Off-Policy Meta-Reinforcement Learning Based on Feature Embedding Spaces

Takahisa Imagawa [1] **Takuya Hiraoka** [1 2 3] **Yoshimasa Tsuruoka** [1 4]

Abstract

Meta-reinforcement learning (RL) addresses the problem of sample inefficiency in deep RL by using experience obtained in past tasks for a new task to be solved. However, most meta-RL methods require partially or fully on-policy data, i.e., they cannot reuse the data collected by past policies, which hinders the improvement of sample efficiency. To alleviate this problem, we propose a novel off-policy meta-RL method, *embedding learning and evaluation of uncertainty* (ELUE). ELUE is characterized by the learning of a shared feature embedding space among tasks. It learns beliefs over the embedding space and a belief conditional policy and Q-function. This approach has two major advantages. It can evaluate the uncertainty of tasks, which is expected to contribute to precise exploration, and it can also improve its performance by updating a belief. We show that our proposed method outperforms existing methods through experiments with a meta-RL benchmark.

1. Introduction

Deep reinforcement learning (DRL) has shown superhuman performance in several domains, such as computer games and board games (Silver *et al.*, 2017; Berner *et al.*, 2019). However, conventional DRL considers only learning for a single task and does not reuse experience from past tasks. This is one of the causes of sample inefficiency in conventional DRL.

Meta-learning has been proposed to overcome this problem (Schmidhuber *et al.*, 1996). Meta-learning is a class of methods for learning how to efficiently learn with a small amount of data on a new task by using previous experience. Meta-learning has two phases: meta-training and meta-testing. In meta-training, the agent prepares for learning in meta-testing. In meta-testing, the agent is evaluated on the basis of its performance on the task to be solved. Although meta-learning aims to improve sample efficiency in meta-testing, the sample efficiency in meta-training is also important in terms of computational cost (Mendonca *et al.*, 2019; Rakelly *et al.*, 2019).

Several meta-learning methods, such as MAML (Finn *et al.*, 2017) and Reptile (Nichol *et al.*, 2018), have been proposed. These methods learn to reduce loss after parameters are updated (e.g., weights of neural networks) over several steps. Finn *et al.* (2017) showed that the sample efficiency of MAML in meta-testing is improved compared with that of naive pretraining. However, most reinforcement learning (RL) applications of these methods need on-policy data (Mendonca *et al.*, 2019), while off-policy methods are more sample efficient because they can reuse data collected by old policies. In addition, the performance of the learned initial parameters in meta-testing may not be good in some cases until the parameters are updated. For example, there are tasks where an agent aims to reach a goal as fast as possible, and the tasks differ in terms of goal positions. Let us assume that there are two tasks in meta-training whose goals are in opposite directions from the initial position of the agent; then, the well-trained policies for the tasks require contradicting actions. Thus, in this case, even if the meta-test task is one of the two tasks, the performance may be poor before the parameters are updated.

PEARL (Rakelly *et al.*, 2019) is another kind of meta learning method that learns how to infer task information in meta-training and uses it in meta-testing. It is based on an idea called "amortized inference" (Srikumar *et al.*, 2012; Stuhlmüller *et al.*, 2013; Liu and Liu, 2019). In this setting, it is assumed that there are many similar tasks to solve and that the agent can offload part of the computational work to shared precomputation (Stuhlmüller *et al.*, 2013). Because of inference, PEARL generally needs less data to improve performance in meta-testing than methods that update the parameters of neural networks. In addition, PEARL's policy and Q-function are trained off-policy, and this also generally further improves sample efficiency.

[1]National Institute of Advanced Industrial Science and Technology, Tokyo, Japan [2]NEC Central Research Laboratories, Kanagawa, Japan [3]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan [4]The University of Tokyo, Tokyo, Japan. Correspondence to: Takahisa Imagawa <imagawa.t@aist.go.jp>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

In some experiments, PEARL showed better sample efficiency than MAML (Rakelly *et al.*, 2019).

In this paper, we extend the idea of PEARL and propose a novel meta-RL method, *embedding learning and evaluation of uncertainty* (ELUE), which has the following features:

Off-policy embedding training
In PEARL, policy training is based on an off-policy method, but the training for task embedding, which is used for calculating distribution over tasks, depends on what the current policy is, i.e., it is on-policy. By dividing the training into training for task embedding and that for policies, we propose a fully off-policy method. Thanks to policy-independent embedding, the training objective is expected to be stable, and data collected by past policies can be reused.

Policy and Q-function conditioned by beliefs over tasks
PEARL introduces a distribution over tasks, but both its policy and Q-function depend on a task variable sampled from the distribution. After the task variable is sampled, the variable contains no information on the uncertainty over the tasks. Instead, in our proposed method, the policy and Q-function are conditioned on the belief over tasks, which can be used to evaluate uncertainty. This leads to more precise exploration as the values that reduce task uncertainty are evaluated.

Combination of belief and parameter update
PEARL does not update the parameters of the policy and Q-function in meta-testing. Thus, PEARL may fail to improve the performance if there are large gaps between the tasks in meta-training and those in meta-testing. To alleviate the gap, our method performs not only inference but also parameter updating.

We compare the performances of PEARL and ELUE through experiments in the Meta-World (Yu *et al.*, 2019) environment. We show that the proposed method performs better than PEARL.

2. Preliminaries

Markov decision processes (MDPs) are models for reinforcement learning (RL) tasks. An MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, T, R, \rho)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and the action spaces respectively, $R : \mathcal{S} \times \mathcal{A} \times \mathbb{R} \to [0, 1]$ is a reward function that determines the probability of the reward amount and $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function that determines the probability of the next state. ρ is the initial state distribution. Let us denote the policy as π, which is the probability of choosing an action at each state. The objective of RL is to maximize the expected cumulative reward, which is discounted by γ, by changing the policy.

We assume that each task in meta-training and meta-testing can be represented by an MDP, where $\mathcal{S}$ and $\mathcal{A}$ are the same among tasks. In addition, we assume that tasks are the same when they differ only in ρ because the difference in ρ does not change the optimal policy. Thus, under these assumptions, a different task means a different T or R.

In our problem, it is assumed that the reward and the transition function are not observable directly. We treat this problem as a partially observable MDP (POMDP) (Humplik *et al.*, 2019; Zintgraf *et al.*, 2020) and introduce a probability over R and T, which is called a "belief". For clarity, let us assume that R and T are parameterized by φ and denote them as R_φ and T_φ. It is known that a POMDP can be transformed into a belief MDP whose states are based on beliefs and that the optimal policy of the belief MDP is also optimal in the original POMDP (Kaelbling *et al.*, 1998). We denote a history as $h_t := (s_0, a_0, r_0, s_1, \ldots, s_t)$, where $s_\tau \in \mathcal{S}$, $a_\tau \in \mathcal{A}$, $r_\tau \in \mathbb{R}$ are the state, the action, and the reward at time τ, respectively. In our problem, a belief at time t is $P(\varphi|h_t)$, and the state of the belief MDP at time t is $s_t^+ = (s_t, P(\varphi|h_t))$, which is often referred to as a hyper-state. The objective of our problem is maximizing $\mathbb{E}_{h_\infty}[\sum_{t=0}^{\infty} \gamma^t r_t]$ by changing a policy which is conditioned on a hyper-state. In our problem, the belief is updated by observations:

$$
\begin{aligned}
&P(\varphi|h_{t+1}) \\
&\propto P(\varphi) \prod_{\tau=0}^{t} R_\varphi(s_\tau, a_\tau, r_\tau) T_\varphi(s_\tau, a_\tau, s_{\tau+1}) \quad (1) \\
&\propto P(\varphi|h_t) R_\varphi(s_t, a_t, r_t) T_\varphi(s_t, a_t, s_{t+1}). \quad (2)
\end{aligned}
$$

However, in general, the exact calculation of this belief update is intractable, so the existing methods approximate beliefs and avoid the calculation (Humplik *et al.*, 2019; Zintgraf *et al.*, 2020; Igl *et al.*, 2018; Kapturowski *et al.*, 2019). In the next section, we introduce the approximated belief update and other parts of our method.

3. Method

In this section we introduce our method, *embedding learning and evaluation of uncertainty* (ELUE), which learns how to infer tasks and how to use beliefs based on embeddings of task features. In addition, for alleviating the gap between meta-training and testing, which may prevent improvements being made if only belief updating is done, it also learns the adaptation of learned policy and Q-function through the updating of their parameters by using meta-test data. We show a sketch of the architecture of our networks in Figure 1.

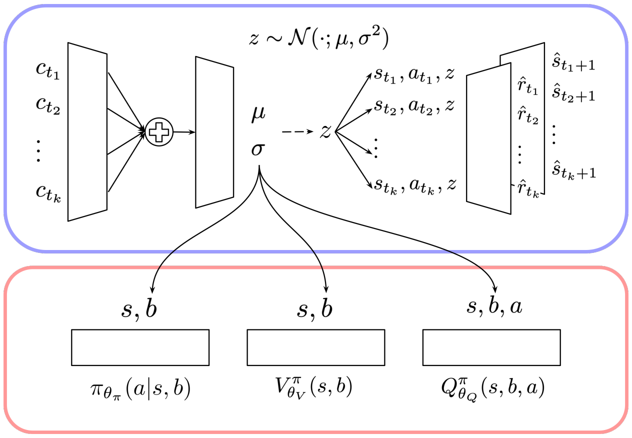


Figure 1: A sketch of the architecture of networks in ELUE. The networks in the blue box are for the embedding, consisting of an encoder (left) and decoders (right), which are introduced in Section 3.1. The input of the encoder is a set of $c_t = (s_t, a_t, r_t, s_{t+1})$ and it outputs the parameters of a Gaussian distribution, as introduced in Equation (6). A part of the inputs of the decoders, z, is sampled from the Gaussian distribution, and the decoders predict rewards and next states. In the red box, there are networks for the policy, V-function, and Q-function, which are trained in a similar way to soft actor-critic (Haarnoja *et al.*, 2018). The details are described in Section 3.2. These networks are conditioned on a belief and use the outputs of the encoder as the belief. Only the networks in the red box are adapted in meta-testing, while all of the networks are pretrained in meta-training, as discussed in Section 3.3.

3.1. Learning Embedding

In meta-training, ELUE learns natural embeddings for task features. In this section, we introduce its theoretical background.

We formulate the embedding learning problem as follows. There is a latent task variable z, whose density is $p(z)$, and let us assume that the reward r_t and next state s_{t+1} are sampled from a parameterized model $p_\phi(r_t, s_{t+1}|s_t, a_t, z)$, which is shared across tasks. If h_t is observed frequently, a reasonable model is expected to give h_t a high density. Thus, in proportion to the frequency of h_t, maximizing the density, $\log \int p(s_0) \prod_{\tau=0}^{t-1} p_\phi(r_\tau, s_{\tau+1}|s_\tau, a_\tau, z)\pi(a_\tau|h_\tau)p(z)dz$ leads to a reasonable model. However, the objective depends on the initial distribution and the current policy. In terms of belief estimation, they do not contribute as shown in proportional expression (1). Thus, instead of the density, we consider the maximization of the ELBO of the following value.

$$\log \int \prod_{\tau=0}^{t-1} p_\phi(r_\tau, s_{\tau+1}|s_\tau, a_\tau, z)p(z)dz \tag{3}$$

We introduce a parameterized variational distribution q_ϕ, and the ELBO is:

$$\log \int \prod_{\tau=0}^{t-1} p_\phi(r_\tau, s_{\tau+1}|s_\tau, a_\tau, z)p(z)dz \tag{4}$$

$$\geq \mathbb{E}_{q_\phi(z|h_t)}[\sum_\tau \log p_\phi(r_\tau, s_{\tau+1}|s_\tau, a_\tau, z)]$$
$$- D_{KL}(q_\phi(z|h_t)||p(z)). \tag{5}$$

We maximize this ELBO in a similar way to a conditional variational autoencoder (Sohn *et al.*, 2015), i.e., optimizing the parameters of encoder q and decoder p. The sum of log-likelihood in the ELBO is permutation-invariant in terms of time t of tuple $c_t := (s_t, a_t, r_t, s_{t+1})$. We introduce the following structure so that $q_\phi(z|h_t)$ is also permutation-invariant.

As shown in Zaheer *et al.* (2017), a function $q(X)$ is invariant to the permutation of instances in X, iff it can be decomposed into the form $g(\sum_{x\in X} f(x))$. We follow this fact and, instead of history conditional posterior $q_\phi(z|h_t)$, we use a posterior conditioned on a set of tuples,

$$q_\phi(z|c_{0:t-1}) := \mathcal{N}\left(z; g_\phi\left(\sum_{\tau=0}^{t-1} f_\phi(c_\tau)\right)\right), \tag{6}$$

where $\mathcal{N}(\cdot)$ is a Gaussian distribution, and $g_\phi(\sum_{\tau=0}^{t-1} f_\phi(c_\tau))$ outputs the parameters of the distribution. Note that $q_\phi(z|c_{0:t-1})$ can be used as an approximated belief over z, i.e., $b_t(z)$ and that it can be updated with low computational cost.

Let us denote a replay buffer of tuples of task i as D_i and a set of sampled tuples $c^i_{t_1}, c^i_{t_2}, \ldots c^i_{t_k}$ from D_i as $c^i_{t_{1:k}}$. We define the loss of embedding, $\mathcal{L}_{embed}(\phi)$, as

$$\mathbb{E}_{i,c^i_{t_{1:k}}}[-\mathbb{E}_{q_\phi(z|c^i_{t_{1:k}})}[\sum_{c_\tau \in c^i_{t_{1:k}}} \log p_\phi(r_\tau, s_{\tau+1}|s_\tau, a_\tau, z)]$$
$$+ D_{KL}(q_\phi(z|c^i_{t_{1:k}})||p(z))]. \tag{7}$$

Note that this loss function does not depend on the policy. Thus, it can reuse data in the replay buffer, which are collected by past policies, and the amount of data in the replay buffer is generally large. Moreover, because of the random sampling of tuples, this training depends less on actual trajectories than naive trajectory-based training and allows for more diversity in data sets. Those features can contribute to stability of the training objective.

In our implementation, we use two decoders, whose outputs are the probability of reward, $p_\phi(r_t|s_t, a_t, z)$, and that of next state, $p_\phi(s_{t+1}|s_t, a_t, z)$.

3.2. Learning Belief-Conditional Policy and Q-Function

ELUE learns a belief-conditional policy and Q-function in meta-training. To clarify the background of ELUE, we

Algorithm 1 Meta-training

```
A set of meta-training tasks, 𝒯 is given
while not done do
    Sample tasks from 𝒯
    Initialize beliefs
    for i ∈ the sampled tasks do
        for step in data collection steps do
            Gather data from task i by policy π(·|s^i, b^i)
            Update belief b^i and replay buffer D^i
        end for
    end for
    for step in training steps do
        Sample tasks from 𝒯
        Calculate L_embed for the sampled tasks, shown as formula (7)
        Update parameters to minimize L_embed
        Calculate L_actor and L_critic for the sampled tasks, shown as formulae (18), (19), and (20)
        Update parameters to minimize L_actor and L_critic
    end for
end while
```

Algorithm 2 Meta-testing

```
A meta-test task is given
for j ∈ the number of iteration steps do
    for step in data collection steps do
        Gather data from meta-test task by policy π(a|s, b)
        Update belief b and replay buffer D
    end for
    if j ≥ the number of iterations to start updating the parameters then
        for step in training steps do
            Calculate modified L_actor and L_critic for the task, shown as formulae (22), (19), and (20)
            Update parameter to minimize modified L_actor and L_critic
        end for
    end if
end for
```

introduce the control as a probabilistic inference framework (Levine, 2018). Following the settings in the reference, we assume a RL problem with finite horizon H. We denote the event that the optimal action is chosen at time t as $\mathcal{O}_t$, $\mathcal{O}_\tau$ for all τ in $t \leq \tau \leq H$ as $\mathcal{O}_{t:H}$ and assume $p(\mathcal{O}_t|r_t) := \exp(r_t)$. Let us assume that a probabilistic model such that $p(s_H^+, \mathcal{O}_{t:H-1}|s_t^+)$ can be represented as

$$p(a_t|s_t^+)p(r_t, s_{t+1}|s_t^+, a_t)p(\mathcal{O}_t|r_t)p(s_H^+, \mathcal{O}_{t+1:H-1}|s_{t+1}^+), \tag{8}$$

where $p(r_t, s_{t+1}|s_t^+, a_t) = \int p(r_t, s_{t+1}|s_t, a_t, z)b_t(z)dz$ and $p(a_t|s_t^+)$ is the action prior, which is assumed to be a uniform distribution over the action space. We introduce a variational distribution of future outputs,

$$q(s_H^+|s_t^+) = \pi(a_t|s_t^+)p(r_t, s_{t+1}|s_t^+, a_t)q(s_H^+|s_{t+1}^+). \tag{9}$$

At time t, ELUE maximizes the ELBO of $\log p(\mathcal{O}_{t:H}|s_t^+)$. Its ELBO is

$$\log p(\mathcal{O}_{t:H}|s_t^+) \tag{10}$$

$$\geq \mathbb{E}_{q(s_H^+|s_t^+)}\left[\log \frac{p(s_H^+, \mathcal{O}_{t:H}|s_t^+)}{q(s_H^+|s_t^+)}\right] \tag{11}$$

$$\geq \mathbb{E}_{q(s_H^+|s_t^+)}\left[\sum_{\tau=t}^{H} \log p(\mathcal{O}_\tau|r_\tau) + \log \frac{p(a_\tau|s_\tau^+)}{\pi(a_\tau|s_\tau^+)}\right]. \tag{12}$$

$p(a_t|s_t^+)$ is assumed to be a uniform distribution over the action space and is thus a constant. Therefore, let us remove the constant, then, the ELBO is

$$\mathbb{E}_{q(s_H^+|s_t^+)}\left[\sum_{\tau=t}^{H} r_\tau - \log \pi(a_\tau|s_\tau^+)\right]. \tag{13}$$

This value is the expected total return from s_t^+ with an entropy bonus as in soft actor-critic (SAC) (Haarnoja *et al.*, 2018), which is one of the most sample efficient off-policy RL method.

By following the control as inference scheme, we derive an objective like SAC. We modify the problem to that with an infinite horizon, $\gamma \leq 1$, and a coefficient of the entropy bonus, α, following SAC. As a result, the following Bellman equation is derived:

$$Q^\pi(s_t^+, a_t) := \mathbb{E}_{p(r_t, s_{t+1}|s_t^+, a_t)}[r_t + \gamma V^\pi(s_{t+1}^+)], \tag{14}$$

where

$$V^\pi(s_t^+) = \mathbb{E}_{q(s_\infty^+|s_t^+)}[\sum_{\tau=t}^{\infty} \gamma^{\tau-t}(r_\tau - \alpha \log \pi(a_\tau|s_\tau^+))] \tag{15}$$

$$= \mathbb{E}_{\pi(a_t|s_t^+)}[Q^\pi(s_t^+, a_t) - \alpha \log \pi(a_t|s_t^+)]. \tag{16}$$

We follow the same way as SAC and update the policy, Q-function, and V-function. Let us denote a belief conditioned on the tuple set $c^i_{t_{1:k-1}}$ as b^i and the belief from b^i updated by an additional tuple, $c^i_{t_k}$, as b'^i. For simplicity, we abbreviate the subscript t_k in $c^i_{t_k}$ and denote it as (s^i, a^i, r^i, s'^i). ELUE minimizes the following losses:

$$\mathcal{L}_{actor}(\theta_\pi) :=$$
$$\mathbb{E}_{i,c^i_{t_{1:k}}}\left[D_{KL}\left(\pi_{\theta_\pi}(\cdot|s^i,b^i)\;||\;\frac{\exp(Q_{\theta_Q}(s^i,b^i,\cdot))}{Z(s^i,b^i)}\right)\right] \tag{17}$$
$$= \mathbb{E}_{i,c^i_{t_{1:k}}}\left[\mathbb{E}_{\pi(a|s^i,b^i)}[\alpha\log\pi_{\theta_\pi}(a|s^i,b^i) - Q_{\theta_Q}(s^i,b^i,a)]\right], \tag{18}$$

$$\mathcal{L}^Q_{critic}(\theta_Q) := \mathbb{E}_{i,c^i_{t_{1:k}}}[(Q_{\theta_Q}(s^i,b^i,a^i) - \hat{Q}(s^i,b^i,a^i))^2], \tag{19}$$

$$\mathcal{L}^V_{critic}(\theta_V) := \mathbb{E}_{i,c^i_{t_{1:k}}}\left[(V_{\theta_V}(s^i) - \hat{V}(s^i))^2\right], \tag{20}$$

where
$$\hat{Q}(s^i,b^i,a^i) = r^i + \gamma V_{\bar{\theta}_V}\left(s'^i,b'^i\right),$$
$$\hat{V}(s^i) = \mathbb{E}_{\pi(a|s^i,b^i)}[Q_{\theta_Q}(s^i,b^i,a) - \alpha\log\pi_{\theta_\pi}(a|s^i,b^i)],$$
and $\bar{\theta}_V$ is a parameter vector that is updated by, $\bar{\theta}_V \leftarrow (1-\lambda)\bar{\theta}_V + \lambda\theta_V$. We show the procedures of our method in meta-training in Algorithm 1.

We introduce details on the implementation of our method. First, to reduce the computational cost, we avoid naively allocating one random sampled tuple set $c^i_{t_{1:k-1}}$ and belief to one additional tuple $c^i_{t_k}$. This is because the amount of data to be sampled is large and time consuming. Therefore, additional tuples share the same tuple set. Second, to train in a variety of situations in terms of the amount of data necessary to infer a task, we randomly sample k, the number of tuples in $c^i_{t_{1:k-1}}$.

3.3. Adaptation in Meta-Test

In meta-testing, ELUE collects data based on the policy conditioned on a belief. The belief is updated at every time step. After updating the belief enough times, our method updates the parameters of neural networks. In meta-testing, there are differences in the parameter update in meta-training:

1. The parameters for embedding, ϕ, are fixed in meta-testing to avoid catastrophic forgetting (French, 1999) about what it was learned in meta-training. Naive updating the parameters about embedding in meta-testing leads to catastrophic forgetting because the number of tasks in meta-testing is one, which means that the decoder can reconstruct the reward and next state without the latent task variable information, if the decoder is sufficiently trained in meta-testing. If the output of the decoder is independent from the latent task variable, only the second term of (7) is relevant to the learning of the encoder, which means that the encoder loss is minimized when its output is the same as that of the prior, $p(z)$. On the basis of these considerations, we fix ϕ.

2. To avoid catastrophic forgetting of the learned policy, we modify $\mathcal{L}_{actor}$ by adding a cross-entropy loss between the policy learned in meta-testing and that in meta-training. More concretely, the total actor loss is

$$\mathcal{L}_{actor}(\theta_\pi) - \mathbb{E}_{i,c^i_{t_{1:k}}}\left[\mathbb{E}_{\pi(a|s^i,b^i)}[\alpha\log\pi_{init}(a|s^i,b^i)]\right] \tag{21}$$
$$= \mathbb{E}_{i,c^i_{t_{1:k}},a}\left[\alpha\log\frac{\pi_{\theta_\pi}(a|s^i,b^i)}{\pi_{init}(a|s^i,b^i)} - Q_{\theta_Q}(s^i,a,b^i)\right], \tag{22}$$

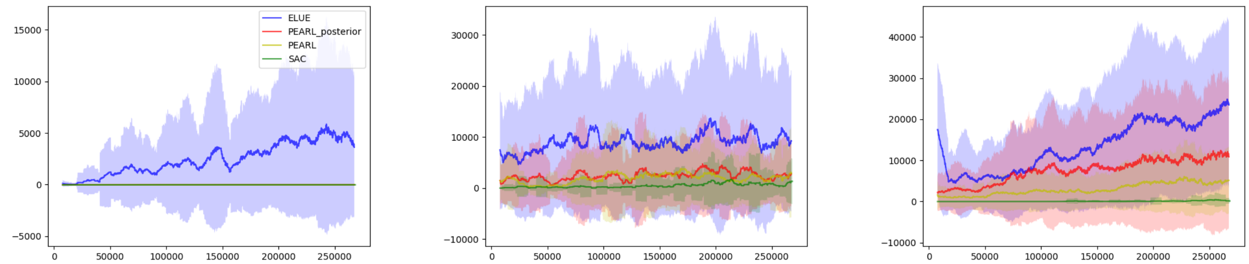
where π_{init} is the policy before its parameters are updated in meta-testing. This objective is expected to contribute to not only avoiding catastrophic forgetting but also improving performance. The entropy of a policy means the KL divergence between the policy and uniform policy if the constant part is ignored. Thus, the performance is expected to be better when using a well-trained policy instead of a uniform one to calculate the KL divergence. This modification improved the performance, as shown in Figure 4.

We introduce pseudo code in Algorithm 2.

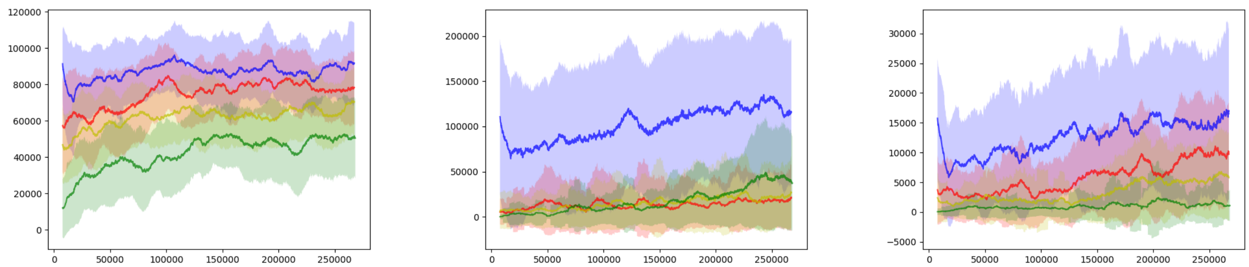
4. Related Work

In this section, we review existing methods related to our method and discuss the differences between them.

Our method is inspired by PEARL, but there are essential differences. First, PEARL has no decoder, and the encoder is trained to minimize the critic loss. It is a simple approach, but its embedding can change depending on the current policy. It has been shown that the performance of PEARL degrades when used with off-policy (i.e., not recent) data. Therefore, PEARL uses an additional buffer for recent data to avoid the degradation; in contrast, our method can train the embedding using old data and does not need an additional buffer. Second, PEARL uses an encoder that can be represented as $q_\phi(z|h_t) \propto \prod_{\tau=0}^{t-1}\mathcal{N}(z;\mu_\tau,\sigma^2_\tau) \propto \mathcal{N}(z;\frac{\sum_{\tau=0}^{t-1}\frac{\mu_\tau}{\sigma^2_\tau}}{\sum_{\tau=0}^{t-1}\frac{1}{\sigma^2_\tau}},\frac{1}{\sum_{\tau=0}^{t-1}\frac{1}{\sigma^2_\tau}})$, where μ_τ and σ^2_τ are the outputs of the neural network, $f_\phi(\cdot)$, whose input is c_τ, and they are the mean and variance of a Gaussian distribution. As discussed in Section 3.1, this is not general form for encoder representation in terms of permutation invariance among c_τ, while our encoder is represented in a general form. Third, PEARL's policy and Q-function, $\pi(s,z)$ and $Q(s,a,z)$, where z is sampled from $q(z|h_t)$, are z conditional, and z itself has no uncertainty information. In comparison, ours are belief-conditional, which has uncertainty information. Fourth, PEARL only considers inference in meta-testing, while our method con-



(a) basket-ball (b) dial-turn (c) pick-place



(d) reach (e) sweep-into (f) window-open

Figure 2: The comparison of learning curves. The vertical axis is moving average $\pm$ standard deviation of average episode rewards in meta-testing and the horizontal axis is the number of time steps.

siders the updating of the parameters of neural networks.

VariBAD (Zintgraf *et al.*, 2020) is more related to our method. It considers embedding just like ours and beliefs over the embedding space. However, this is an on-policy algorithm, and its sample efficiency is not as high as PEARL. Zintgraf *et al.* (2020) compared the performances of PEARL and variBAD after several rollouts in meta-testing, where each algorithm was trained with the same number of frames in meta-training, and they showed that PEARL outperformed variBAD. In addition, its encoder is based on recurrent neural networks whose input is simply a history. Moreover, it only infers in meta-testing.

Humplik *et al.* (2019) proposed several methods for training a belief network over tasks. However, unlike ours, their beliefs regard direct prediction, e.g., predicting the task ID, and the parameters of the task. In addition, their encoder is based on RNNs and fed histories naively.

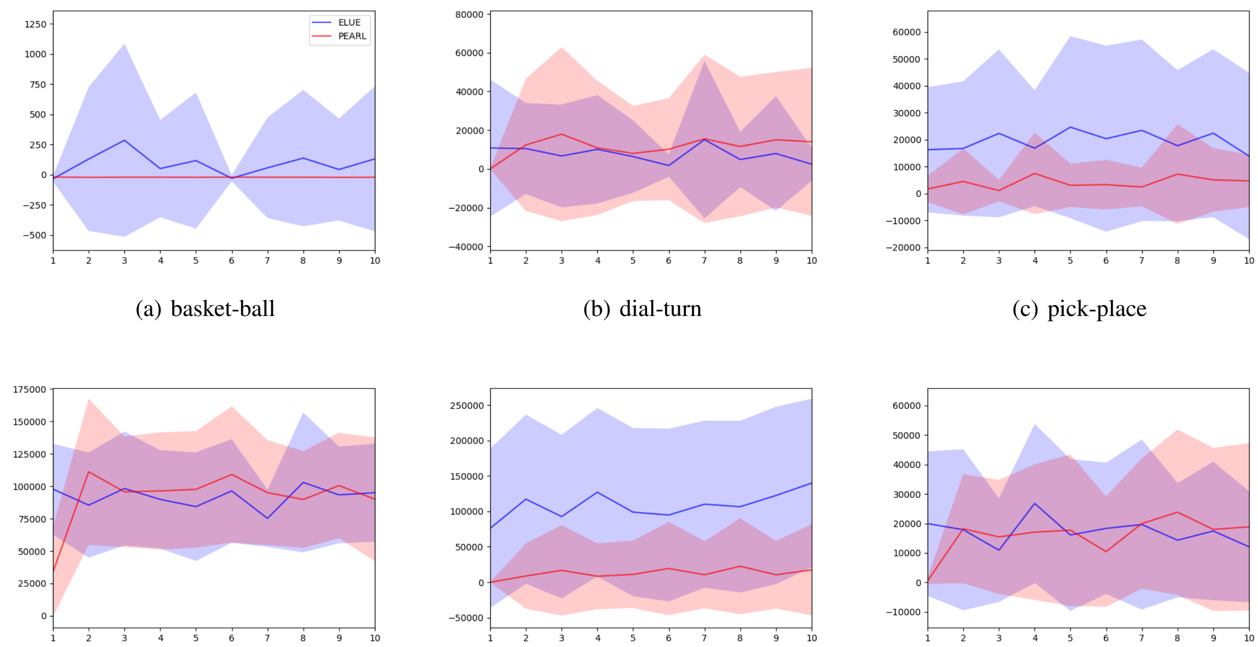
Vuorio *et al.* (2019) proposed MMAML, which is an extension of PEARL. It is a combination of PEARL-like task inference and MAML-like parameter updating. However, this is an on-policy algorithm, and it does not consider the uncertainty of tasks. Although combining our method and MAML would be an interesting direction for future work, it would not be straight-forward to combine a fully off-policy algorithm and the MAML objective for better adaptation in terms of parameter updating.

As for off-policy approaches, guided meta policy search (Mendonca *et al.*, 2019) is introduced as an off-policy meta learning method. However, it is on-policy in meta-testing. In addition, it is not based on amortized inference.

5. Experiments

To examine the effectiveness of our method, we compared the performance of PEARL and our method. The environment of the experiments was Meta-World with MuJoCo 2.0. Meta-World is a collection of robot arm tasks, and there are 50 types of tasks and several benchmarks. We followed the ML1 benchmark scheme, where a difference in tasks means difference in goals. We chose six types of tasks, basket-ball, dial-turn, pick-place, reach, sweep-into, window-open, that were chosen from the types of meta-test tasks of the ML10 benchmark[1]. Basket-ball is a task where the agent dunks a basketball into a basket, and each task has different basketball and basket positions. Dial-turn is a task where the agent rotates a dial 180 degrees, and each task has different dial positions. Pick-place is a task where the

[1]The names of the types of tasks in the Meta-World paper are different from those of the Meta-World program. We refer to the names of the program.



(a) basket-ball (b) dial-turn (c) pick-place

(d) reach (e) sweep-into (f) window-open

Figure 3: Comparison of learning curve with inference of each algorithm. Vertical axis is moving average $\pm$ standard deviation of average episode rewards in meta-testing. Horizontal axis is number of episodes.

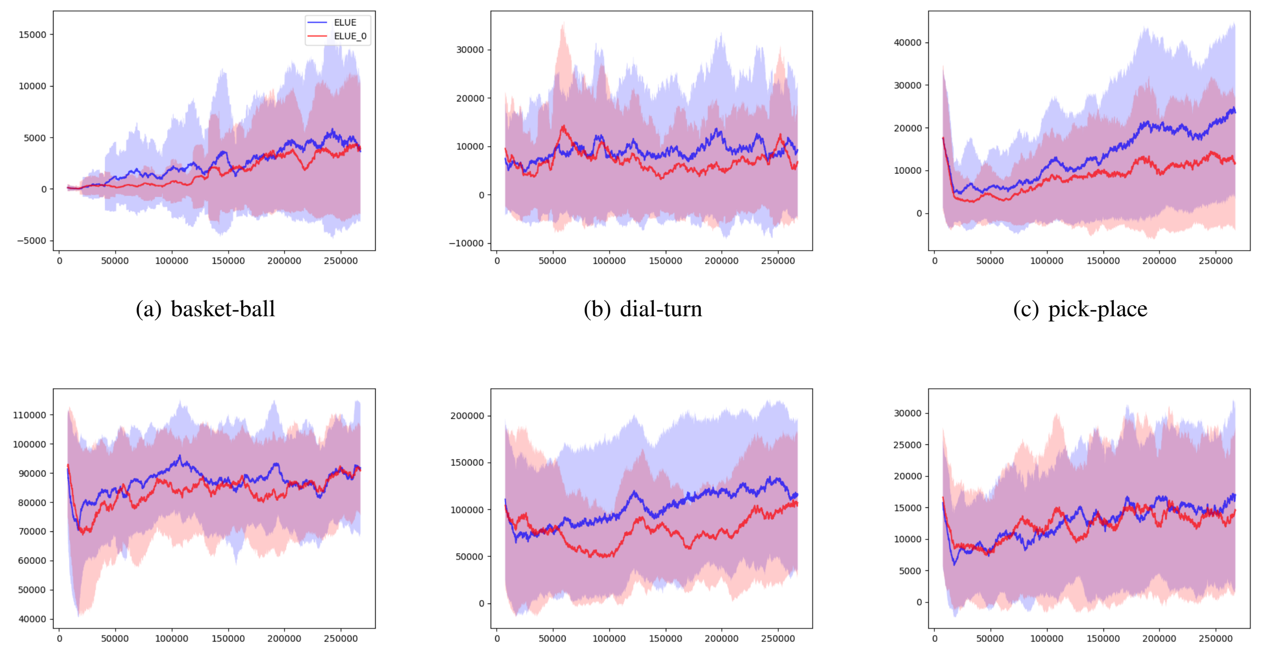
agent picks and places a puck at a goal, and each task has different puck and goal positions. Rearch is a task where the agent reaches a goal position, and each task has different goal positions. Sweep-into is a task where the agent sweeps a puck into a hole, and each task has different puck positions. Window-open is a task where the agent pushes and opens a window, and each task has different window positions.

For each type of task in meta-training, ten tasks were sampled from the meta-training task distribution defined by the ML1 benchmark. In meta-testing, one task was sampled from the task distribution of meta-testing, which was different from that of meta-training. We executed meta-training three times, where each algorithm was trained with the same number of time steps. For each learned networks, meta-tests were executed three times. Meta-training was executed for 300 iterations. We used trained networks at the 290th iteration. The total number of time steps in meta-training in the environment of each method was the same (582,000 steps).

In the first experiment, we compared the learning curves of the algorithms in meta-testing. In the experiment, each algorithm updated the parameters of its learned networks. The original PEARL algorithm does not consider parameter updates in meta-testing, so we revised it to alleviate the differences between tasks in meta-training and meta-testing. We compared ELUE, a naive extension of PEARL ("PEARL"), and PEARL with modifications ("PEARL-posterior"). Also, to clarify the amount of improvement with our method, we executed SAC, which learns from scratch in meta-testing. "PEARL" is a naive application of the original PEARL's meta-training procedures for parameter updates in meta-testing. "PEARL-posterior" is a modification of PEARL that samples the latent task variable with only the posterior distribution in meta-testing (except for the first episode of every iteration), for better sample efficiency. The original PEARL algorithm samples the latent task variable with not only the posterior but also the prior distribution. The results are shown in Figure 2. The proposed method achieved better results, especially for sweep-into and basket-ball. In basket-ball, performance of ELUE improved gradually, while the other methods did not.

In the second experiment, to compare the performance when the amount of meta-testing data was very small, we evaluated the methods without parameter updates in meta-testing. Figure 3 shows that our method outperformed PEARL in basket-ball, sweep-into, and pick-place. In addition, our method was better in terms of the cumulative reward in the first episode in the other tasks.

In the third experiment, as an ablation study, we compared ELUE and ELUE without cross-entropy loss ("ELUE_0"), which we introduced in Section 3.3. Figure 4 shows the



(a) basket-ball (b) dial-turn (c) pick-place

(d) reach (e) sweep-into (f) window-open

Figure 4: Comparison with or without cross-entropy loss of ELUE. Vertical axis is moving average $\pm$ standard deviation of average episode rewards in meta-testing, and horizontal axis is number of time steps.

results. Without the cross entropy loss, the performance degraded slightly on some tasks.

6. Conclusion

We proposed a novel off-policy meta-learning method, ELUE. It learns the natural embeddings of task features, beliefs over the embedding space, belief conditional policies and Q-functions. We apply a general permutation-invariant form to the belief representations in our method. Because of this, ELUE can train independently from actual trajectories, which can lead to diversity in data set and stable training. The belief-conditional policy and Q-function are learned in a manner similar to soft actor-critic. Because of the beliefs, the performance can be improved by updating the beliefs, especially when the meta-test task is similar to the meta-training tasks. ELUE also updates the parameters of neural networks in meta-testing, which can alleviate the gap between tasks in meta-testing and those in meta-training. In experiments, we examined the sample efficiency of ELUE and PEARL with Meta-World benchmarks, and we showed that ELUE outperformed PEARL.

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Linfeng Liu and Liping Liu. Amortized variational inference with graph convolutional networks for gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2291–2300, 2019.

Russell Mendonca, Abhishek Gupta, Rosen Kralev, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Guided meta-policy search. In *Advances in Neural Information Processing Systems*, pages 9653–9664, 2019.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *international conference on machine learning*, pages 5331–5340, 2019.

Juergen Schmidhuber, Jieyu Zhao, and MA Wiering. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

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

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

Vivek Srikumar, Gourab Kundu, and Dan Roth. On amortizing inference cost for structured prediction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1114–1124. Association for Computational Linguistics, 2012.

Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. Learning stochastic inverses. In *Advances in neural information processing systems*, pages 3048–3056, 2013.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, pages 1–12, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020.