Proceedings of The 2022 IEEE International
Conference on Real-time Computing and Robotics
July 17-22, 2022, Guiyang, China

2022 IEEE International Conference on Real-time Computing and Robotics (RCAR) | 978-1-6654-6983-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/RCAR54675.2022.9872291

Celebrating Robustness in Efficient Off-Policy Meta-Reinforcement Learning

Ziyi Liu[1*], Zongyuan Li[1*], Qianqian Cao[1], Yuan Wan[2], Xian Guo[1]

***Abstract*— Deep reinforcement learning algorithms can enable agents to learn policies for complex tasks without expert knowledge. However, the learned policies are typically specialized to one specific task and can not generalize to new tasks. While meta-reinforcement learning (meta-RL) algorithms can enable agents to solve new tasks based on prior experience, most of them build on on-policy reinforcement learning algorithms which require large amounts of samples during meta-training and do not consider task-specific features across different tasks and thus make it very difficult to train an agent with high performance. To address these challenges, in this paper, we propose an off-policy meta-RL algorithm abbreviated as CRL (Celebrating Robustness Learning) that disentangles task-specific policy parameters by an adapter network to shared low-level parameters, learns a probabilistic latent space to extract universal information across different tasks and perform temporal-extended exploration. Our approach outperforms baseline methods both in sample efficiency and asymptotic performance on several meta-RL benchmarks.**

I. INTRODUCTION

Deep reinforcement learning (DRL) has made significant advances in various domains [1]–[4] and enabled agents to autonomously learn policies for complex tasks from scratch [1]–[3], [5]. However, learning these tasks generally necessitates a huge number of samples. Meta-reinforcement learning (meta-RL) algorithms can enable agents to leverage prior experience generated by previously seen related tasks to solve new tasks [6]–[8], but most of these methods rely on on-policy reinforcement learning algorithms which still require a large number of samples during meta-training [6], [7], [9], [10], prohibiting their abilities to solve real-world complex problems. Furthermore, current meta-RL methods [9], [11]–[13] typically learn universal feature representations from all meta-training tasks and do not consider task-specific features during training, which will further damage their sample efficiency.

Most meta-RL methods can be divided into two categories [14], i.e., optimization-based and black-box adaptation based. Optimization-based methods use a variant of model-agnostic meta-learning (MAML) and optimize their gradients to adapt to new tasks [6], [15]. Black-box adaptation methods typically utilize an encoder, such as recurrent models [7], [10], attention mechanisms [16] or variational inference [11], to encode experience for task adaptation. The latter generally struggle when training tasks and testing tasks are drawn from different distributions, i.e., distributional shift, and even if minor distribution mismatch turn out to damage their performance [17]. Optimization-based methods are robust to distributional shift since gradient descent leads to a well-defined and consistent learning process that has a guarantee of improvement regardless of the task [18]. However, these optimization-based methods [6], [15] adopt on-policy policy gradient algorithms for meta-RL training that require a large number of samples during the meta-training procedure [11]. Furthermore, they have poor exploration ability due to the lack of structured stochasticity in the exploration strategy [9].

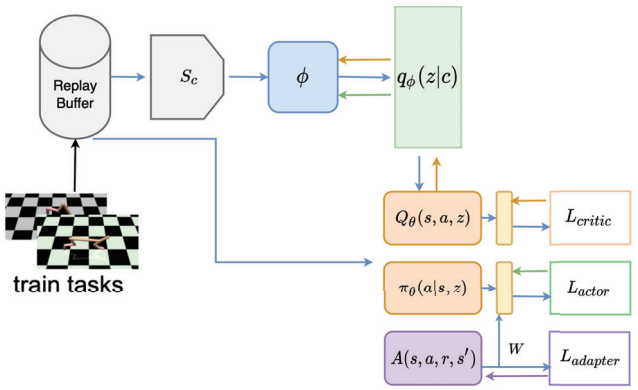

Fig. 1. **Meta-training stage of CRL.** The probabilistic encoder $q_\phi(z|c)$ encodes near on-policy context data c from experience buffer S_c to latent variables z which conditions the actor $\pi_\theta(a|s,z)$ and critic $Q_\theta(s,a,z)$, and is optimized with gradients from critic and KL divergence objective function. Actor and Critic are optimized with off-policy data from replay buffer, and their output layer parameters are learned by the prediction model $A(s,a,r,s')$ which are optimized using supervised learning. Blue line indicates the direction of data flow, and other lines with different color represent gradient flows.

We aim to address these challenges by combining the advantage of optimization based meta-learners such as MAML [6] which have better generalization ability and block-box adaptation based off-policy methods like PEARL [11] which are sample efficient and allow exploration to be time-correlated. Specifically, we assume that similar tasks share common features in low-level neural network layers but have task-specific features in high-level layers. During meta-training, we learn shared low-level policy network parameters between different tasks using Soft Actor Critic (SAC) [19] and predict task-specific parameters by a separate adapter network, which is trained in a supervised learning method using stochastic gradient descent. During meta-

*These authors contribute equally to this work. This work is supported by the National Natural Science Foundation of China under grant 61873132. Corresponding Author: Ziyi Liu

[1]These authors are with the Institute of Robotics and Automatic Information System and the Tianjin Key Laboratory of Intelligent Robotics, College of Artificial Intelligence, NanKai University, Tianjin 300350, China

[2]Wan Yuan with the Department of Mathematical, Wuhan University of Technology, WuHan 430070, China

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.

testing, we can update the agent's high level task-specific policy parameters by gradient-based adaptation.

To further achieve rapid adaptation and exploration across different tasks, we adopt a probabilistic encoder that extracts the minimum universal features from tasks' distribution into the latent variables which are sampled from a learned latent probabilistic space. Disentangling task-specific policy parameters by adapter network to shared policy parameters and learning a prediction model makes our approach generalize better during mete-testing. Furthermore, off-policy data can be utilized to optimize policy network to improve sample efficiency while on-policy data is used to train the probabilistic encoder to ensure consistency between meta-training and meta-testing [20].

The core contribution of our work is a robust off-policy meta-RL algorithm named CRL (Celebrating Robustness Learning). Empirically, our method is more robust than black-box adaptation methods, achieves excellent sample efficiency during meta-training, enables fast adaptation by learning a prediction model, and performs structured exploration by reasoning about uncertainty over latent probabilistic space.

II. Background

A. Meta-Reinforcement Learning

We consider the standard Markov Decision Process (MDP) setting defined by a tuple $\mathscr{T} =< S,A,p,r,\gamma >$, where S is the state space, A is the action space. At each time step, the agent receives its state s and selects an action a based on its stochastic policy $\pi_\theta(a|s)$. The environment transitions to next state s' according to the transition probability $p(s'|s,a)$ and the agent receives its reward $r(s,a,s')$. $\gamma \in (0,1)$ is the discount factor.

In meta-RL, there is a task distribution $p(\tau)$ over MDPs, from which tasks are sampled and each task τ_i is a different MDP. The reward function and transition probability vary across tasks. The meta-training process learns a policy that can adapt to a set of training task sampled from $p(\tau)$. At meta-testing, given a small amount of task data D^{τ_i}, the agent can fast adapt to the specific task.

B. Soft Actor Critic.

The Soft Actor Critic (SAC) [19] is an off-policy actor-critic algorithm with maximum entropy objective function. The objective encourages the agent to perform stochastic exploration by augmenting the reward with the entropy of the policy. We can optimize the policy parameters θ_π by minimizing the

$$J_\pi(\theta_\pi) = E_{\mathscr{D},\pi}[D_{KL}\left(\pi_{\theta_\pi}(\cdot|s_t)\middle|\middle|\frac{exp(\frac{1}{\alpha}Q_{\theta_Q}(s_t,\cdot))}{Z(s_t)}\right)], \quad (1)$$

where Q_{θ_Q} is the soft Q-function and $Z(s_t)$ is the normalizing constant.

The parameters of soft Q-function θ are optimized by minimizing the soft Bellman residual

$$J_Q(\theta_Q) = E_{\mathscr{D},\pi}\left[\frac{1}{2}(Q_{\theta_Q}(s_t,a_t) - y(s_t,a_t))^2\right], \quad (2)$$

where $y(s_t,a_t)$ denotes the temporal difference target

$$r(s_t,a_t) + \gamma\left[Q_{\bar{\theta}_Q}(s_{t+1},a_{t+1}) - \alpha\log\pi_{\theta_\pi}(a_{t+1} \mid s_{t+1})\right] \quad (3)$$

and α is the entropy temperature coefficient. [19] proposed to dynamically adjust the α by taking a gradient step with respect to the loss

$$J(\alpha) = \mathbb{E}_{\mathscr{D},\pi_{\theta_\pi}}[\log\alpha\cdot(-\log\pi_{\theta_\pi}(a_t \mid s_t) - \mathscr{H}_T)] \quad (4)$$

[21] takes the minimum over two Q-function approximators to compute the target in equation 3 and policy objective in equation 1.

C. Variational Autoencoders.

The variational autoencoders (VAEs) [22] are a special type of latent variable models which perform a probabilistic inference on regular autoencoders [23] and be tractably trained to represent the generative model. For the generative process, VAE methods sample latent variables z from a fixed prior distribution $p(z)$ which usually correspond to a standard Gaussian distribution and train a neural network with parameters θ to approximate the conditional distribution $p_\theta(x|z)$ to generate data x, i.e., $x \sim p_\theta(x|z)$. For probabilistic inference, VAE methods perform amortized inference by training a separate network $q_\phi(z|x)$ with parameter ϕ to represent the posterior z given an observation x.

VAE maximizes the evidence lower bound (ELBO) of the training data's log-likelihood. For a given data x^i, the ELBO(x^i) is

$$-D_{KL}(q_\phi(z|x^i)||p_\theta(z)) + E_{q_\phi(z|x_i)}[\log p_\theta(x^i|z)] \quad (5)$$

where D_{KL} is the KL divergence.

In practice, the encoder and decoder parameters, ϕ and θ respectively, are jointly optimized by maximizing

$$E_{q_\phi(z|x_i)}[\log p_\theta(x^i|z)] - \beta D_{KL}(q_\phi(z|x^i)||p_\theta(z)) \quad (6)$$

where β is hyperparameter that has been shown to encourage the generative model to learn representations that are more disentangled when $\beta > 1$ [24]. In CRL, we use an amortized variational inference approach to encode salient information about the task into latent variables Z and perform probabilistic inference for efficient exploration while the encoder is jointly optimized with critic via RL objective.

D. Exploration via Latent Space Sampling.

Modeling the probabilistic latent variables allows us to perform efficient exploration by posterior sampling at both meta-training and meta-testing time. Practically, Sampling from probabilistic latent space and augment state space not only allows us extract universal features across tasks but injects temporal-extended noise into state space. Latent variable models have been explored in [9], [25], [26]. But

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.

these works either not explicitly train for fast adaptation or not in the context of meta-learning.

III. Method

In this section, we describe our CRL algorithm, and its architecture is showed in Fig.1. Our algorithm can be divided into three parts: Adapter Network, Probabilistic Latent Encoder and Off-Policy Learning Algorithm. Firstly we present our assumption and motivation, and then we describe three parts respectively.

A. Assumption and Motivation

Inspired by the progress of Representation Learning [27]–[29] which indicates that neural network is a kind of feature extractor. Low-level layers extract universal features and high-level layers extract fine grained features. Furthermore, very recent studies [30], [31] have attributed the success of MAML to high-quality features before the task-specific updates from the meta-initialized parameters, which implies that a good feature representation can generalize across tasks. We build on this one step further and assume that similar tasks share common features in shallow part of neural network layers but have task-specific features in high-level layers, e.g., output layer. The universal features across the whole task distribution can be learned and stored in shallow parts of policy parameters and we can perform a task-specific update during meta-testing by utilizing an adapter network to predict the parameters of policy network's output layer based on data collected by the agent. So we propose to disentangle task-specific policy parameters by adapter network to shared low-level parameters which can improve agent's generalization ability and extract fine grained task-specific features.

B. Adapter Network

We use a deep neural network to approximate the actor, sharing all except the output layer parameters across all the meta-training tasks, and each meta-training task has a unique task-specific output layer during meta-training. The shared shallow parameters of the policy network capture the universal features across the whole meta-training tasks.

Mathematically, we parametrize the policy via a neural network as $\pi(\phi, w_i, b_i) = \sigma(\langle \phi, w_i \rangle + b_i)$, where w_i and b_i are unique weights and bias respectively for task τ_i. σ represents the linear activation for the output layer and ϕ represents shared parameters across tasks. So the whole policy parameters are denoted as $\theta = (\phi, w_i, b_i)$. The hierarchical architecture of the policy network is showed in Fig.2.

In meta-training step, adapter network is trained with agent's experiences collected by interacting with the environment of current task to predict the policy network's task-specific parameters. The loss function of adapter network is:

$$\begin{aligned} & f_{\text{Adapter}}(\cdot) : (s,a,r,s')_i \rightarrow (\hat{w}_i, \hat{b}_i) \\ & \mathscr{L} = \frac{1}{n} \sum_{i=1}^{n} \left(f_{\text{Adapter}} - (w_i, b_i) \right)^2 \end{aligned} \tag{7}$$

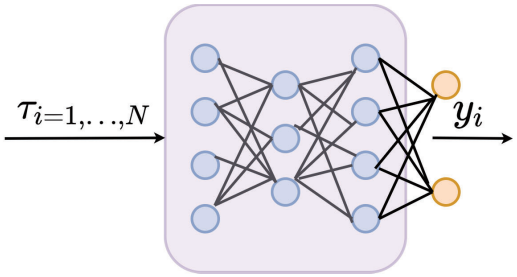

Fig. 2. **Structure of layered policy.** To introduce task information and improve robustness, we layered the policy net into shared part (blue cells) and unique part (yellow cells), make it possible to adapt to a specific meta-testing task.

where $(s,a,r,s')_i$ is the transition of task τ_i, $(\hat{w}_i, \hat{b}_i)$ are parameters predicted by adapter network, (w,b) are true parameters learned by the policy network, and n is the total number of meta-training tasks.

In meta-testing step, the adapter network predicts task-specific parameters of the policy network based on a small amount of data collected by the agent interacting with the new task, so the agent can adapt to the new task efficiently. We fix the shallow part parameters ϕ of the policy network during meta-testing, and only update the task-specific parameters for each testing task:

$$w_{\text{new}}, b_{\text{new}} = f_{\text{Adapter}}((s,a,r,s')_t)_{t=1}^{H} \tag{8}$$

where H is a hyper-parameter which indicates the number of time steps used to adapt a new task.

C. Learning Probabilistic Latent Variables from Task Experience

Given a set of meta-training tasks sampled from task distribution $p(\tau)$, we denote the transitions collected by the agent as context c, so $c_n^{\tau_i} = (s_n, a_n, s_{n+1}, r_n)^i$ is one transition in task τ_i and $c_{1:T}^{\tau_i}$ denotes the experiences collected so far. To extract universal knowledge across the task distribution and learn how the current task should be performed, we adopt an amortized variational inference approach [22] to learn a posterior distribution over latent variables Z. Conditioned on context c of current task, Z encodes salient information about the task and the posterior distribution contains universal features across the meta-training tasks.

Computing the posterior distribution over tasks' experiences exactly is intractable, so we adopt a variational inference approach to maximize the lower bound on the log-likelihood objective [22] similar as Equation 6. Specifically, we use beta-vae to train an inference network $q_\phi(z|c)$ to estimate the true posterior $p(z|c)$. The inference network can be optimized in a generative way by sampling z using reparameterization trick to minimize critic's TD erros. The resulting variational lower bound is:

$$\mathbb{E}_{\mathscr{T}} \left[\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathscr{T}})} \left[R(\tau, \mathbf{z}) - \beta D_{\text{KL}} \left(q_\phi \left(\mathbf{z} \mid \mathbf{c}^{\mathscr{T}} \right) \| p(\mathbf{z}) \right) \right] \right] \tag{9}$$

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.

where $p(z)$ is the standard Gaussiaisn prior over Z, and $R(\tau, \mathbf{z})$ is the reward function of current task. The KL divergence term constrains the distance between posterior distribution and the prior distribution which can be viewed as an information bottleneck [32]. This bottleneck filters task-agnostic noisy information. The parameters of q_ϕ are optimized during meta-training jointly with the critic while fixed at meta-testing step, and the probabilistic latent variables are inferred conditioned on context collected from the new task.

D. Implementing CRL

The variational posterior is factorized as $q_\phi(z|c_{1:T}) \propto \prod_{i=1}^{T} \Psi_\phi(z|c_i)$, and the Ψ_ϕ is Gaussian distribution parameterized by ϕ. Then we can use the soft actor-critic (SAC) [19] which is a sample efficient off-policy algorithm to optimize the inner loop of meta-RL, i.e., task-specific update. The actor $\pi_\theta(a_t|s_t, z_t)$ and the critic $Q_\theta(s_t, a_t, z_t)$ are conditioned on the probabilistic latent variables z_t, and trained by the SAC method. The parameters of the inference network $q_\phi(z|c)$ are jointly optimzied with the critic $Q_\psi(s_t, a_t, z_t)$, and we use the reparameterization trick [22] to compute gradients for parameters of $q_\phi(z|c)$ through sampled z's. The critic loss can then be written as,

$$\mathscr{L}_{critic} = \mathop{\mathbb{E}}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathscr{B}, \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}} \left[Q_{\theta_Q}(\mathbf{s},\mathbf{a},\mathbf{z}) - \left(r(\mathbf{s},\mathbf{a}) + \bar{V}\left(\mathbf{s}', \bar{\mathbf{z}}\right)\right)\right]^2 \tag{10}$$

where $\bar{z}$ indicates that gradient computing is disabled. The parameters of the actor network are optimzied by minimizing

$$\mathscr{L}_{\text{actor}} = \mathop{\mathbb{E}_{\mathbf{KL}}}_{\substack{\mathbf{z}\sim\mathscr{B}, \mathbf{a}\sim\pi_{\theta_\pi} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}} \left[D_{\text{KL}} \left(\pi_{\theta_\pi}(\mathbf{a} \mid \mathbf{s}, \bar{\mathbf{z}}) \| \frac{\exp\left(Q_{\theta_Q}(\mathbf{s},\mathbf{a},\bar{\mathbf{z}})\right)}{Z_{\theta_Q}(\mathbf{s})} \right) \right] \tag{11}$$

IV. Experiments

To evaluate the performance of our method, we compare CRL with other baselines, including PEARL [11], MQL [13] and SAC [19], in four meta-RL environments [11]. First, we describe the environmental setup. Then we present the evaluation results compared with other baselines.

A. Baselines

SAC [19] is an off-policy reinforcement learning algorithm.

PEARL [11] is an off-policy meta-RL method proposed by Kate Rakelly et.al. which adopts variational inference over latent context variables to adapt to new tasks.

MQL [13] is an off-policy meta-RL method building upon Q-learning. MQL is a strong baseline since it can access to past trajectory and optimizes a multi-task objective across all meta-training tasks which is intractable in real world meta-learning settings.

Algorithm 1: CRL: Meta-Training

Require : Batch of training tasks $\{\tau_i\}_{i=1...N}$ from $p(\tau)$, learning rates $\alpha_1, \alpha_2, \alpha_3, \alpha_4$

For each training task, initialize replay buffers $\mathscr{B}^i$
while not done **do**
  for each τ_i **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for k = 1,..., K **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      roll-out data from $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ and add them to $\mathscr{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t, r_t)\}_{t:1...T} \sim \mathscr{B}^i$
    end
  end
  for step in training steps **do**
    for each τ_i **do**
      Sample context $\mathbf{c}^i \sim \mathscr{S}_c(\mathscr{B}^i)$ and RL batch $(\mathbf{s},\mathbf{a},\mathbf{r},\mathbf{s}')^i \sim \mathscr{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathscr{L}^i_{actor} = \mathscr{L}_{actor}$
      $\mathscr{L}^i_{critic} = \mathscr{L}_{critic}$
      $\mathscr{L}^i_{KL} = \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{c}^i)||p(\mathbf{z}))$
      weights = Flatten (w_{τ_i})
      target=Concat (weights, b_{τ_i})
      $\mathscr{L}^i_{\text{adapter}} = \mathscr{L}^i_{\text{adapter}}$ $((\mathbf{s},\mathbf{a},\mathbf{r},\mathbf{s}')$, target)
    end
    $\phi \leftarrow \phi - \alpha_1 \bigtriangledown_\phi \Sigma_i (\mathscr{L}^i_{critic} + \mathscr{L}^i_{KL})$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \bigtriangledown_\theta \Sigma_i \mathscr{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \bigtriangledown_\theta \Sigma_i \mathscr{L}^i_{critic}$
    $\theta_{\text{adapter}} \leftarrow \theta_{\text{adapter}} - \alpha_4 \nabla_\theta \sum_i \mathscr{L}^i_{\text{adapter}}$
  end
end

Algorithm 2: CRL: Meta-Testing

Require : Batch of testing tasks τ from $p(\tau)$
Initialize context $\mathbf{c}^\tau = \{\}$
for m = 1,..., M **do**
  sample $z \sim q_\phi(z|c_\tau)$
  roll-out policy $\pi_\theta(a|s,z)$ to collect data $D^\tau_k = (s_k, a_k, r_k, s'_k)^\tau$
  Accumulate context $c^\tau = c^\tau \cup D^\tau_k$
  $\hat{w}^\tau, \hat{b}^\tau = Adapter(s_k, a_k, r_k, s'_k)$
  Update the parameters of the actor's output layer by $\hat{w}^\tau, \hat{b}^\tau$
end

B. Experimental Setup

The environment details and experiment setting are described as follows:

Half-Cheetah-Fwd-Back. Control a simulated cheetah to move forward and backward (100 train tasks, 30 test tasks).

Ant-Fwd-Back. Control a simulated ant to go forward or back according to the requirements of the sampled task (2 train tasks, 2 test tasks).

Humanoid-Direc-2D. Control a simulated person to walk towards a given direction on 2D grid (10 train tasks, 30 test tasks).

Walker-2D-Params. Control a simulated simplified walking person initialized with some randomized system dynam-

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.

ics parameters, and make him move forward on 2D grid (40 train tasks, 10 test tasks).

In each experiment, all the algorithms adopt the same hyper-parameters, except for the parameters of unique components, e.g., the adapter network. Furthermore, we carry out all experiments on two NVIDIA RTX 3090 GPUs.

C. Results

We evaluate CRL and other baseline methods, including MQL, PEARL and SAC in four meta-RL environments. The experiment results are shown from Fig. 3 to Fig. 6. We do not evaluate other on-policy meta-RL methods since PEARL has outperformed them significantly.

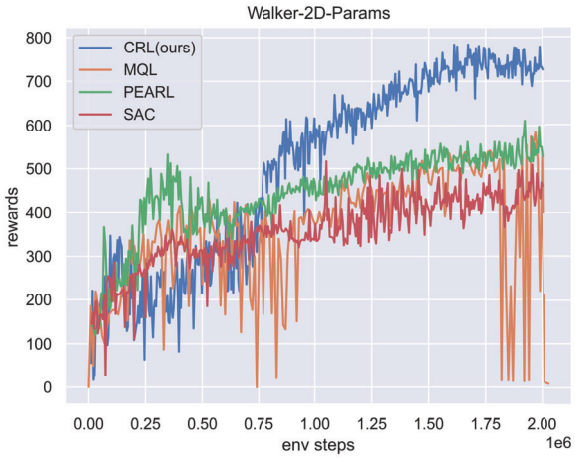

Fig. 3. Average Rewards of Walker-2D-Params in meta-testing.

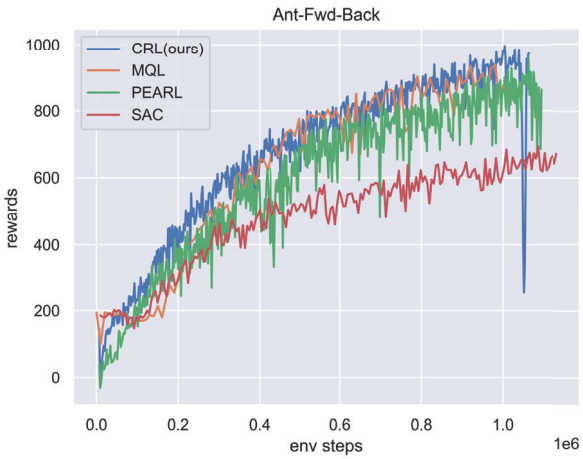

Fig. 4. Average Rewards of Ant-Fwd-Back in meta-testing.

The experiment results show that CRL has better sample efficiency and outperforms other baselines in three of four meta-RL environments, especially in Walker-2D environment.

It is worth noting that the probabilistic latent encoder and adapter network greatly improve the performance of CRL.

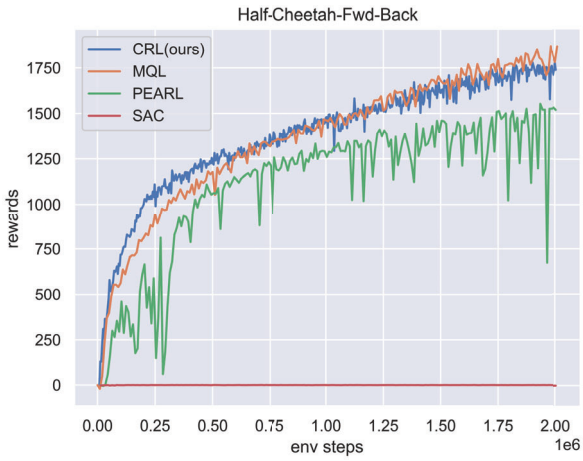

Fig. 5. Average Rewards of Half-Cheetah-Fwd-Back in meta-testing.

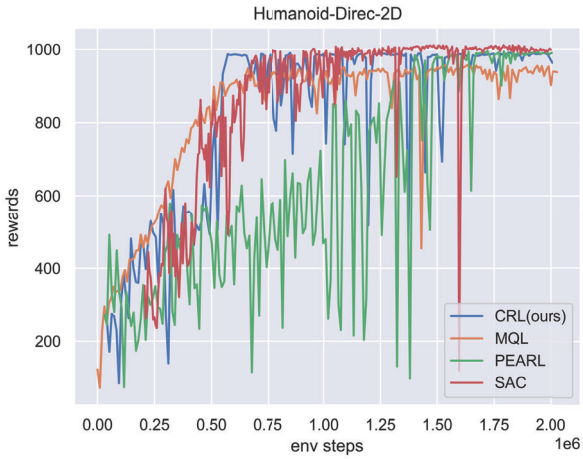

Fig. 6. Average Rewards of Humanoid-Direc-2D in meta-testing.

Probabilistic encoder improves the ability of CRL to extract universal task features over the whole meta-training tasks, and task-specific parameters of the actor can capture task-specific information fine grainedly, thus can achieve better asymptotic performance.

The maximum and average return of all methods in the last 50k meta-testing time steps in four meta-RL environments are shown in Table. I and Table. II. CRL outperforms MQL in three of four environments significantly, indicating the robustness and effectiveness of our method.

V. CONCLUSIONS

In this paper, we present a robust and efficient off-Policy meta-reinforcement learning algorithm CRL. We assume that similar tasks share common features in low-level neural network layers but have task-specific features in high-level layers. So we propose to disentangle task-specific policy parameters by adapter network to shared low-level parameters which can improve the agent's generalization ability and extract fine grained task-specific features. Furthermore,

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.

TABLE I

MAXIMUM CUMULATIVE REWARD OBTAINED BY THE AGENT IN THE LAST 50000 TIME STEPS ON THE META TEST TASK

Task name	Algorithm			
	SAC	PEARL	MQL	CRL(ours)
Half-Cheetah-Fwd-Back	0.38	1533.73	**1871.12**	1779.23
Ant-Fwd-Back	673.13	869.75	936.43	**972.05**
Humanoid-Direc-2D	**1011.06**	986.79	953.24	**988.42**
Walker-rand-2D	468.15	597.25	527.13	**779.59**

TABLE II

AVERAGE CUMULATIVE REWARD OBTAINED BY THE AGENT IN THE LAST 50000 TIME STEPS ON THE META TEST TASK

Task name	Algorithm			
	SAC	PEARL	MQL	CRL(ours)
Half-Cheetah-Fwd-Back	-0.07	1356.17	**1815.72**	1739.25
Ant-Fwd-Back	654.75	810.59	885.75	**890.71**
Humanoid-Direc-2D	**1007.94**	979.38	934.02	**986.31**
Walker-rand-2D	453.29	552.74	155.80	**740.28**

CRL learns a posterior distribution over probabilistic latent variables to extract universal information across meta-training tasks and perform temporal-extended exploration. Experiment results show that compared with other baseline methods, our method has better asymptotic performance in a sample efficient way on several meta-RL benchmarks.

REFERENCES

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[7] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[8] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "Meta-learning with temporal convolutions," *arXiv preprint arXiv:1707.03141*, vol. 2, no. 7, p. 23, 2017.

[9] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," *Advances in neural information processing systems*, vol. 31, 2018.

[10] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.

[11] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[12] T. Z. Zhao, A. Nagabandi, K. Rakelly, C. Finn, and S. Levine, "Meld: Meta-reinforcement learning from images via latent state models," *arXiv preprint arXiv:2010.13957*, 2020.

[13] R. Fakoor, P. Chaudhari, S. Soatto, and A. J. Smola, "Meta-q-learning," *arXiv preprint arXiv:1910.00125*, 2019.

[14] C. Finn and S. Levine, "Meta-learning: from few-shot learning to rapid reinforcement learning," in *ICML*, 2019.

[15] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, and P. Abbeel, "Promp: Proximal meta-policy search," *arXiv preprint arXiv:1810.06784*, 2018.

[16] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[17] R. Mendonca, X. Geng, C. Finn, and S. Levine, "Meta-reinforcement learning robust to distributional shift via model identification and experience relabeling," *arXiv preprint arXiv:2006.07178*, 2020.

[18] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," *arXiv preprint arXiv:1710.11622*, 2017.

[19] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[20] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[21] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *Advances in neural information processing systems*, vol. 28, 2015.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[23] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[24] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[25] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.

[26] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1704.03012*, 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[29] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[30] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," *arXiv preprint arXiv:1909.09157*, 2019.

[31] S. M. Arnold, S. Iqbal, and F. Sha, "Decoupling adaptation from modeling with meta-optimizers for meta learning," 2019.

[32] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on December 20,2022 at 07:15:25 UTC from IEEE Xplore. Restrictions apply.