 locomotion controlling tasks. Results show that in environments with both in-distribution and out-of-distribution (OOD) parameter changes, ESCP can better recover the environment encoding and adapt more rapidly to the changing environment ($10\times$ faster in the grid-world), compared with the state-of-the-art (SOTA) meta RL methods, meanwhile keeps or improves the return performance. Ablation studies also disclose that the components of ESCP are essential.

Background

Preliminaries

An RL task is often formalized as a Markov decision process (MDP), described by a tuple $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, p is the transition distribution that maps (s_t, a_t) to s_{t+1} with $p(s_{t+1} \mid s_t, a_t)$, $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0,1)$ is the discount factor, and $\rho_0(s_0)$ is the initial state distribution. At each time step t, the RL agent observes a state s_t and chooses an action a_t following a policy $\pi(a_t \mid s_t)$. Then the agent will observe a new state s_{t+1} following $p(s_{t+1} \mid s_t, a_t)$, and get an immediate reward $r(s_t, a_t)$. $U_t = \sum_{i=0}^{\infty} \gamma^i r(s_{t+i}, a_{t+i})$ is the discounted accumulated reward, a.k.a. return. The value function $V^\pi(s) = \mathbb{E}_\pi [U_t \mid s_t = s]$ is the expectation of U_t at state $s_t = s$. Similarly, the action-value function $Q^\pi(s,a) = \mathbb{E}_\pi [U_t \mid s_t = s, a_t = a]$ is the expectation of U_t when $s_t = s, a_t = a$. The objective of RL is to find a policy π that maximizes the expectation of return, i.e., $\max_\pi \mathbb{E}_{\rho_0(s_0)} [V^\pi(s_0)]$.

Related Works

Meta RL (Duan et al. 2016; Yu et al. 2018) is able to learn an adaptive policy and has potentials to handle the adaptation problem. Meta RL methods usually discover the common task structures (a.k.a. meta-knowledge) from various environments. Then they will utilize the meta-knowledge to fast adapt to a new environment. Different meta RL methods vary in the meta knowledge they learned.

The meta RL methods based on MAML (Finn, Abbeel, and Levine 2017) usually learn a pre-trained model from a set of training environments and adapt the pre-trained model to a new environment via several policy gradient steps (Rothfuss et al. 2019; Stadie et al. 2018; Nagabandi et al. 2019; Clavera et al. 2018). These methods need additional model updates in the new environment. Meanwhile, the data used for model updating is typically at least one trajectory, which will slow down the adaptation speed and make it hard to track environment changes.

Another category of meta RL is memory-based methods. These methods often maintain a memory buffer (Santoro et al. 2016) or adopt RNN architectures to facilitate the policies (Duan et al. 2016; Wang et al. 2016; Kirsch and Schmidhuber 2020) or help construct the optimization objective of the policies (Houthooft et al. 2018; Kirsch, van Steenkiste, and Schmidhuber 2020). These methods utilize the past environment interactions to update the policy parameters either explicitly (updating the network parameters) or implicitly (updating the hidden states of the RNNs). Among the memory-based methods, the context-based methods (Yu et al. 2017) satisfy the suddenly changing environment adaptation problem well. The context-based methods augment the state with an (latent) environment-related context. The policy will adapt to a new environment conditioned on the context (Lee et al. 2020; Yu et al. 2020; Yu, Liu, and Turk 2019; Finn et al. 2017). These methods mainly vary in context extraction. In OSI (Yu et al. 2017), a context encoder is used to predict the environment parameters like ground friction via supervised learning. Context encoders are also introduced in CaDM (Lee et al. 2020) and EPI (Zhou, Pinto, and Gupta 2019), where the context encoder generates a latent environment context that can help a transition model predict the next state more precisely. Meanwhile, in PEARL (Rakelly et al. 2019), a context encoder is learned to help action-value function learning as well as keep the randomness of the encoding. The context encoders in current context-based methods can continuously predict the environment context from recent experiences, thus have the potentials to track the environment changes. However, existing context-based methods need many interactions to recognize an environment and its changes. As a result, they cannot handle sudden changes in a short time. In contrast, ESCP forces the context encoder to recognize the environment as fast as possible, and thus the policy can adapt to a changing environment fast.

Self-Supervised Representation Learning has been used in several RL domains in recent work (Laskin, Srinivas, and Abbeel 2020; Zhang et al. 2021; Stooke et al. 2021). Existing methods mainly utilize a contrastive loss to encode the image data to a low dimensional space, as done in the visual representation learning domain (He et al. 2020; Chen et al. 2020). ESCP employs self-supervised learning to extract environment latent context from environment interactions data rather than learn image embeddings. Specifically, a self-supervised loss is used in ESCP to encode the environment contexts.

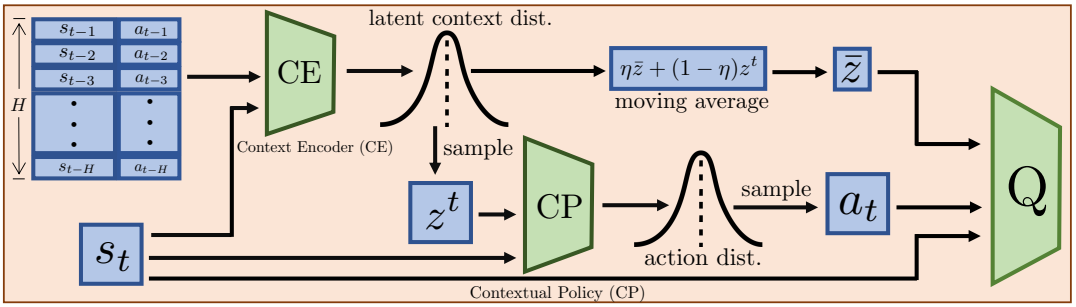

Figure 2: Overview of ESCP with Q as the critic function. A latent context distribution is inferred by a context encoder (CE). A contextual policy (CP) makes decisions according to the latent context and the current state. The action-value function receives a state-action pair and a moving-averaged latent context to infer the Q value.

Problem Setting and Formalization

In a changing environment, the transition distribution depends on an time-dependent environment context $c^t \in \mathcal{C}$, where $\mathcal{C}$ is the context space. Environment contexts could be the length of robot arms or friction of a car's wheels, which influence the transition of the environment. Conditioned on a context c^t, (s_t, a_t) will be mapped to s_{t+1} with probability $p(s_{t+1} \mid s_t, a_t, c^t)$. In stationary environments, contexts are constant when the policy is being executed: $\forall t \geq 0, c^t = c$. Thus, we can use a single environment context to represent a stationary environment. In changing environments, instead, environment contexts may change from one to the another at arbitrary steps, i.e., $c^t \neq c^{t+1}$, for some t.

Although the deployment environment is changing, we still hope the agent can be trained in stationary environments for two reasons: (1) the changes of the environment are always unexpected thus it is intractable to construct all possible changes during training. A meta-knowledge learned in stationary environments will not overfit to certain patterns of changes and be more universal; (2) it is consistent with current training procedure of meta RL methods so a replacement of algorithms is easy.

Based on the above discussion, the changing environment adaptation problem can be formalized as follows. There is a context distribution $\rho(c)$, from which we can draw M contexts: $\{c_0, c_1, \ldots, c_{M-1}\}$ and construct M stationary environments accordingly. By interacting with the environments, we try to train a policy reaching the highest performance in changing environments. During the deployment phase, we sample a context c^d from $\rho(c)$ to get a pre-change environment, and execute the policy in this environment. At some step t, another context c'^d is sampled to generate a new post-change environment. The goal is to maximize the cumulative return across the entire episode, i.e. both pre- and post-change environments. The distribution where c'^d is sampled can be either $\rho(c)$ or another distribution if we want to test the ability of OOD adaptation.

In this problem, policy *adaptation speed* is the key target, which can be quantified by the number of interactions that a policy needs to choose the optimal action in the post-change environment. Similarly, the number of interactions that a context-based policy needs to extract the environment-related latent context can quantify the environment *identification speed*. A fast adaptation or identification speed denotes fewer interactions required to accomplish the adaptation or identification.

Our Method

In this section, we first provide an overview of ESCP, and then propose the 3 key components of ESCP. Finally, we present a practical ESCP algorithm.

Framework

Similar to the previous context-based meta RL methods (Yu et al. 2017; Lee et al. 2020; Peng et al. 2018), ESCP contains two main modules: a context encoder (CE) to encode environment contexts and a contextual policy (CP) to make adaptive decisions conditioned on the encoding. At time step t, the CE receives historical interactions and state s_t, and infers a latent context distribution of the current environment. Afterward, a latent context z^t will be sampled from the distribution and concatenated to the current state s_t. Finally, the CP will predict an action a_t conditioned on the concatenated vector (s_t, z^t). The overview of ECSP with Q as critic function is presented in Figure 2.

The input to the action-value function is also augmented with latent contexts, predicted by the same CE. The action-value function does not need to make any online adaptation and can use the averaged latent contexts to represent environment information because the averaged value is more stable and has a lower variance. Practically, we maintain a set of latent context moving-average vectors $\{\bar{z}_i\} := \{\bar{z}_0, \bar{z}_1, \ldots, \bar{z}_{M-1}\}$ for the M environments. When we need to predict the Q value of a state-action pair sampled from the i-th environment, the pair will be augmented with $\bar{z}_i$ and fed into the Q network.

Consequently, the contextual policy and corresponding action-value function can be represented as $\pi(a \mid s, z; \theta)$ and $Q(s, a, \bar{z}; \psi)$, where θ and ψ are the parameters of the policy and action-value function, respectively.

The training of the CP follows typical SAC (Haarnoja et al. 2018) paradigms by simply substituting the state by the augmented vector (s_t, z^t) or $(s_t, \bar{z}_i)$. Note that other actor-critic algorithms, e.g. TRPO (Schulman et al. 2015), PPO

(Schulman et al. 2017), and TD3 (Fujimoto, van Hoof, and Meger 2018), can also be integrated into ESCP. The CE is fixed during the training of CP. The optimization of the CE will be discussed in the next sub-section.

Optimization of the Context Encoder

The context encoder (CE) encodes the historical interactions to latent contexts, and the contextual policy (CP) makes decisions based on the learned latent contexts. After the CE produces the environment-related encodings, an optimal CP can make optimal decisions in the environment accordingly. Hence, if the CE can identify and track the sudden changes of the environment fast, the CP can also adapt to the environment fast. We expect the CE can identify an environment as fast as possible so as to recognize a sudden change in time.

Suppose the latent context vector of the i-th environment is denoted as $u_i \in \mathcal{Z}$, where $\mathcal{Z}$ is the space of the latent contexts. To force the CE to predict u_i fast, we hope $\forall t \geq 0$, z_i^t is close to u_i. Thus, we can use the L2 distance between u_i and z_i^t as the optimization objective:

$$\mathcal{L}^i_{\text{distance}} = \mathbb{E}[\|z_i^t - u_i\|_2^2], \tag{1}$$

where $\mathbb{E}[\cdot] \coloneqq \mathbb{E}_{z_i^t \sim \mathcal{D}_i}[\cdot]$ for brevity, and $\mathcal{D}_i$ denotes a set of z_i^t predicted from the data collected in the i-th environment. This loss will help CE's output converge to the latent contexts fast at the early time steps and keep consistent for the remaining time steps. By bias-variance decomposition, we have

$$\mathcal{L}^i_{\text{distance}} = \mathbb{E}[\|z_i^t - \mathbb{E}[z_i^t]\|_2^2] + \|\mathbb{E}[z_i^t] - u_i\|_2^2.$$

Consequently, the CE optimization objective can be divided into two parts: the former part is the variance objective that regards the expectation of z_i^t as the latent context of the i-th environment and forces z_i^t to converge to it fast and stably. The latter part pushes the expectation of z_i^t to u_i.

While it is easy to construct the variance loss, u_i in the latter part is undetermined yet. We expect u_i to be separable and able to represent the environments uniquely. The choice of u_i is various as long as it can separate different environments and prevent the variance minimization process from generating trivial solutions. This condition can be satisfied by a set of u_i with large diversity. The problem is then transformed to finding a latent context vector set $\{u_i \in \mathcal{Z} \mid i = 0, 1, \ldots, M-1\}$ that has the largest diversity: $\{u_i^\star\} = \arg\max_{\{u_i\}} \text{DIV}(\{u_i\})$, where $\text{DIV}(\cdot)$ measures the diversity of a vector set. In Determinant Point Process (DPP) (Kulesza and Taskar 2012), the diversity of a vector set can be measured by the determinant of a relational matrix constructed from the set: $\text{DIV}(\{u_i\}) = \det(R_{\{u_i\}})$, where $\det(\cdot)$ denotes the matrix determinant, $R_{\{u_i\}}$ is the relational matrix of $\{u_i\}$, and each element of $R_{\{u_i\}}$ is:

$$R_{\{u_i\}}(i,j) = d(u_i, u_j), \;\; i, j \in \{0, 1, \ldots, M-1\}. \tag{2}$$

Here, d measures the correlations between two vectors and could be a radius basis function $d(u_i, u_j) = \exp(-\alpha\|u_i - u_j\|_2^2)$, where α is a hyper-parameter controlling the radius. To understand such an objective intuitively, we can consider two extreme scenarios: all u_i's are identical or irrelevant. In the former case, each entry of $R_{\{u_i\}}$ is 1 thus $\det(R_{\{u_i\}})$ is 0. In the latter case, $R_{\{u_i\}}$ is an identity matrix thus $\det(R_{\{u_i\}})$ is 1. Note that 0 and 1 are the lower and upper bounds of $\det(R_{\{u_i\}})$ and correspond to the smallest and largest diversity, respectively. We follow this idea and determine u_i by relational matrix determinant maximization:

$$\{u_i^\star\} = \arg\max_{\{u_i\}} \; \log\det\left(R_{\{u_i\}}\right). \tag{3}$$

We need to obtain u_i by Eq. (3) and push $\mathbb{E}[z_i^t]$ to u_i. However, notice that u_i is an intermediate variable that is used to guide $\mathbb{E}[z_i^t]$, we can simplify the optimization process by eliminating u_i. The objective of $\mathbb{E}[z_i^t]$ can be directly represented as $\max_{\{\mathbb{E}[z_i^t]\}} \log\det(R_{\{\mathbb{E}[z_i^t]\}})$. Together with the variance minimization objective, we get the CE's objective:

$$\mathcal{L}_{\text{CE}} = \lambda \textstyle\sum_{i=0}^{M-1} \mathbb{E}\left[\|z_i^t - \mathbb{E}[z_i^t]\|_2^2\right] - \log\det(R_{\{\mathbb{E}[z_i^t]\}}), \tag{4}$$

where λ is a regularization factor controlling the weight of the variance loss. Practically, the expectation in Eq. (4) is estimated by the average value of a batch of data. However, a small batch size will lead to a large variance of the average value, which will make the target of z_i^t in the first term unstable. To improve the stability of the training, we estimate $\mathbb{E}[z_i^t]$ in Eq. (4) with the moving average latent context $\{\bar{z}_i\}$, which can be regarded as a weighted average of all past data and has lower variance. With a batch of latent contexts $\mathcal{B}_z$, we will first update the moving average latent context $\bar{z}_i$ by:

$$\bar{z}_i \leftarrow \eta\,\mathtt{sg}(\bar{z}_i) + (1-\eta)\mathbb{E}_{z_i^t \sim \mathcal{B}_z}\left[z_i^t\right], \tag{5}$$

where $\eta \in [0, 1)$ is a hyper-parameter to control the moving-average horizon, $\mathtt{sg}(\cdot)$ denotes gradient stop, for which auto-gradient tools will discard the gradient of $\bar{z}_i$ w.r.t. the CE parameters. As a result, the gradient of $\mathtt{sg}(\bar{z}_i)$ w.r.t. the parameters of the CE is always zero, while the gradient of $\bar{z}_i$ is not. Then, the empirical CE loss can be written as:

$$\hat{\mathcal{L}}_{\text{CE}} = \lambda \textstyle\sum_{i=0}^{M-1} \mathbb{E}_{z_i^t \sim \mathcal{B}_z}\left[\|z_i^t - \mathtt{sg}(\bar{z}_i)\|_2^2\right] - \log\det(R_{\{\bar{z}_i\}}). \tag{6}$$

History-Truncated Context Encoder

The context encoder is instantiated by a recurrent neural network to capture the latent context from recent experiences. At time step t, in the i-th environment, the predicted latent context z_i^t can be written as:

$$z_i^t \sim g(\tau_t, s_t; \phi),$$

where $g(\tau_t, s_t; \phi)$ is the RNN network, and $\tau_t = \{s_0, a_0, \ldots, s_{t-1}, a_{t-1}\}$. $g(\tau_t, s_t; \phi)$ predicts the environment latent contexts along the full history. However, in the environments that could change suddenly, the early observations cannot help the identification for the changes and the post-change environments. Because the environment could have changed and the early experiences might correspond to the former environment contexts, which could even injure the environment encoding for the post-change environment.

To prevent the old interaction memories from interfering the context encoding, the history fed into the CE is truncated.

Algorithm 1: Training Process of ESCP

Input: Training environments $\{p_0, \dots, p_{M-1}\}$, context encoder $g(z^{t+1} \mid \tau_{t-H:t}, s_t; \phi)$, contextual policy $\pi(a_t \mid s_t, z^t; \theta)$, Q function $Q(s_t, a_t, \bar{z}; \psi)$, RNN history length H, moving-average context $\{\bar{z}_0, \dots, \bar{z}_{M-1}\}$, and replay buffer $\mathcal{R}$.

for *step* $= 0, 1, 2, \dots$ **do**
  Sample env. i from $\{0, 1, \dots, M-1\}$ uniformly;
  $t \leftarrow 0$;
  while *not done* **do**
    Sample latent context z_i^t following Eq. (7);
    Sample a_t from $\pi(a_t \mid s_t, z_i^t; \theta)$;
    Apply a_t to env. p_i, then obtain s_{t+1} and r_t;
    Append $\{s_t, a_t, s_{t+1}, r_t, i\}$ to $\mathcal{R}$;
    if *need to update parameter* **then**
      Sample a transition batch $\mathcal{B}$ from $\mathcal{R}$;
      Infer context z for $\mathcal{B}$ from context encoder g;
      Update $\{\bar{z}_0, \dots, \bar{z}_{M-1}\}$ by Eq. (5) with $\mathcal{B}$;
      Update θ, ψ via SAC with $\mathcal{B}$, z, and $\bar{z}$;
      Get CE loss $\hat{\mathcal{L}}_{\text{CE}}(\phi)$ by Eq. (6) with $\mathcal{B}$, z, $\bar{z}$;
      Update ϕ with $\hat{\mathcal{L}}_{\text{CE}}(\phi)$ via Adam optimizer;
    $t \leftarrow t + 1$;

CE only uses the most recent H interactions to predict the latent context. Formally, CE predicts the latent context z_i^t by

$$z_i^t \sim g(\tau_{t-H:t}, s_t; \phi), \tag{7}$$

where $\tau_{t-H:t} = \{s_{t-H}, a_{t-H}, \dots, s_{t-1}, a_{t-1}\}$. When predicting a latent context, CE will set the initial hidden state of RNN to zero. Then the most recent H transitions will be fed into the CE sequentially. The final (the H-th) output of the CE is used as the current predicted latent context.

The ESCP Algorithm

The context encoder (CE) and the contextual policy (CP) are instantiated by gated recurrent units (GRUs) (Cho et al. 2014) and a fully connected neural network, respectively. The latent context distribution is a Gaussian distribution, whose mean is the output of the CE and variance is fixed. Before sampling a trajectory, an environment will be sampled uniformly from a training environment set. When the agent interacts with each environment, the transition and the current environment index will be stored to a replay buffer $\mathcal{R}$. At the parameter optimizing phase, with the data in $\mathcal{R}$, the loss of the CP can be calculated by the means of the SAC algorithm (Haarnoja et al. 2018). Everything concerning CP and action-value function training is identical to SAC except for augmenting the state for CP with pre-calculated latent context z and augmenting the state for action-value function with the moving-averaged context $\bar{z}$. Finally, parameters in the CE network ϕ will be optimized to minimize $\hat{\mathcal{L}}_{\text{CE}}(\phi)$ via Adam (Kingma and Ba 2015). The training process is summarized in Algorithm 1.

Experiments

This section compares ESCP with other methods on 1 grid-world and 5 MuJoCo tasks. Each experiment is repeated by at least 3 seeds and the shaded area indicates standard errors. We release our code at Github[1].

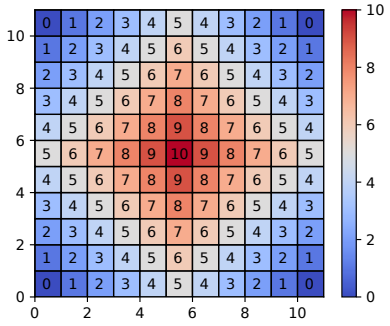

Figure 3: Illustration of the grid-world environment. This environment is an 11×11 grid-world, where each cell is marked with its reward.

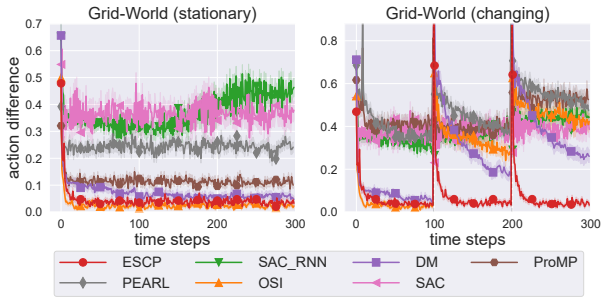

Figure 4: Action differences of the learned policy to the optimal policy. Left: Differences in stationary environments. Right: Differences in suddenly changing environments.

Baselines

In the following experiments, we compare ESCP with some existing algorithms, as below.

- *SAC* (Haarnoja et al. 2018) and *SAC-RNN*, a SOTA RL algorithm and its variant with RNN policy and Q networks.
- *DM* (Lee et al. 2020; Zhou, Pinto, and Gupta 2019), which learns the CE to help a dynamic model (DM) making a more precise prediction of next state.
- *OSI* (Yu et al. 2017), which learns the CE to predict the real environment contexts via supervised learning.
- *PEARL* (Rakelly et al. 2019), which learns a probabilistic fully-connected CE module. The objective of CE is making a more precise prediction of the action value.
- *ProMP* (Rothfuss et al. 2019), which does not adopt a CE module and learns a pre-trained policy from the training environments. It will adapt to testing environments by several steps of policy parameters updates.

[1] https://github.com/FanmingL/ESCP

Results in Grid-World

We first conduct experiments on a manually designed grid-world environment. We present the environment in Figure 3. In this task, the state space is all possible coordinate (x, y) of the grids. The action space is $\{(x_a, y_a) \mid x_a, y_a \in \{-3, -2, \ldots, 3\}\}$, where x_a and y_a denote the sizes that the agent moves along x and y axis, respectively. There is a pair of environment contexts that the agent cannot observe: $\{(x_w, y_w) \mid x_w, y_w \in \{-2, -1, \ldots, 2\}\}$. (x_w, y_w) is used to simulate a wind that influences the transition function. The reward at each grid is marked in Figure 3. After an agent takes action (x_a, y_a) at state (x, y) in an environment with context of (x_w, y_w), it will move to a new state:

$$\texttt{in\_range}\{(x + x_a + x_w, y + y_a + y_w)\},$$

where `in_range` denotes constraining the position in the grid-world to prevent the agent from being out of bound. In such an environment, the agent needs to identify the wind and makes optimal decisions accordingly.

At the training phase, we sample 16 environments with different contexts. At the testing phase, we also sample 16 environments make them change suddenly. In the suddenly changing environments, the environment contexts change suddenly for every 100 steps. The post-change environment is sampled from the same distribution uniformly.

Figure 4 depicts the action difference of the learned policy to the optimal policy at each step in both stationary and changing environments. At the 100-th and 200-th time step, in changing environments, the action difference increases suddenly as the environment changes. With the ability of fast adaptation, ESCP can reduce its action difference to 0.1 within 10 steps in the changing environments. On the contrary, none of other methods can control their action difference to 0.1 within 100 steps in the changing environments. This result implies that the policy adaptation speed of ESCP is $10\times$ faster than the previous baselines in the changing environments. OSI, PEARL, and DM explicitly consider the environment context in their training objective. As a result, in stationary environments, they outperform SAC and SAC-RNN which are not aware of the environment contexts. However, when the environments change suddenly, they require a long time to adapt to the post-change environment. If two changes continuously occur, PEARL and OSI are even inferior to SAC. ProMP which is updated from its pre-trained model also has low action differences in stationary environments but fails to adapt to changing environments.

Such a fast adaptation ability leads to a better performance in suddenly changing environments. Figure 5 plots the learning curves in terms of return in stationary and suddenly changing environments. In stationary environments, ESCP is comparable with OSI and DM. When it comes to the suddenly changing environments, the performances of both methods drop a lot and are much lower than ESCP.

Results in MuJoCo Tasks

We also made comparisons on 5 MuJoCo tasks (Todorov, Erez, and Tassa 2012; Brockman et al. 2016), which are more complicated than the grid-world task. In these tasks,

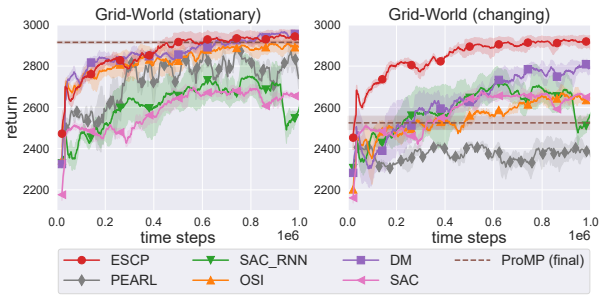

Figure 5: Learning curves in the grid-world environment. Left: Return in stationary environments. Right: Return in in-distribution changing environments.

we choose gravity as the changing factor to simulate the load changing, which is common in robotics applications. Suppose there is a robotic arm lifting a plate, its load increases suddenly after we put something on the plate. Such a scenario can be simulated by changing the gravity size. We change the environment contexts every 100 steps in changing environments.

Latent Contexts Visualization. In MuJoCo tasks, ESCP encodes the environment contexts to two-dimensional vectors. We present the visualization results of the encodings in Figure 6. In the top sub-figures, we regard the latent context vector of each environment as a coordinate (x, y). We scatter the points onto a two-dimensional space and color them according to the context of each environment. The corresponding real context value of the colors is shown in the color bar. Here, the real context is the `gravity` which has been normalized by the gravity sampling range. The latent contexts in both in-distribution and OOD environments are plotted. The bottom sub-figures show the latent context changes in a changing environment. In them, the labels *x* and *y* denote the two dimensions of the predicted latent context vectors, and *real* is the value of the real environment context. As the environment context suddenly changes every 100 steps, the real context will change every 100 steps.

The results in Figure 6 imply ESCP has 4 advantages concerning context encoding. (1) In the top sub-figures, ESCP extracts meaningful latent context information from the historical interactions. The environments whose contexts are similar are also close in the context space. (2) We find the latent contexts can be extended to the unseen and OOD environments in the same pattern. This result shows that the CE in ESCP can even provide informative latent contexts to the CP in OOD environments. (3) From the bottom sub-figures, we can find that the latent contexts are also state-agnostic and robust. The encoding is only related to the environment rather than the state. Before environment changes, although the state changes continuously, the contexts stay stable and flat. Note that the latent contexts are not identical to the real context because the latent contexts are latent representations and do not have realistic meaning. (4) The bottom sub-figures also imply that ESCP can recognize and track the environment changes fast. When the environment changes, the encoding will respond immediately and converge within

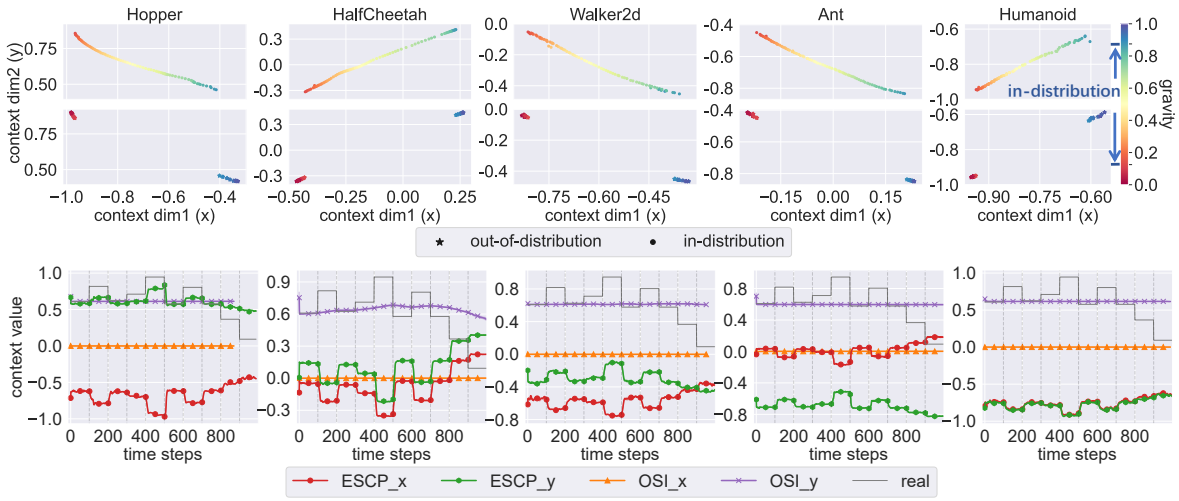

Figure 6: Encoding visualization. Top: Two dimensional encodings for in-distribution and OOD environments. Coordinate of each point (x, y) denotes a two-dimensional encoding of an environment. Color of each point corresponds to the normalized `gravity` of the environment. Bottom: Latent context curves in a single trajectory. Here, x and y denote the two dimensions of latent context vectors, respectively, and "*real*" shows the value of the real context.

Environment type	ESCP	OSI	SAC-RNN	DM	PEARL	ProMP
Stationary environments	$\mathbf{7821 \pm 31}$	6856 ± 44	6982 ± 136	6070 ± 147	5261 ± 4	2233 ± 91
ID changing environments	$\mathbf{7237 \pm 53}$	5197 ± 375	5997 ± 141	5633 ± 9	4605 ± 216	1217 ± 40
OOD changing environments	$\mathbf{6658 \pm 325}$	5203 ± 94	6101 ± 197	5285 ± 29	4251 ± 158	792 ± 65

Table 1: Final average return $\pm$ standard error in HalfCheetah. The results in stationary and changing environments are reported.

10 time steps. In contrast, OSI can precisely predict the real environment context within 3 steps initially but fail to track the environment changes. Such characteristics of the encodings can help the policy adapt to in-distribution and OOD environment changes rapidly and be stable when the environment is temporarily stationary.

Comparisons of Return in (OOD) Changing Environments. In order to study the bonus brought by the encodings, we compare the final return of the policies in MuJoCo tasks. In the OOD environments, the post-change environments are sampled from a distribution that is 33% broader than the training distribution. The table shows that ESCP significantly outperforms other methods in stationary and (OOD) changing environments. In stationary environments, ESCP outperforms the best baseline (SAC-RNN) by 839, while in changing environments, the gap between ESCP and the best baseline increases to 1240. Moreover, the transferring from the stationary environments to the changing environments results in 584 performance drop for ESCP. However, the best baseline in stationary environments (SAC-RNN) suffers a return drop of 985, which is 68.7% larger than ESCP. We notice that OSI, which is near to the highest baseline in the stationary environments, has the largest performance drop. Such a drop results from its encodings, which are insensitive to the environment changes as shown in Figure 6. After environments change, the context encoder of OSI still outputs the encoding that corresponding to the pre-change environment. It might choose the actions that can reach high returns in the pre-change environments but fail in the post-change environments. These results reveal that ESCP is more robust to the environment changes. We believe such a robustness comes from the informative and environment-sensitive encodings. Still with the context encoder which is able to identify the OOD environments, ESCP can also obtain the highest return in OOD suddenly changing environments.

Ablation Studies

In this part, we analyze the effect of each module in ESCP via ablation studies. Figure 7 shows the learning curves in the HalfCheetah concerning the context encoder loss $\hat{\mathcal{L}}_{\text{CE}}$ and the memory length. Without the encoder loss, the context encoder will be optimized by policy gradient. This method shows a large performance drop, which reveals that the encoder loss significantly improves the performance of ESCP. We also substitute the CE loss with a CURL contrastive loss (Laskin, Srinivas, and Abbeel 2020). In this method, the encodings corresponding to the same/different environments are expected to be classified to the same/different class by a simple classifier. It does not contain an explicit variance minimization objective, which is essential in the changing environments. We find it converging faster than the method without the CE loss but also suffering a performance drop in terms of asymptotic performance.

Environment type	ESCP	OSI	SAC-RNN	DM	PEARL
Stationary environments	$\mathbf{6114 \pm 236}$	6105 ± 448	5426 ± 811	5319 ± 460	5168 ± 280
ID changing environments	$\mathbf{5374 \pm 158}$	4241 ± 622	4505 ± 569	4653 ± 537	4813 ± 243
OOD changing environments	$\mathbf{5234 \pm 868}$	4775 ± 433	3667 ± 140	4286 ± 77	4908 ± 147

Table 2: Final average return $\pm$ standard error in HalfCheetah. The changeable parameters are `dof_damping` and `gravity`. The results in stationary and (OOD) changing environments are reported.

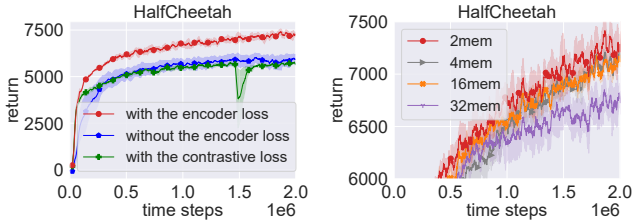

Figure 7: Ablation studies in HalfCheetah. Left: On the context encoder loss. Right: On the memory length.

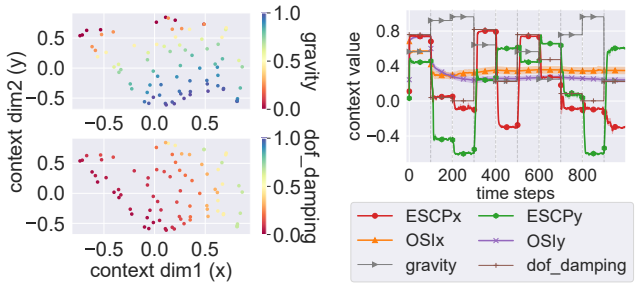

Figure 8: Encoding visualization in HalfCheetah, where `gravity` and `dof_damping` are changeable. Left: Two dimensional encodings. We plot the two dimensional encodings of different environments and color them according to the normalized `gravity` (top) and `dof_damping` (bottom). Right: Latent context curves in a single trajectory. Here, x and y denote the two dimensions of the encodings, respectively.

In the right sub-figure, we study the sensitivity of the memory length used to predict the latent contexts. We can find that there is no significant difference between the training curves. But the algorithm that uses the memory length of 2 has the higher final performance and faster convergence speed than the one that uses 32. This verifies our argument that a longer memory length will influence the policy performance.

Performance with More Changeable Contexts

In addition to `gravity`, we also compare ESCP with the baselines in the environments which change `gravity` and `dof_damping` simultaneously. The recovered encodings are presented in Figure 8. Like Figure 6, we scatter the two dimensional encodings onto a figure and color them according to the environment contexts. Because the dimension of the changeable environment contexts is 2, we color the points twice according to different contexts (top and bottom sub-figures in the left of Figure 8). The latent context curves in a changing environment is presented in the right of Figure 8.

From Figure 8, we can make the similar conclusions like Figure 6: (1) The learned encodings are informative. In the latent context space, `dof_damping` has a positive correlation with the first dimension of the contexts (x), while `gravity` has the negative correlation with the second dimension of the contexts (y). The two kinds of the environment contexts are obviously disentangled, which means the CE can still extract informative data when different environment changes are combined. (2) The latent contexts are robust and sensitive to the environment changes. When the environment has not changed, the latent context is stable and plat. When changes occur, ESCP can track the changes fast. In contrast, OSI can predict both `gravity` and `dof_damping` in the early stage but fail to track the changes after the environment changed.

Table 2 shows the results in terms of the return in stationary, in-distribution (ID), and out-of-distribution (OOD) changing environments. We can find ESCP obtains the highest return in all settings. Moreover, in the right of Figure 8, we can find that OSI can predict the environment context precisely before environment changes. As a result, OSI can obtain similar return to ESCP in stationary environments. However, once the environments start changing, the performance drop of OSI is 151% larger than ESCP as its prediction fails. These comparisons imply the latent contexts can also improve the return of ESCP in changing environments with more than one changeable contexts.

Conclusions

In this work, we study the adaptation problem in changing environments. To tackle it, we propose ESCP, which can recognize an unseen and suddenly changing environment by a context encoder and adapt to the change by a contextual policy conditioned on the context encoding. We propose the variance minimization loss to achieve a fast environment recognition and the relational matrix determinant maximization loss to avoid trivial solutions. We also truncate environment history to prevent old memory from influencing the context encoding. We assessed ESCP with in-distribution and OOD parameter changes. Empirically, ESCP can recover informative environment encodings and obtain superior results compared to existing meta RL methods.

Acknowledgements

This work is supported by National Key Research and Development Program of China (2020AAA0107200), NSFC (61921006, 61876119), and Alibaba Group through Alibaba Research Fellowship Program. We thank Rong-Jun Qin, Jing-Cheng Pang, Tian Xu, Xue-Kun Jin, and anonymous reviewers for their constructive advice to improve this paper.

References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI gym. *CoRR*, abs/1606.01540.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. E. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 1724–1734.

Clavera, I.; Rothfuss, J.; Schulman, J.; Fujita, Y.; Asfour, T.; and Abbeel, P. 2018. Model-based reinforcement learning via meta-policy optimization. In *CoRL*, 617–629.

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

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135.

Finn, C.; Yu, T.; Zhang, T.; Abbeel, P.; and Levine, S. 2017. One-shot visual imitation learning via meta-learning. In *CoRL*, 357–368.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *ICML*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 1856–1865.

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

Houthooft, R.; Chen, Y.; Isola, P.; Stadie, B. C.; Wolski, F.; Ho, J.; and Abbeel, P. 2018. Evolved policy gradients. In *NeurIPS*, 5405–5414.

Kingma, D. P.; and Ba, J. 2015. ADAM: A method for stochastic optimization. In *ICLR*.

Kirsch, L.; and Schmidhuber, J. 2020. Meta learning backpropagation and improving it. *CoRR*, abs/2012.14905.

Kirsch, L.; van Steenkiste, S.; and Schmidhuber, J. 2020. Improving Generalization in Meta Reinforcement Learning using Learned Objectives. In *ICLR*.

Kulesza, A.; and Taskar, B. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2-3): 123–286.

Kumar, A.; Fu, Z.; Pathak, D.; and Malik, J. 2021. RMA: Rapid motor adaptation for legged robots. In *RSS*.

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

Lee, K.; Seo, Y.; Lee, S.; Lee, H.; and Shin, J. 2020. Context-aware dynamics model for generalization in model-based reinforcement learning. In *ICML*, volume 119, 5757–5766.

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

Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2019. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*.

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

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

Rothfuss, J.; Lee, D.; Clavera, I.; Asfour, T.; and Abbeel, P. 2019. ProMP: Proximal meta-policy search. In *ICLR*.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. P. 2016. Meta-Learning with Memory-Augmented Neural Networks. In Balcan, M.; and Weinberger, K. Q., eds., *ICML*, 1842–1850.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; Lillicrap, T. P.; and Silver, D. 2020. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588: 604–609.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Stadie, B. C.; Yang, G.; Houthooft, R.; Chen, X.; Duan, Y.; Wu, Y.; Abbeel, P.; and Sutskever, I. 2018. Some considerations on learning to explore via meta-reinforcement learning. *CoRR*, abs/1803.01118.

Stooke, A.; Lee, K.; Abbeel, P.; and Laskin, M. 2021. Decoupling representation learning from reinforcement learning. In *ICML*, 9870–9879.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT Press.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *IROS*, 5026–5033.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and

Botvinick, M. 2016. Learning to reinforcement learn. *CoRR*, abs/1611.05763.

Yu, W.; Liu, C. K.; and Turk, G. 2019. Policy transfer with strategy optimization. In *ICLR*.

Yu, W.; Tan, J.; Bai, Y.; Coumans, E.; and Ha, S. 2020. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2): 2950–2957.

Yu, W.; Tan, J.; Liu, C. K.; and Turk, G. 2017. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*.

Yu, Y.; Chen, S.; Da, Q.; and Zhou, Z. 2018. Reusable reinforcement learning via shallow trails. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6): 2204–2215.

Zhang, A.; McAllister, R. T.; Calandra, R.; Gal, Y.; and Levine, S. 2021. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*.

Zhang, C.; Yu, Y.; and Zhou, Z. 2018. Learning environmental calibration actions for policy self-evolution. In *IJCAI*, 3061–3067.

Zhou, W.; Pinto, L.; and Gupta, A. 2019. Environment probing interaction policies. In *ICLR*.