Online Meta-Critic Learning for Off-Policy Actor-Critic Methods

Wei Zhou 1  Yiying Li 1,2  Yongxin Yang 2  Huaimin Wang 1  Timothy M. Hospedales 2,3

Abstract

Off-Policy Actor-Critic (Off-PAC) methods have proven successful in a variety of continuous control tasks. Normally, the critic’s action-value function is updated using temporal-difference, and the critic in turn provides a loss for the actor that trains it to take actions with higher expected return. In this paper, we introduce a novel and flexible meta-critic that observes the learning process and meta-learns an additional loss for the actor that accelerates and improves actor-critic learning. Compared to the vanilla critic, the meta-critic network is explicitly trained to accelerate the learning process; and compared to existing meta-learning algorithms, meta-critic is rapidly learned online for a single task, rather than slowly over a family of tasks. Crucially, our meta-critic framework is designed for off-policy based learners, which currently provide state-of-the-art reinforcement learning sample efficiency. We demonstrate that online meta-critic learning leads to improvements in a variety of continuous control environments when combined with contemporary Off-PAC methods DDPG, TD3 and the state-of-the-art SAC.

1. Introduction

Off-policy Actor-Critic (Off-PAC) methods are currently central in deep reinforcement learning (RL) research due to their greater sample efficiency compared to on-policy alternatives. On-policy requires new trajectories to be collected for each update to the policy, and is expensive as the number of gradient steps and samples per step increases with task-complexity even for contemporary TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017) and A3C (Mnih et al., 2016) algorithms.

Off-policy methods, such as DDPG (Lillicrap et al., 2016), TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018b) achieve greater sample efficiency due to their ability to learn from randomly sampled historical transitions without a time sequence requirement, thus making better use of past experience. Their critic estimates the action-value (Q-value) function using a differentiable function approximator, and the actor updates its policy parameters in the direction of the approximate action-value gradient. Briefly, the critic provides a loss to guide the actor, and is trained in turn to estimate the environmental action-value under the current policy via temporal-difference learning (Sutton et al., 2009). In all these cases the learning objective function is hand-crafted and fixed.

Recently meta-learning, or “learning-to-learn” has become topical as a paradigm to accelerate RL by learning aspects of the learning strategy, for example, through learning fast adaptation strategies (Finn et al., 2017; Rakelly et al., 2019; Riemer et al., 2019), exploration strategies (Gupta et al., 2018), optimization strategies (Duan et al., 2016b), losses (Houthooft et al., 2018), hyperparameters (Xu et al., 2018; Veeriah et al., 2019), and intrinsic rewards (Zheng et al., 2018). However, the majority of these works perform meta-learning on a family of tasks or environments and amortize this huge cost by deploying the trained strategy for fast learning on a new task.

In this paper we introduce a novel meta-critic network to enhance existing Off-PAC learning frameworks. The meta-critic is used alongside the vanilla critic to provide a loss to guide the actor’s learning. However, compared to the vanilla critic, the meta-critic is explicitly (meta)-trained to accelerate the learning process rather than merely estimate the action-value function. Overall, the actor is trained by gradients provided by both critic and meta-critic losses, the critic is trained by temporal-difference as usual, and the meta-critic is trained to generate maximum learning performance improvements in the actor. In our framework, both the critic and meta-critic use randomly sampled off-policy transitions for efficient and effective Off-PAC learning, providing superior sample efficiency compared to existing on-policy meta-learners. Furthermore, we demonstrate that our meta-critic can be successfully learned online within a single task. This is in contrast to the currently widely used meta-learning research paradigm – where entire task families are required to provide enough data for meta-learning, and to provide new tasks to amortize the huge cost of meta-learning.
Essentially our framework meta-learns an auxiliary loss function, which can be seen as an intrinsic motivation towards optimum learning progress (Oudeyer & Kaplan, 2009). As analogously observed in several recent meta-learning studies (Franceschi et al., 2018), our loss-learning can be formalized as a bi-level optimization problem with the upper level being meta-critic learning, and lower level being conventional learning. We solve this joint optimization by iteratively updating the meta-critic and base learner online while solving a single task. Our strategy is thus related to the meta-loss learning in EPG (Houthooft et al., 2018), but learned online rather than offline, and integrated with Off-PAC rather than their on-policy policy-gradient learning. The most related prior work is LIRPG (Zheng et al., 2018), which meta-learns an intrinsic reward online. However, their intrinsic reward just provides a helpful scalar offset to the environmental reward for on-policy trajectory optimization via policy-gradient (Sutton et al., 2000). In contrast our meta-critic provides a loss for direct actor optimization just based on sampled transitions, and thus achieves dramatically better sample efficiency than LIRPG reward learning in practice. We evaluate our framework on several contemporary continuous control benchmarks and demonstrate that online meta-critic learning can be integrated with and improve a selection of contemporary Off-PAC algorithms including DDPG, TD3 and SAC.

2. Background and Related Work

Policy-Gradient (PG) Methods. On-policy methods usually update actor parameters in the direction of greater cumulative reward. However, on-policy methods need to interact with the environment in a sequential manner to accumulate rewards and the expected reward is generally not differentiable due to environment dynamics. Even exploiting tricks like importance sampling and improved application of A2C (Zheng et al., 2018), the use of full trajectories is less effective than off-policy transitions, as the trajectory needs a series of continuous transitions in time. Off-policy actor-critic architectures aim to provide better sample efficiency by reusing past experience (previously collected transitions). DDPG (Lillicrap et al., 2016) borrows two main ideas from Deep Q Networks (Mnih et al., 2013; 2015): a big replay buffer and a target Q network to give consistent targets during temporal-difference backups. TD3 (Twin Delayed Deep Deterministic policy gradient) (Fujimoto et al., 2018) develops a variant of Double Q-learning by taking the minimum value between a pair of critics to limit over-estimation. SAC (Soft Actor-Critic) (Haarnoja et al., 2018a;b) proposes a maximum entropy RL framework where its stochastic actor aims to simultaneously maximize expected action-value and entropy. The latest version of SAC (Haarnoja et al., 2018b) also includes the “the minimum value between both critics” idea in its implementation.

Meta Learning for RL. Meta-learning (a.k.a. learning to learn) (Santoro et al., 2016; Finn et al., 2017) has received a resurgence in interest recently due to its potential to improve learning performance, and especially sample-efficiency in RL (Gupta et al., 2018). Several studies learn optimizers that provide policy updates with respect to known loss or reward functions (Andrychowicz et al., 2016; Duan et al., 2016b; Meier et al., 2018). A few studies learn hyperparameters (Xu et al., 2018; Veeriah et al., 2019), loss functions (Houthooft et al., 2018; Sung et al., 2017) or rewards (Zheng et al., 2018) that steer the learning of standard optimizers. Our meta-critic framework is in the category of loss-function meta-learning, but unlike most of these we are able to meta-learn the loss function online in parallel to learning a single extrinsic task rather. No costly offline learning on a task family is required as in Houthooft et al. (2018); Sung et al. (2017). Most current Meta-RL methods are based on on-policy policy-gradient, limiting the sample efficiency. For example, while LIRPG (Zheng et al., 2018) is one of the few prior works to attempt online meta-learning, it is ineffective in practice due to only providing a scalar reward increment rather than a loss for direct optimization. A few meta-RL studies have begun to address off-policy RL, for conventional multi-task meta-learning (Rakelly et al., 2019) and for optimising transfer vs forgetting in continual learning of multiple tasks (Riemer et al., 2019). The contribution of our Meta-Critic is to enhance state-of-the-art Off-PAC RL with single-task online meta-learning.

Loss Learning. Loss learning has been exploited in ‘learning to teach’ (Wu et al., 2018) and surrogate loss learning (Huang et al., 2019; Grabocka et al., 2019) where a teacher network predicts the parameters of a manually designed loss in supervised learning. In contrast our meta-critic is itself a differentiable loss, and is designed for use in reinforcement learning. Other applications learn losses that improve model robustness to out of distribution samples (Li et al., 2019; Balaji et al., 2018). Our loss learning architecture is related to (Li et al., 2019), but designed for accelerating single-task Off-PAC RL rather than improving robustness in multi-domain supervised learning.

3. Methodology

We aim to learn a meta-critic that provides an auxiliary loss $L_{\omega}^{aux}$ to assist the actor’s learning of a task. The auxiliary loss parameters $\omega$ are optimized in a meta-learning process. The main policy loss $L_{\omega}^{main}$ and auxiliary loss $L_{\omega}^{aux}$ train the actor $\pi_\phi$ off-policy via stochastic gradient descent.

3.1. Review of Off-Policy Actor-Critic RL

Reinforcement learning involves an agent interacting with the environment $E$. At each time $t$, the agent receives an observation $s_t$, takes a (possibly stochastic) action $a_t$
Online Meta-Critic Learning for Off-Policy Actor-Critic Methods

based on its policy $\pi : S \rightarrow A$, and receives a scalar reward $r_t$ and new state of the environment $s_{t+1}$. We call $(s_t, a_t, r_t, s_{t+1})$ a single point transition. The objective of RL is to find the optimal policy $\pi^*$, which maximizes the expected cumulative return $J$.

In on-policy RL, $J$ is defined as the discounted episodic return based on a sequential trajectory over the horizon $H$: $J = \mathbb{E}_{t,s_t \sim \rho} \left[ \sum_{t=0}^{H-1} \gamma^t r_t \right]$. In the usual implementation of A2C, $r$ is represented by a surrogate state-value $V(s_t)$ from its critic. Since $J$ is only a scalar value, the gradient of $J$ with respect to policy parameters $\phi$ has to be optimized to maximize the performance on the main task as measured by validation meta loss. Note that in principle the lower-level optimization could purely rely on $L^\text{aux}_\omega$ analogously to the procedure in EPG (Houthooft et al., 2018), but we find that optimizing their sum greatly increases learning stability and speed. Eq. (3) is satisfied when the meta-critic successfully trains the actor for good performance on the main task as measured by validation meta loss. Note that the vanilla critic update is also in the lower loop, but as it updates as usual, so we focus on the actor and meta-critic optimization for simplicity of exposition.

In this setup the meta-critic is a neural network $h_\omega(d_{\text{trn}}; \phi)$ that takes as input some featurisation of the actor $\phi$ and the states and actions in $d_{\text{trn}}$. This auxiliary neural network must produce a scalar output, which we can then treat as a loss $L^\text{aux}_\omega := h_\omega$, and must be differentiable with respect to $\phi$. We next discuss the overall optimization flow, and discuss the specific meta-critic architecture later.

Meta-Optimization Flow. To optimize Eq. (3), we iteratively update the meta-critic parameters $\omega$ (upper-level) and actor and vanilla-critic parameters $\phi$ and $\theta$ (lower-level). At each iteration, we perform: (i) Meta-train: Sample a mini-batch of transitions and putatively update policy $\phi$ according to the main $L^\text{main}_\phi$ and meta-critic $L^\text{aux}_\omega$ losses. (ii) Meta-test: Sample another mini-batch of transitions to evaluate the performance of the updated policy according to $L^\text{meta}_\phi$. (iii) Meta-optimization: Update the meta-critic parameters $\omega$ to maximize the performance on the validation batch, and perform the real actor update according to both losses. In this way the meta-critic is trained online and in parallel to the main task and meta-critic-provided losses on some training samples.
Algorithm 1 Online Meta-Critic Learning for Off-PAC RL

Input: Initialized parameters $\phi$, $\theta$, $\omega$, replay buffer $D \leftarrow \emptyset$
Output: optimized $\phi$

1. begin
2. for each iteration do
3. for each environment step do
4. $a_t \sim \pi_\phi(s_t)$
5. $s_{t+1} \sim p(s_{t+1}|s_t, a_t), r_t$
6. $D \leftarrow D \cup \{(s_t, a_t, r_t, s_{t+1})\}$
7. for each gradient step do
8. Sample mini-batch $d_{trn}$ from $D$
9. $\theta \leftarrow \theta - \lambda \nabla_\theta J_Q(\theta)$ // Update critic
10. Meta-train:
11. $L_{\text{main}} \leftarrow \text{Eqs. (1), (8) or (9)}$ // Main actor loss
12. $L_{\text{aux}} \leftarrow \text{Eqs. (6) or (7)}$ // Auxiliary actor loss
13. $\phi_{\text{old}} = \phi - \alpha \nabla_\phi L_{\text{main}}$
14. $\phi_{\text{new}} = \phi_{\text{old}} - \alpha \nabla_\phi L_{\text{aux}}$
15. meta-test:
16. Sample mini-batch $d_{val}$ from $D$
17. $L_{\text{meta}}(d_{val}; \phi_{\text{old}}, \phi_{\text{new}}) \leftarrow \text{Eq. (4)}$
18. meta-optimization:
19. $\phi \leftarrow \phi - \eta (\nabla_\phi L_{\text{main}} + \nabla_\phi L_{\text{aux}})$ // Update actor
20. $\omega \leftarrow \omega - \eta \nabla_\omega L_{\text{meta}}$ // Update meta-critic
21. end
22. end

we find it helpful for optimization efficiency to optimize the difference between the updates with- and without meta-critic’s input. Specifically, we use

$$L_{\text{meta}} = \tanh(L_{\text{main}}(d_{val}; \phi_{\text{new}}) - L_{\text{main}}(d_{val}; \phi_{\text{old}})),$$

which is simply a monotonic re-centering and re-scaling of $L_{\text{main}}$. This leads to

$$\omega \leftarrow \arg \min \tanh(L_{\text{main}}(d_{val}; \phi_{\text{new}}) - L_{\text{main}}(d_{val}; \phi_{\text{old}})).$$

Note that here the updated actor $\phi_{\text{new}}$ has dependence on the feedback given by meta-critic $\omega$ and $\phi_{\text{old}}$ does not. Thus only the first term is optimized for $\omega$. In his setup the $L_{\text{main}}(d_{val}; \phi_{\text{new}})$ term should obtain high reward/low loss on the validation batch and the latter provides a baseline, analogous to the baseline widely used to accelerate and stabilize policy-gradient RL. The use of tanh reflects the idea of diminishing marginal utility, and ensures that the meta-loss range is always nicely distributed in $(-1, 1)$. In essence, the meta-loss is for the agent to ask itself the question: “Did meta-critic improve validation performance?” and adjusts the meta-critic (auxiliary task) parameters accordingly.

Designing Meta-Critic ($h_\omega$). The meta-critic network $h_\omega$ implements the auxiliary loss for the actor. The design-space for $h_\omega$ has several requirements: (i) Its input must depend on the policy parameters $\phi$, because this auxiliary loss is also used to update policy network. (ii) It should be permutation invariant to transitions in $d_{trn}$, i.e., it should not make a difference if we feed the randomly sampled transitions indexed $[1,2,3]$ or $[3,2,1]$. The naivest way to achieve (i) is given in MetaReg (Balaji et al., 2018) which meta-learns a parameter regularizer: $h_\omega(\phi) = \sum_i \omega_i |\phi_i|$. Although this form of $h_\omega$ acts directly on $\phi$, it does not exploit state information, and introduces a large number of parameters in $h_\omega$, as $\phi$ may be a high-dimensional neural network. Therefore, we design a more efficient and effective form of $h_\omega$ that also meets both of these requirements. Similar to the feature extractor in supervised learning, the actor needs to analyse and extract information from states for decision-making. We assume the policy network can be represented as $\pi_\phi(s) = \tilde{\pi}(\tilde{\pi}(s))$ and decomposed into the feature extraction $\tilde{\pi}_\phi$ and decision-making $\tilde{\pi}_\phi$ (i.e., the last layer of the full policy network) modules. Thus the output of the penultimate layer of full policy network is just the output of feature extraction $\tilde{\pi}_\phi(s)$, and such output of feature joint encodes $\phi$ and $s$. Given this encoding, we implement $h_\omega(d_{trn}; \phi)$ as a three-layer multi-layer perceptron (MLP) whose input is the extracted feature from $\tilde{\pi}_\phi(s)$. Here we consider two designs for meta-critic ($h_\omega$): using our joint feature alone (Eq. (6)) or augmenting the joint feature with

$$L_{\text{meta}} = \tanh(L_{\text{main}}(d_{val}; \phi_{\text{new}}) - L_{\text{main}}(d_{val}; \phi_{\text{old}})),$$

**Note that the parameter $\omega$ that minimises $L_{\text{meta}}$ as Eq. 4 is also the minimum of $L_{\text{main}}$ and vice-versa.**
Online Meta-Critic Learning for Off-Policy Actor-Critic Methods

states and actions (Eq. (7)):

\begin{equation}
(i) \quad h_w(d_{trn}; \phi) = \frac{1}{N} \sum_{i=1}^{N} \text{MLP}_\omega(\pi_\phi(s_i)),
\end{equation}

\begin{equation}
(ii) \quad h_w(d_{trn}; \phi) = \frac{1}{N} \sum_{i=1}^{N} \text{MLP}_\omega(\pi_\phi(s_i), s_i, a_i).
\end{equation}

\(h_w\) is to work out the auxiliary loss based on such batch-wise set-embedding (Zaheer et al., 2017) of our joint actor-state feature. That is to say, \(d_{trn}\) is a randomly sampled mini-batch transitions from the replay buffer, and then the \(s\) (and \(a\)) of the transitions are inputted to the \(h_w\) network in a permutation invariant way, and finally we can obtain the auxiliary loss for this batch \(d_{trn}\). Here, our design of Eq. (7) also includes the cues features in LIRPG and EPG where \(s_i\) and \(a_i\) are used as the input of their learned reward and loss respectively. We set a softplus activation to the final layer of \(h_w\), following the idea in TD3 that the vanilla critic may over-estimate and so the introduction of a non-negative actor auxiliary loss can mitigate such over-estimation. Moreover, we point out that only \(s_i\) (and \(a_i\)) from \(d_{trn}\) are used when calculating \(L^{\text{main}}\) and \(L^{\text{aux}}\) for the actor, while \(s_i, a_i, r_i\) and \(s_{i+1}\) are all used for optimizing the vanilla critic.

**Implementation on DDPG, TD3 and SAC.** Our meta-critic module can be incorporated in the main Off-PAC methods DDPG, TD3 and SAC. In our framework, these algorithms differ only in their definitions of \(L^{\text{main}}\), and the meta-critic implementation is otherwise exactly the same for each. Further implementation details can be found in the supplementary material.

DDPG (Fujimoto et al., 2018) borrows the Double Q-learning idea and use the minimum value under both critics to make unbiased value estimates. At the same time, computational cost is obtained by using a single actor optimized with respect to \(Q_{\theta_1}\). Thus the corresponding \(L^{\text{main}}\) for actor becomes:

\(L^{\text{main}} = -\mathbb{E}_{s \sim p_s} Q_{\theta_1}(s, a) | a = \pi_\phi(s)\).

In SAC, two key ingredients are considered for the actor: maximizing the policy entropy and automatic temperature hyper-parameter regulation. At the same time, the latest version of SAC (Haarnoja et al., 2018b) also draws lessons from “taking the minimum value between both critics”. The \(L^{\text{main}}\) for SAC actor is:

\(L^{\text{main}} = \mathbb{E}_{s \sim p_s} \left[ \alpha \log (\pi_\phi(a|s)) - Q_{\phi}(s, a) | a = \pi_\phi(s) \right].\)

4. Experiments and Evaluation

The goal of our experimental evaluation is to demonstrate the versatility of our meta-critic module in integration with several prior Off-PAC algorithms, and its efficacy in improving their respective performance. We use the open-source implementations of DDPG, TD3 and SAC algorithms as our baselines, and denote their enhancements by meta-critic as DDPG-MC, TD3-MC, SAC-MC respectively. All -MC agents have both their built-in vanilla critic, and the meta-critic that we propose. We take Eq. (6) as the default meta-critic architecture \(h_w\), and we compare the alternative in the later ablation study. For our implementation of meta-critic, we use a three-layer neural network with an input dimension of \(\pi\) (300 in DDPG and TD3, 256 in SAC), two hidden feed-forward layers of 100 hidden nodes each, and ReLU non-linearity between layers.

We evaluate the methods on a suite of seven MuJoCo continuous control tasks (Todorov et al., 2012) in OpenAI Gym (Brockman et al., 2016), two MuJoCo tasks in rllab (Duan et al., 2016a), and the simulated racing car environment TORCS (Loiacono et al., 2013). For MuJoCo-Gym, we use the latest V2 tasks instead of V1 used in TD3 and the old-SAC (Haarnoja et al., 2018a) work without any modification to their original environment or reward.

**Implementation Details.** For DDPG, we use the open-source implementation “OurDDPG” which is the re-tuned version of DDPG implemented in Fujimoto et al. (2018) with the same hyper-parameters of the actor and critic for MuJoCo tasks. For TD3 and SAC, we use the open-source implementations of TD3 and SAC. In MuJoCo cases we integrate our meta-critic with learning rate 0.001. The hyper-parameters for TORCS can be found in the supplementary material.

4.1. Evaluation of Meta-Critic Off-PAC learning

**DDPG** Figure 2 shows the learning curves of DDPG and DDPG-MC. The experimental results corresponding to each task are averaged over 5 random seeds (trials) and network initialisations, and the standard deviation confidence intervals are represented as shaded regions over the time steps. Following (Fujimoto et al., 2018), curves are uniformly smoothed for clarity (window size=10 for TORCS, 30 for others). We run the gym-MuJoCo experiments for 1-10 million depending on to environment, rllab experiments for 3 million steps and TORCS experiment for 100 thousand steps. Every 1000 steps we evaluate our policy over 10 episodes with no exploration noise.

From the learning curves in Figure 2, we can see that DDPG-MC generally outperforms the corresponding DDPG baseline in terms of the learning speed and asymptotic performance. Furthermore, it usually has smaller variance. The summary results for all tasks in terms of max average return are given in Table 1. -MC usually provides consis-
Online Meta-Critic Learning for Off-Policy Actor-Critic Methods

Figure 2. Learning curve Mean and Std-Deviation of vanilla DDPG and meta-critic enhanced DDPG-MC for continuous control tasks.

Table 1. Max Average Return over 5 trials over all time steps. Max value for each task is bolded.

| Environment     | DDPG | DDPG-MC | TD3  | TD3-MC | SAC  | SAC-MC | PPO  | PPO-LIRPG |
|-----------------|------|---------|------|--------|------|--------|------|-----------|
| HalfCheetah     | 8440.2 | 10187.5 | 12735.7 | 15064.0 | 16651.8 | 16815.9 | 20861.5 | 1882.6 |
| Hopper          | 2097.5 | 3253.6 | 3807.0 | 3854.3 | 3610.6 | 3738.4 | 3762.0 | 2750.0 |
| Walker2d        | 2920.1 | 3733.7 | 5942.7 | 5955.5 | 6398.8 | 7164.9 | 4432.6 | 3652.9 |
| Ant             | 2375.4 | 3661.1 | 5914.8 | 6280.0 | 6954.4 | 7204.3 | 684.2  | 23.6     |
| Reacher         | -3.6  | -3.7   | -3.0  | -2.9   | -2.8  | -2.7   | -6.08  | -7.53    |
| InvPend         | 1000.0 | 1000.0 | 1000.0 | 1000.0 | 1000.0 | 1000.0 | 1000.0 | 1000.0  |
| InvDooPend      | 9307.5 | 9326.5 | 9357.4 | 9358.7 | 9359.6 | 9359.6 | 7266.0 | 6974.9  |
| HalfCheetah(rllab) | 5860.8 | 6254.6 | 8029.6 | 8552.1 | 10011.0 | 10597.0 | -     | -        |
| Ant (rllab)     | 2500.8 | 2721.1 | 3672.6 | 4776.8 | 8014.8 | 8353.8 | -     | -        |
| TORCS           | 6188.1 | 9353.3 | 14841.7 | 33684.2 | 25020.6 | 32869.0 | -     | -        |

Figure 3 reports the learning curves for TD3. For some tasks vanilla TD3 performance declines in the long run, while our TD3-MC shows improved stability with much higher asymptotic performance. Generally speaking, the learning curves show that TD3-MC providing comparable or better learning performance in each case, while Table 1 shows the clear improvement in the max average return.

Figure 4 report the learning curves of SAC. Note that we use the most recent update of SAC (Haarnoja et al., 2018b), which can be regarded as the combination SAC+TD3. Although this SAC+TD3 is arguably the strongest existing method, SAC-MC still gives a clear boost on the asymptotic performance for several of the tasks.

Comparison vs PPO-LIRPG Intrinsic Reward Learning for PPO (Zheng et al., 2018) is the most related method to our work in performing online single-task meta-learning of an auxiliary reward/loss via a neural network. The original PPO-LIRPG study evaluated on a modified environment with hidden rewards. Here we apply it to the standard unmodified learning tasks that we aim to improve. The results in Table 1 demonstrate that: (i) In this conventional setting, PPO-LIRPG worsens rather than improves basic PPO performance. (ii) Overall Off-PAC methods generally perform better than on-policy PPO for most environments. This shows the importance of our meta-learning contribution to the off-policy setting. In general Meta-Critic is preferred compared to PPO-LIRPG because the latter only provides a scalar reward bonus only influences the policy indirectly via high-variance policy-gradient updates, while Meta-Critic provides a direct loss.

Summary Table 1 and Figure 5 summarize all the results in terms of max average return. We can see that SAC-MC generally performs best; the Meta-Critic-enhanced methods are generally comparable or better than their corresponding vanilla alternatives; and Meta-Critic usually provides improved variance in return compared to the baselines.

4.2. Further Analysis

Loss Analysis. To analyse the learning dynamics of our algorithm, we take a simple learning problem, tabular MDP (Duan et al., 2016b) (|S| = 2, |A| = 2) as an example, and
We see some interesting behavior in these results. Figure 6 compares DDPG vs DDPG-MC. Figure 6 reports the main loss $L^{\text{main}}$ curves of actor and the loss curve of $h_{\omega}$ (i.e., $L^{\text{aux}}$) and $L^{\text{meta}}$ over 5 trials for DDPG-MC. In addition, we plot the model optimization trajectories (pink dots) via a 2D weight-space slice in Figure 7. These are plotted over the average reward surface for this slice. Following the neural network visualization method of Li et al. (2018), we calculate the subspace to plot as: Let $\phi_i$ denote model parameters at episode $i$ and the final estimate as $\phi_n$ (we set $n = 100$). We apply PCA to the matrix $M = [\phi_0 - \phi_n, \ldots, \phi_{n-1} - \phi_n]$, and take the two most explanatory directions of this optimization path. Model parameters are then projected onto the plane defined by these directions for plotting; and models at each point on this plane are densely evaluated to calculate average reward.

We see some interesting behavior in these results. Figure 6 shows: (i) DDPG-MC shows faster convergence to a lower value of $L^{\text{main}}$, demonstrating the auxiliary loss’s ability to accelerate learning. (ii) The meta-loss (which corresponds to the success of the meta-critic in improving actor learning) shows a pattern: ‘positive’ → ‘negative’ → ‘converging to zero’. This pattern is expected because: At the start, $h_{\omega}$ is randomly initialised and knows little about how to help the actor, thus $\phi_{\text{old}}$-based model outperforms $\phi_{\text{new}}$-based model. Then as $\omega$ is trained by the meta-loss, it begins to make $\phi_{\text{new}}$ better than $\phi_{\text{old}}$. In the late stage, meta-loss goes towards zero, which indicates that all of $h_{\omega}$’s knowledge has been distilled to help the actor. (iii) The auxiliary loss converges smoothly under the supervision of the meta-loss. In Figure 7 (iv) DDPG-MC has a very direct optimization trajectory to the high reward zone, while the vanilla DDPG model moves slowly through the low reward space and before finally finding the direction to the high-reward zone.

**Ablation on $h_{\omega}$ design.** To analyse the designs of $h_{\omega}$, we run Walker2d experiments under SAC-MC with the alternative $h_{\omega}$ architecture from Eq. (7) or MetaReg (Balaji et al., 2018) format (input actor parameters directly). As shown in Table 2, we record the max average return and sum average return (regarded as the area under the average reward curve) of all evaluations during all time steps. Eq. (7) achieves the highest max average return and our default $h_{\omega}$ (Eq. (6)) attains the highest mean average return. We can also see...
some improvement for $h_\omega(\phi)$ using MetaReg format, but the huge number (73484) of parameters is expensive. Overall, all meta-critic module designs provides at least a small improvement on vanilla SAC.

**Ablation on baseline in meta-loss.** In Eq. (4), we use $L_\text{main}(d_{\text{val}}; \phi_{\text{old}})$ as a baseline to improve numerical stability of the gradient update. To evaluate this design, we remove the $\phi_{\text{old}}$ baseline and optimize $\omega \leftarrow \arg \min_\omega \text{tanh}(L_\text{main}(d_{\text{val}}; \phi_{\text{new}}))$. The last column in Table 2 shows that this barely improves on vanilla SAC, validating our design choice to use a baseline.

**5. Conclusion**

We present Meta-Critic, an auxiliary critic module for Off-PAC methods that can be meta-learned online during single task learning. The meta-critic is trained to generate gradients that improve the actor’s learning performance over time, and leads to long run performance gains in continuous control. The meta-critic module can be flexibly incorporated into various contemporary Off-PAC methods to boost performance. In future work, we plan to apply the meta-critic to conventional meta-learning with multi-task and multi-domain RL.

**References**

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and Freitas, N. D. Learning to learn by gradient descent by gradient descent.
In *NIPS*, 2016.

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. In *arXiv*, 2016.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016a.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. *RL2*: Fast reinforcement learning via slow reinforcement learning. In *arXiv*, 2016b.

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

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

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

Grabocka, J., Scholz, R., and Schmidt-Thieme, L. Learning surrogate losses. In *arXiv*, 2019.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *NeurIPS*, 2018.

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

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. In *arXiv*, 2018b.

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

Huang, C., Zhai, S., Talbott, W., Bautista, M. Á., Sun, S., Guestrin, C., and Susskind, J. Addressing the loss-metric mismatch with adaptive loss alignment. In *ICML*, 2019.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.
Sung, F., Zhang, L., Xiang, T., Hospedales, T., and Yang, Y. Learning to learn: meta-critic networks for sample efficient learning. In arXiv, 2017.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In NIPS, 2000.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvri, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In ICML, 2009.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In IROS, 2012.

Veeriah, V., Hessel, M., Xu, Z., Lewis, R., Rajendran, J., Oh, J., van Hasselt, H., Silver, D., and Singh, S. Discovery of useful questions as auxiliary tasks. In NeurIPS, 2019.

Wu, L., Tian, F., Xia, Y., Fan, Y., Qin, T., Jian-Huang, L., and Liu, T.-Y. Learning to teach with dynamic loss functions. In NeurIPS, 2018.

Xu, Z., van Hasselt, H., and Silver, D. Meta-gradient reinforcement learning. In NeurIPS, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In NIPS, 2017.

Zheng, Z., Oh, J., and Singh, S. On learning intrinsic rewards for policy gradient methods. In NeurIPS, 2018.