IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 10, 2019, accepted April 29, 2019, date of publication May 2, 2019, date of current version May 13, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2914469*

# Learning to Learn: Hierarchical Meta-Critic Networks

**ZHIXIONG XU, LEI CAO, AND XILIANG CHEN**
College of Command and Control Engineering, Army Engineering University, Nanjing 210000, China

Corresponding author: Zhixiong Xu (xu.nj@foxmail.com)

This work was supported in part by the National Natural Science Fund Projects under Grant 61203192, and in part by the Natural Science Fund Project of Jiangsu province under Grant BK2011124.

**ABSTRACT** In recent years, deep reinforcement learning methods have achieved impressive performance in many different fields, including playing games, robotics, and dialogue systems. However, there are still a lot of restrictions here, one of which is the demand for massive amounts of sampled data. In this paper, a hierarchical meta-learning method based on the actor-critic algorithm is proposed for sample efficient learning. This method provides the transferable knowledge that can efficiently train an actor on a new task with a few trials. Specifically, a global basic critic, meta critic, and task specified network are shared within a distribution of tasks and are capable of criticizing any actor trying to solve any specified task. The hierarchical framework is applied to a critic network in the actor-critic algorithm for distilling meta-knowledge above the task level and addressing distinct tasks. The proposed method is evaluated on multiple classic control tasks with reinforcement learning algorithms, including the start-of-the-art meta-learning methods. The experimental results statistically demonstrate that the proposed method achieves state-of-the-art performance and attains better results with more depth of meta critic network.

**INDEX TERMS** Deep reinforcement learning, hierarchical framework, knowledge, meta-learning.

## I. INTRODUCTION

Despite deep reinforcement learning (DRL) has successfully solved both simulated and real-world tasks, such sophisticated and large-scale task environments as Atari [1] and Go [2], there is still one limit in DRL compared with human performance [3]. That is sample efficiency. DRL has to learn from scratch and require far more experience than humans, while humans and animals are able to learn a new task in a very small number of trials and flexibly adapt to changing task conditions.

In recent years, there is a large body of related research seeking to improve the sample efficiency of DRL, including the incorporation of a good prior. Flexible, data-efficient learning naturally requires the operation of prior biases [4]. One of sources of such biases has been explored in the machine learning literature under the rubric of meta learning [5], [6], which also has various alternative names, including life-long learning, learning to learn, etc.

In this paper, the authors consider solving distributions of related tasks, with the goal of learning a new task quickly. How to learn common knowledge, also called meta knowledge, between different tasks is the key to learning a new task efficiently. Motivated by the vision of meta learning and the framework of Actor-Critic [7] method where the actor network outputs actions according to states, and the critic network learns to criticize the action made by the actor network, the authors propose a hierarchical meta-critic method to learn meta knowledge shared between different tasks by training multiple actors at given tasks.

The authors apply hierarchical framework to the proposed method for helping agents to learn knowledge above the task level. The framework of the proposed approach is somewhat similar to the Options [8] in hierarchical reinforcement learning, but applied to the setting of a task distribution. To achieve this vision, the authors introduce the idea of task specified network into actor-critic method for encoding the information of tasks into the learning structure. The task specified network reads in a series of trajectory data of the current task, and produces a task specified embedding, so as to encode the information of the current task into meta-critic network.

An end-to-end training approach is also designed for the proposed model that allows for effectively solving a new task, treated solely by learning a hierarchical meta-critic network.

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

In sum, the contributions of this paper are outlined as follows.

- The authors propose a novel meta learning framework based on actor-critic algorithm with a global basic critic, meta critic and task specified network that learn to criticize any actor at any given tasks. The authors regard multiple actor networks as 'options' in the Options framework for learning the distributions of related tasks during training.
- The authors design a new end-to-end training method to consider learning basic/meta critic and task specified networks in the upper structure of the model, and actor networks in the underlying structure of the model respectively.
- The empirical evaluation results show that the effectiveness of the proposed method, which not only outperforms pure deep reinforcement learning algorithms on a new task, but also is comparable to other state-of-the-art methods in meta learning.

The organization of this paper is as follows. Related work is introduced in Section 2. In Section 3, the authors formulate an optimization problem and present the framework and training details of the proposed method. In Section 4, the authors invalid the proposed algorithm on a wide range of classic control tasks, and discuss the effect of depth of networks on the model. Concluding remarks with a discussion of the extended work of the proposed approach are in the final section.

## II. RELATED WORK

The goal of prior work in hierarchical reinforcement learning (HRL) is to improve the learning efficiency by recombining a set of temporally extended primitives, among which one of the most classic framework is Options framework. Previous work about options framework mostly assume that the options are designed in advance, more recent work seek to learn the options automatically [9], [10]. Vezhnevets *et al*. [11] propose an architecture in which a high-level controller explicitly sets sub-goals for and provides appropriate rewards to a low-level controller. Florensa *et al.* [12] present a general framework that first learns useful skills in a pre-training environment, and then leverages the acquired skills for learning faster in downstream tasks. Bacon *et al*. [13] propose a new option-critic architecture capable of learning both the internal policies and the termination conditions of options, in tandem with the policy over options, and without the need to provide any additional rewards or sub-goals. Levy *et al*. [14] present a novel approach to hierarchical reinforcement learning called Hierarchical Actor-Critic (HAC) that enables agents to learn to break down problems involving continuous action spaces into simpler sub-problems belonging to different time scales. Yang *et al*. [15] propose to learn basic skills and compound skills simultaneously through a hierarchical deep reinforcement learning algorithm, which contains two levels of hierarchy, in the first level of hierarchy, multiple basic skills are learned simultaneously, while the second level of hierarchy is designed for learning compound skills. The experimental results have proved that the proposed algorithm can learn both basic skills and compound skills on a Pioneer 3AT robot in three different navigation scenarios. Other approaches [16]–[18] are committed to learning a decomposition of complex mixed task tasks into sub-goals. These previous work almost focus on the single and static task, while ignoring multi-task and dynamic tasks setting. This paper considers solving the distributions of related tasks by distilling meta knowledge shared between different tasks, with the goal of learning a shared high-level network architecture.

Meta learning has a long history, but has grown to prominence recently as many have advocated for it as a key to achieving human-level intelligence in the future [19]. The goal of meta learning is to take advantage of a variety of related tasks to train single model, such that it is able to solve a new task with only a very small number of trials. At present, the popular meta learning methods fall into two categories: one is utilizing gradients in past training tasks to help initialize the networks for a new task. Finn *et al.* [20] propose a Model-Agnostic Meta Learning (MAML) method for fast adaptation of deep networks, which doesn't introduce additional parameters for metal-learning nor require a particular learner architecture. Al-Shedivat *et al*. [21] develop a gradient-based meta learning algorithm similar to MAML algorithm and suitable for continuous adaptation of RL agents in nonstationary environments. More concretely, the agents are capable of learning to anticipate the changes in the environment and update their policies accordingly. Xu *et al*. [22] discuss how to learn the meta-parameters of a return function while interacting with the environment, and show that adjusting the meta-parameters of basic deep reinforcement learning algorithms could help achieve much higher performance than previously observed on Atari 2600 games [23].

The other one is engineering the prior biases into the learning system for remembering meta knowledge shared by different tasks. The representative method is RNN-based (Recurrent Neural Network), which can be classified into two main categories. The first one is using a recurrent model $r$ as the meta-learner with parameters $\theta$, which takes as input the training dataset $D_{tr}$ for a particular task $T$ and a new test input $x_{te}$, and produces the estimated output $y_{te}$ for that input:

$$y_{te} = r(D_{tr}, x_{te}; \theta) = r((x_1, y_1), \ldots, (x_k, y_k), x_{te}; \theta) \quad (1)$$

Wang *et al*. [4] introduce a novel method called Deep meta-RL, which consists three key elements: (1) Utilize a deep reinforcement learning algorithm to train a recurrent model, (2) The training dataset involves the distribution of related tasks, not a single task, (3) The action chose by the agent and the reward received from the environment in the previous time step will be used as the additional input to the current neural network. Santoro *et al*. [24] introduce a memory-augmented neural network to be trained for storing and retrieving memories on each classification task.

Duan *et al.* [25] encode reinforcement learning algorithm with the weights of the RNN, which are learned slowly by a general-purpose reinforcement learning algorithm. Besides, multiple episodes from a series of different MDPs (Markov Decision Process) is used as the input of the RNN, and a policy gradient update through the whole temporal span of the RNN. Mishra *et al.* [26] present a novel and generic architecture, which combines temporal convolutions and soft attention as a meta-learner. The temporal convolutions are able to aggregate information from experience in the past, and the soft attention will pinpoint specific pieces of information.

The second one is taking as input the training dataset $D_{tr}$ for a particular task $T$ and the parameters $\varphi$ of a learner model $c$, then, the meta-learner $r$ outputs new parameters $\varphi^*$ for the learner model. Then, the test input $x_{te}$ is fed into the learner model to produce the predicted output $y_{te}$:

$$\begin{aligned} y_{te} &= r(x_{te}; \varphi^*) = r(x_{te}; c(D_{tr}; \varphi)) \\ &= r(x_{te}; c((x_1, y_1), \ldots, (x_k, y_k); \varphi)) \end{aligned} \tag{2}$$

Andrychowicz *et al.* [27] consider a recurrent neural network as the output of meta learning, which is treated as an optimization method to fit other models to data. Ravi & Larochelle [28] propose an LSTM-based meta learning model as a optimization method for training another neural network classifier in the few-shot regime. Bertinetto *et al.* [29] map a training dataset to the weights of a deep neural network by training a meta-learner, which is utilized to classify future examples. Maclaurin *et al.* [30] successfully optimize the validation performance by the way of adapting the hyper-parameters of gradient descent, which back-propagates through the chain of gradient steps.

The above contents briefly introduce previous work on hierarchical reinforcement learning and employing RNN in the context of meta learning. The proposed HMCN (Hierarchical Meta-Critic Networks) method not only takes advantage of the characteristics of hierarchical framework to add another learning structure into actor-critic method, but also distills meta knowledge from the distribution of related tasks by learning a global basic/meta critic and task specified networks, which are capable of supervising any actor trying to solve any specified task. The authors will introduce the Hierarchical Meta-Critic Networks method in the following content.

## III. METHODOLOGY

Firstly, the authors introduce the problem statement and the background of the proposed method (Section A), then, the proposed method and the training process are explained in detail (Section B).

### A. BACKGROUND

In the reinforcement learning problem, at each time $t$, an agent will take an action $a_t$ according to the observed state $s_t$ and the received reward $r_t$, which is based on the previous action $a_{t-1}$ acting on the environment, then,

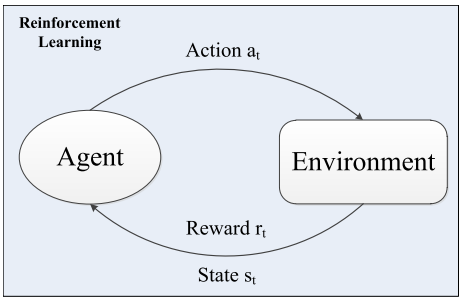



**FIGURE 1.** The model of standard reinforcement learning.

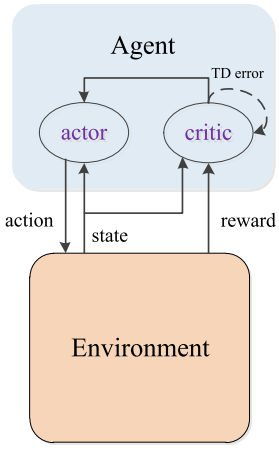



**FIGURE 2.** The model of Actor-Critic method.

the environmental state $s_t$ transfers to the state $s_{t+1}$ at time $t+1$. As shown in Figure 1.

The reinforcement learning algorithms study a specific, single task $M$ with a Markov decision process, while in meta learning scenario, a distribution of different but related tasks are considered. The goal of meta learning is to find a strategy $\pi_\theta$ that can quickly learn new tasks $M_i$:

$$\min_{\theta} \sum_{M_i} \mathrm{E}_{\pi_\theta}[L_{M_i}] \tag{3}$$

where $\mathrm{E}_{\pi_\theta}[]$ represents the average expectation, $L_{M_i} = -\sum_{t=1}^{H} R_i(s_t, a_t)$ represents the cumulative loss of the task $M_i$, $H$ is the length of the episode, and $R(s, a)$ represents the reward function in reinforcement learning.

The proposed method is based on the actor-critic (AC) algorithm, which uses neural networks as value function approximators for critic and actor. The critic is the state-action value function [31] and estimates the future discount return by minimizing the TD error (temporal difference error), while the actor outputs the actual action [32], [33]. As shown in Figure 2.

The critic network with weight parameters $\varphi$ takes as input the state $s_t$ and action $a_t$, outputs the future discount return: $Q^{\pi_\theta}(s_t, a_t; \phi) = r_t + \gamma Q^{\pi_\theta}(s_{t+1}, a_{t+1}; \phi)$, where the notation $Q^{\pi_\theta}$ refers to the critic is trained to model the return

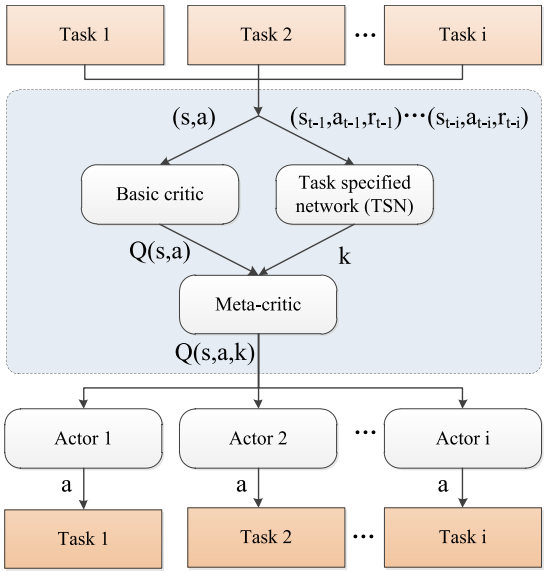



**FIGURE 3.** Diagram of hierarchical meta-critic network architecture.

of policy $\pi_\theta$. The goal of the critic network is to predict the future discount return accurately.

The actor network with weight parameters $\theta$ takes as input the state $s_t$, outputs an actual action $a_t$, where $a_t = \pi_\theta(s_t)$. The policy $\pi_\theta$ defined in reinforcement learning outputs a distribution over actions given a state. The goal of the actor network is to maximize the future discount return, assumed to be the return estimated by the critic network. The process of optimization is to alternatively update the actor network and critic network:

$$\theta \leftarrow \underset{\theta}{argmax}\, Q^{\pi_\theta}(s_t, a_t; \phi) \tag{4}$$

$$\phi \leftarrow \underset{\phi}{argmin}(Q^{\pi_\theta}(s_t, a_t; \phi)) - r_t - \gamma Q^{\pi_\theta}(s_{t+1}, a_{t+1}; \phi))^2 \tag{5}$$

where $a_t = \pi_\theta(s_t)$ and $a_{t+1} = \pi_\theta(s_{t+1})$.

The goal of the proposed method is to learn a policy for a new test task $t^*$ by leveraging the background tasks $T$, in the case of considering a set of training tasks $T = \{t_i\}$ plus allowed environmental interactions. In contrast to other meta learning methods solving this problem, the authors propose a novel and flexible meta learning method for knowledge transfer by adding an high-level network architecture for distilling meta knowledge from multiple source tasks.

### B. HIERARCHICAL META-CRITIC NETWORKS ALGORITHM

As shown in Figure 3, the authors propose a novel framework for incorporating meta knowledge into the architecture of networks. The core of the proposed framework is establishing a high-level network structure for learning meta knowledge, which is based on the perspective of learning a global meta-critic network from a basic critic and a task specified network by supervising multiple actors at given tasks.

#### 1) BASIC CRITIC AND TASK SPECIFIED NETWORK

In order to help agents learn the knowledge of the task level, the authors introduce a task specified network to encode the feature of tasks via a task embedding $k^t_\phi$. The task specified network takes a series of trajectory data

$$\Gamma^t_{t-i} = \{(s_{t-i}, a_{t-i}, r_{t-i}), \ldots, (s_{t-1}, a_{t-1}, r_{t-1})\} \tag{6}$$

as input and outputs a task specified encoding $k^t_\phi$. The past state-action pairs as input reveal the characteristics of the actor it is to criticize, and the past rewards record the feature of the task the actor is solving.

Different from Actor-Critic method proposed before, the authors additionally add a basic critic network to record the current state-action value $Q_\varphi(s_t, a_t)$. The authors believe that it is necessary to separate the evaluation of the current state from the meta critic network through a basic critic network, which is beneficial for meta-critic to consider both historical information and current information respectively. In the end, the output of the task specified network and the output of the basic critic are concatenated together as the input of meta-critic network.

#### 2) META-CRITIC NETWORK

Basic, meta-critic and task specified network are shared across all tasks and actors. Assuming $L$ tasks, the update formula of meta-critic and actors are:

$$\theta = \underset{\theta}{\text{argmax}}\, Q_\beta(s^i_t, a^i_t, k^i_t) \quad \forall i \in \{1, 2, \ldots, L\} \tag{7}$$

$$\beta, \phi, \varphi = \underset{\beta,\phi,\varphi}{\text{argmax}} \sum\nolimits_{i=1}^{L} (Q_\beta(s^i_t, a^i_t, k^i_t) - r^i_t - \gamma Q_\beta(s^i_{t+1}, a^i_{t+1}, k^i_{t+1}))^2 \tag{8}$$

When optimizing $\theta$, $k^i_t$ is regarded as a constant rather than a function. The authors divide the learning proceeds of meta learning into two stage: meta training stage and specific actor training stage. In meta training stage, the authors train multiple actors on multiple source tasks along with a shared high-level network architecture, which contains a basic critic, meta critic and task specified networks. In specific actor training stage, the authors train an actor at given tasks with a very small number of trials, where the parameters of the basic/meta critic and task specified network are fixed. The processes of two stages are summarized in Table 1 and 2.

## IV. EXPERIMENTS

### A. IMPLEMENTATION AND DETAILS

For all the following experiments, the authors tested four algorithms: Actor-Critic (AC): Both critic and actor network are trained from scratch directly for each new task. Model-Agnostic Meta Learning (MAML): One of state-of-the-art meta learning methods. Meta-Critic (MC): recently proposed meta learning method for sample efficient learning. Hierarchical Meta-Critic Networks (HMCN): The proposed method as describe above.

For HMCN method, the architecture of neural networks are as follows: (i) Basic critic is a three-layer MLP network,

**TABLE 1. Meta training stage.**

| Stage 1: Meta Learning Stage of HMCN |
|---|
| Input: The distribution of related tasks $T$ |
| Output: Trained basic critic and meta-critic networks |
| Output: Trained task specified network |
| Init: basic critic and meta-critic networks; |
| Init: task specified network; |
| For $episode = 1,...,M$ do |
| Generate $L$ tasks from $T$; |
| Init: $L$ actor networks; |
| For $step = 1,...,m$ do |
| Sample batches of tasks; |
| For each task in batches do |
| Sample training data from task; |
| Train corresponding actor; |
| End |
| Train meta-critic network; |
| Train basic critic |
| Train task specified network; |
| End |
| End |

**TABLE 2. Specific actor training stage.**

| Stage 2: specific actor training stage of HMCN |
|---|
| Input: An new unseen task |
| Input: Trained basic critic and meta-critic networks |
| Input: Trained task specified network |
| Output: Trained actor network |
| Init: one actor network; |
| For $step = 1,...,m$ do |
| Sample few training data from task; |
| Train actor; |
| End |

which has 20 neutrons in hidden layers. The number of neurons in the input layer equals the dimensions of state plus the dimensions of action, and the number of neurons in the output layer is one. (ii) Actor is a three-layer MLP network, which has 80 neurons in hidden layers. The number of neurons in the input layer is one, and the number of neurons in the output layer is equal to the action size. (iii) Task specified network (TSN) is a one-layer LSTM with 30 neurons, and the number of neurons in the output layer is three. (iiii) Meta-critic network is a four-layer MLP network, which has 80 cell units in each hidden layers. The number of neurons in the input layer is equal to the number of output of basic network plus the number of output of TSN, and the number of output neurons equals one. The value of hyper-parameters is shown in Table 4.

For AC method, the authors use the same size of basic critic network and actor network as HMCN algorithm. For MC and MAML methods, the network architecture and hyper-parameter settings are same as the original paper, which can be found in literature [20] and [34] for more details. All the networks were built and trained in Pytorch [35].

### B. DEPENDANT MULTI-ARM BANDIT

In the Dependant Multi-arm Bandit (MAB) task, the authors assumed the rewards of arm are dependent: the probability of rewards obeys a Dirichlet distribution, which is unknown to the agent. Each task has a different arm reward configuration: multiple sample drawn from the same Dirichlet distribution. The authors prepared the experiments for 2-arm, 4-arm and 6-arm bandits.

For the rigor of comparisons, the authors generated 200 tasks with ample samples for training AC, MAML, MC and HMCN methods. Each task instance comprises 10 trials. In each experiment, the algorithms were trained for 2000 episodes. For HMCN method, in specific actor training stage, the authors trained actors for each new task, where the parameters of basic, meta critic and task specified network were fixed. In the test experiment, the authors generated 50 new tasks along with different rewards distribution and only a few trials were allowed per task. The authors independently carried out each experiment 10 times respectively to reduce the randomness of receiving a rewards. The learned policy was tested every 5 trials to calculate the average scores.

The experimental results are shown in Table 3, where the average reward is the point product of the softmax output of actor network and the probability of receiving a reward by pulling each arm. The proposed method shows better performance than other meta learning methods at any given test.

The experiment results in Table 3 have demonstrated that the validity of the proposed method. The authors attribute this outstanding performance to the design of network structure of the algorithm. By training multiple MAB tasks along with a shared high-level network architecture, which contains a basic/meta critic and task specified networks, the authors distill and persevere common knowledge from multiple source tasks in the form of network parameters. In specific actor training stage, with previously saved common knowledge, meta critic network can criticize and learn an actor more efficiently than other learning algorithms while solving a new MAB task.

### C. CARTPOLE CONTROL

Cartpole control is a classic control task in reinforcement learning research, which was first studied in literature by Donaldson [36]. In the control system, there is an inverted pendulum mounted on a pivot point on a cart. The goal of control system is to keep the pendulum upright by apply horizontal forces to the cart. The observation $s$ consists of four elements: the cart position $x$, the velocity of cart $x_V$, the pole angle $\theta$, the velocity of pole angle $\theta_V$. The action $a$ is a horizontal force of $+1$ or $-1$ applied to the cart. The reward of $+1$ will be provided for every timestep when the pendulum remains upright. The threshold of every task is set $\Delta = 200$, which means if the pole remains upright for 200 continuous timesteps or falls, the episode will automatically end up.

For the rigor of comparisons, the authors sampled pole lengths in the range [1, 8] to generate 500 tasks for training AC, MAML, MC and HMCN methods. Each task instance comprises 10 trials. In each experiment, agents were trained

**TABLE 3.** Comparison of several meta learning methods in MAB task.

| Num of Bandits | 2-arm | | | 4-arm | | | 6-arm | | |
|---|---|---|---|---|---|---|---|---|---|
| Num of Pulls | 5 | 10 | 15 | 10 | 15 | 20 | 15 | 20 | 25 |
| Random | 0.5 | | | 0.25 | | | 0.17 | | |
| AC | 0.60 (0.18) | 0.60 (0.18) | 0.60 (0.18) | 0.35 (0.13) | 0.36 (0.13) | 0.36 (0.12) | 0.20 (0.18) | 0.21 (0.18) | 0.21 (0.17) |
| MAML | 0.63 (0.26) | 0.65 (0.26) | 0.65 (0.25) | 0.37 (0.20) | 0.37 (0.19) | 0.38 (0.22) | 0.26 (0.15) | 0.27 (0.15) | 0.27 (0.14) |
| MC | 0.70 (0.19) | 0.70 (0.19) | 0.70 (0.18) | 0.38 (0.14) | 0.39 (0.16) | 0.39 (0.15) | 0.26 (0.10) | 0.27 (0.10) | 0.28 (0.10) |
| HMCN | **0.71 (0.19)** | **0.72 (0.18)** | **0.72 (0.18)** | **0.39 (0.13)** | **0.40 (0.13)** | **0.41 (0.12)** | **0.28 (0.10)** | **0.29 (0.10)** | **0.30 (0.11)** |

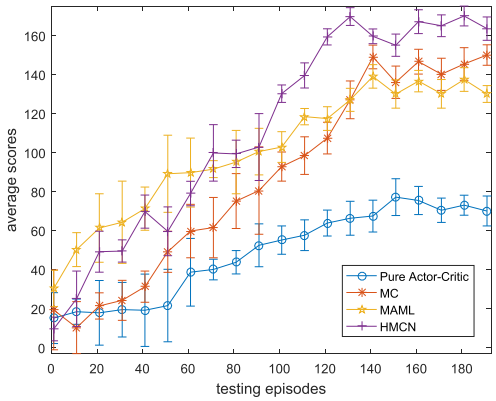



**FIGURE 4.** Average scores and standard deviations on different pole length tasks.

**TABLE 4.** Value of hyper parameters.

| Description | Value |
|---|---|
| Basic critic base learning rate | 0.0001 |
| Task specified network base learning rate | 0.0001 |
| Meta critic base learning rate | 0.0001 |
| Actors base learning rate (meta training stage) | 0.0001 |
| Actors base learning rate (specific actor training stage) | 0.001 |
| Initial exploration parameter | 1 |
| Minimum exploration parameter | 0.1 |
| Discount factor | 0.99 |

**TABLE 5.** The average score and standard deviation on different pole length tasks.

| Task<br>AVG (STD) | Random | Pure Actor-Critic | MAML | MC | HMCN |
|---|---|---|---|---|---|
| Cartpole with different pole length | 9.6 | 71.1 (4.6) | 132.5 (4.2) | 145.2 (4.9) | 166.2 (3.4) |

for 5000 episodes, which consists of around 100000 update iterations. For HMCN method, in specific actor training stage, the authors trained actors for each new task, where the parameters of basic, meta critic and task specified network were fixed. The four algorithms were tested on 100 new tasks, which were repeated 10 times for each task. The learned policy was tested every 10 training episodes to calculate the average scores of multiple runs. The score refers to the cumulative reward of agents in each episode.

The average scores and standard deviations in the Figure 4 show that the HMCN learns faster than MC and pure actor-critic in the early stage of specific actor training stage, but learns slower than MAML starting from a well initialized network. Along with the training of actor network, HMCN quickly learns to outperform the others and achieves the highest scores in end of specific actor training stage.

Table 5 records the average scores and standard deviations after the convergence of the four algorithms on the different pole length of Cartpole tasks, and quantitatively analyzes the experimental results. Compared with the pure actor-critic, MAML, and meta critic algorithms, HMCN improves the scores by 57.2%, 20.4% and 12.6%, In terms of stability, the standard deviation is reduced by 26.1%, 19.0% and 30.6%.

To verify robustness of the proposed algorithm, the authors not only focus on different pole lengths, but also assume different pole weight to find whether the proposed algorithm still works. For AC, MAML, MC and HMCN methods, the authors generated 500 tasks by sampling pole weight in the range [1, 3], and each task instance comprises 10 trials. In each experiment, agents were trained for 5000 episodes, which consists of around 100000 update iterations. The four algorithms were tested on 100 new tasks, which were repeated 10 times for each task. The learned policy was tested every 5 training episodes to calculate the average scores of multiple runs. The score refers to the cumulative reward of agents in each episode.

The Figure 5 shows that average scores and standard deviations of four learning algorithms on different pole weight tasks. The authors observe that MAML has a clear lead in scoring at the beginning of specific actor training stage, but is gradually overtaken by HMCN and MC in the later stages of testing. HMCN achieves the best performance in the end since basic/meta critic and task specified network start helping train actors and accelerate the learning process.

Table 6 records the average scores and standard deviations after the convergence of the four algorithms on the different pole weight of Cartpole tasks, and quantitatively analyzes the experimental results. Compared with the pure actor-critic, MAML and meta critic algorithms, HMCN improves the scores by 49.4%, 20.9% and 11.9 %, the standard deviation is reduced by 32.4%, 13.8% and 24.2%.

In contrast to basic reinforcement learning and state-of-the-art meta learning algorithms, the proposed method significantly increases task scores, besides, the stability also has been improved greatly.

To understand why hierarchical meta-critic framework works well while dealing with a new cartpole task, consider

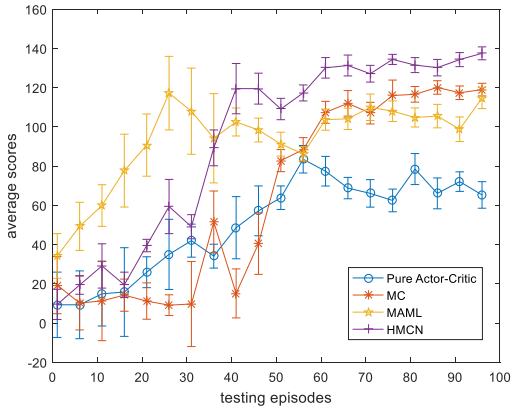



**FIGURE 5.** Average scores and standard deviations on different pole weight tasks.

**TABLE 6.** The average score and standard deviation on different pole weight tasks.

| Task AVG (STD) | Random | Pure Actor-Critic | MAML | MC | HMCN |
|---|---|---|---|---|---|
| Cartpole with different pole weight | 9.2 | 67.8 (3.7) | 106.3 (2.9) | 118.8 (3.3) | 134.0 (2.5) |

**TABLE 7.** The hypothesis test of Cartpole.

| (H,SIGNIFICANCE) | Alternative hypotheses | | |
|---|---|---|---|
| | $S_{AC} < S_{HMCN}$ | $S_{MAML} < S_{HMCN}$ | $S_{MC} < S_{HMCN}$ |
| Cartpole with different pole length | (1, 1.02e-11) | (1, 9.78e-07) | (1, 1.25e-04) |
| Cartpole with different pole weight | (1, 6.07e-10) | (1, 2.24e-06) | (1, 4.75e-05) |

that during specific actor training stage meta-critic network is able to criticize accurately a new cartpole task based on the information given by basic critic and task specified network, then from the point of view the new task's actor, it credits to the meta-critic network's pre-training, which in fact preserves common knowledge between multiple different cartpole tasks, i.e. meta-knowledge, in the form of network parameters during meta training stage. When HMCN method solves a new cartpole task, basic critic network tells meta critic network which state the current task is in, and task specified network helps meta critic network decide which type of current task is. Combining the knowledge of basic critic network and task specified network, meta critic network can criticize an actor accurately and learn a new cartpole task more efficiently than other learning algorithms.

Furthermore, the authors performed several t-tests for four algorithms on two kinds of Cartpole tasks. The significance level was set as $\alpha = 5\%$. As shown in Table 7, the score of pure actor critic is $S_{AC}$, the score of MAML is $S_{MAML}$, the score of MC is $S_{MC}$, the score of HMCN is $S_{HMCN}$. The test of HMCN all rejects original hypothesis and accepts alternative hypothesis. It further shows that compared to pure reinforcement learning and meta learning algorithms, HMCN method leads to statistically-significant improvement in the performance while solving a new task.

The reason why the proposed method addresses sample inefficient in DRL is as follows: The general reinforcement learning method solves a new problem by training the network from scratch, while the proposed method is able to learn a new task with only a few trials. As a result, when faced with a distribution of different tasks, the general reinforcement learning method has to learn tasks from scratch one by one, in the contrast, the proposed method relies on the already distilled meta knowledge and only needs to train one actor with very few trials to achieve better performance.

The proposed method draws on the idea of knowledge distillation [37], which supervises a student network by building a teacher network. In the proposed approach, meta-critic network acts as the role of a teacher network, basic critic and task specified networks provide the teacher network with additional information about the current task. Different from previous knowledge distillation methods, the authors design multiple actors as student networks, where one actor corresponds to one task. The teacher network distills and preserves meta knowledge by supervising multiple actors trying to solve any specified task. The proposed HMCN approach provides a route to knowledge transfer that can efficiently adapt to a new task and learn better.

### D. EFFECT OF DEPTH

The proofs in Section B and C has proved the validity of the proposed meta learning method in several classic control tasks. Meta knowledge can be distilled from different but related tasks for accelerating learning speed and improve the performance. Now, the authors seek to experimentally explore this new method and aim to answer the question: is there a scenario for which hierarchical meta-critic networks requires a deeper representation for meta critic or actor networks to achieve better performance?

To answer this question, the authors will study a series of mountain car tasks with different heights of goals, where the tangential forces are applied to forcing the car to reach the target height on the right. The state of mountain car contains the horizontal position $x$ and the horizontal velocity $x$ of the car. Actions are set as the tangential forces of $-1$ or 1 applied on the car. The reward $r(s, a) = -1$ is provided for every timestep. When the car reaches a target height or the

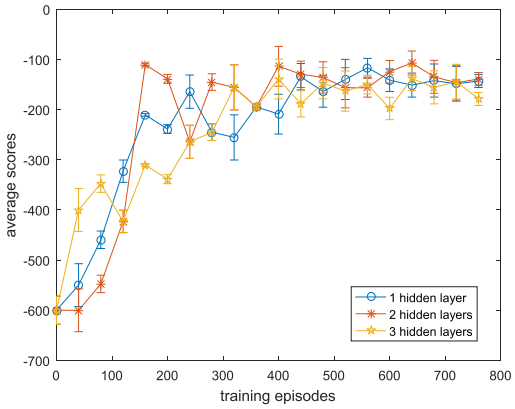



**FIGURE 6.** Different depth of actor networks.

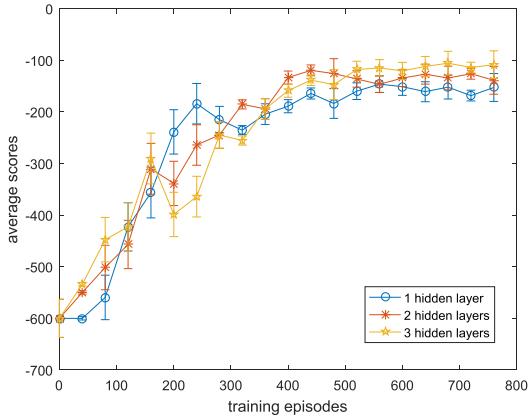



**FIGURE 7.** Different depth of meta-critic networks.

horizon of $H$ exceeds 600, the current episode automatically terminates.

To compare the relationship between depth of meta critic/actor networks and performance, the authors will vary the meta critic/actor networks depth from 1 to 3 hidden layers and compare the performance of those models, which have the same number of neurons and activation functions in each hidden layers. The height of goal is sampled in the range [0.2, 1.0] to generate 1000 tasks, and each task instance comprises 10 trials. In each experiment, agents were trained for 10000 episodes, which consists of around 6000000 update iterations. The authors tested a new mountain car task along with the target height of 0.5 and only a few trials were allowed for testing. The authors independently carried out each experiment 10 times respectively to reduce the randomness of receiving a rewards. The learned policy was tested every 50 training episodes to calculate the average scores of multiple runs. The score refers to the cumulative reward of agents in each episode.

The results, shown in Figure 6, demonstrate that deeper hidden layers of actor networks don't seem to bring the improvement of performance. In the contrast, Figure 7 shows that the proposed HMCN method indeed achieves better performance with a deeper meta critic network. This statistical results suggest that the depth of meta-critic network is pivotal for effective meta reinforcement learning using HMCN.

The reason why a deeper meta-critic network can achieve better performance is that a deeper meta-critic network might have a stronger ability to persevere and express meta knowledge, which can help criticize the actor efficiently with only a few trials. In the contrast, since the actor network is training from scratch every time, the depth of the actor network has little effect on the performance under the condition of very few trials.

## V. CONCLUSIONS AND FUTURE WORK

In this paper, the authors present a novel meta learning method to provide a route for knowledge transfer. The authors combine hierarchical framework into actor-critic method for distilling meta knowledge above distributions of different but related tasks. An end-to-end training approach is also designed for the proposed method. Empirical evaluation results show that the proposed method achieves better or comparable results.

The proposed method draws inspiration from the research of both meta learning and hierarchical reinforcement learning, which can be classified as meta reinforcement learning [38]. One of future work is to extend the proposed approach to other deep reinforcement learning algorithms, even supervised learning methods. Moreover, how to combine gradient signal into the proposed approach is also a promising research direction.

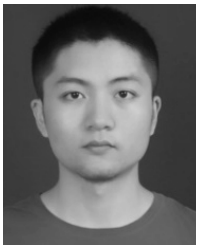
## ACKNOWLEDGMENT

The authors declare that there is no conflict of interest regarding the publication of this article.

## REFERENCES

[1] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[2] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[3] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behav. Brain Sci.*, vol. 40, no. 8, pp. 1–15, 2017.
[4] J. X. Wang *et al.* (2016). "Learning to reinforcement learn." [Online]. Available: https://arxiv.org/abs/1611.05763
[5] J. Schmidhuber, J. Zhao, and M. Wiering, "Simple principles of metalearning," *Tech. Rep. IDSIA*, vol. 69, no. 16, pp. 1–23, 1996.
[6] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to Learn*. New York, NY, USA: Springer, 1998, pp. 3–17.
[7] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.
[8] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 181–211, 1999.
[9] C. Daniel, H. Van Hoof, J. Peters, and G. Neumann, "Probabilistic inference for determining options in reinforcement learning," *Mach. Learn.*, vol. 104, nos. 2–3, pp. 337–357, 2016.
[10] A. Vezhnevets *et al.*, "Strategic attentive writer for learning macro-actions," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3486–3494.
[11] A. S. Vezhnevets *et al.*, "Feudal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 3540–3549.

[12] C. Florensa, Y. Duan, and P. Abbeel. (2017). "Stochastic neural networks for hierarchical reinforcement learning." [Online]. Available: https://arxiv.org/abs/1704.03012
[13] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proc. AAAI 31st Conf. Artif. Intell.*, 2017, pp. 456–489.
[14] A. Levy, G. Konidaris, R. Platt, and K. Saenko. (2017). "Learning multi-level hierarchies with hindsight." [Online]. Available: https://arxiv.org/abs/1712.00948
[15] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018.
[16] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*. Burlington, MA, USA: Morgan Kaufmann, 1992, pp. 271–278.
[17] B. Ghazanfari and M. E. Taylor. (2017). "Autonomous extracting a hierarchical structure of tasks in reinforcement learning and multi-task reinforcement learning." [Online]. Available: https://arxiv.org/abs/1709.04579
[18] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3675–3683.
[19] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.
[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1126–1135.
[21] M. Al-Shedivat, T. Bansal, Y. Burda, I. Sutskever, I. Mordatch, and P. Abbeel. (2017). "Continuous adaptation via meta-learning in nonstationary and competitive environments." [Online]. Available: https://arxiv.org/abs/1710.03641
[22] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 2402–2413.
[23] R. Adamski, T. Grel, M. Klimek, and H. Michalewski, "Atari games and intel processors," in *Proc. Workshop Comput. Games*. Cham, Switzerland: Springer, 2017, pp. 1–18.
[24] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1842–1850.
[25] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. (2016). "$RL^2$: Fast reinforcement learning via slow reinforcement learning." [Online]. Available: https://arxiv.org/abs/1611.02779
[26] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel. (2017). "A simple neural attentive meta-learner." [Online]. Available: https://arxiv.org/abs/1707.03141
[27] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3981–3989.
[28] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 458–589.
[29] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 523–531.
[30] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 2113–2122.
[31] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 2568–2598.
[33] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1589–1599.
[34] F. Sung, L. Zhang, T. Xiang, T. Hospedales, and Y. Yang. (2017). "Learning to learn: Meta-critic networks for sample efficient learning." [Online]. Available: https://arxiv.org/abs/1706.09529
[35] N. Ketkar, "Introduction to pytorch," in *Deep Learning With Python*. Berkeley, CA, USA: Apress, 2017, pp. 195–208.
[36] P. E. K. Donaldson, "Error decorrelation: A technique for matching a class of functions," in *Proc. 3rd Int. Conf. Med. Electron.*, 1960, pp. 173–178.
[37] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Comput. Sci.*, vol. 14, no. 7, pp. 38–39, 2015.
[38] J. X. Wang *et al.*, "Prefrontal cortex as a meta-reinforcement learning system," *Nature Neurosci.*, vol. 21, no. 6, pp. 123–156, 2018.



**ZHIXIONG XU** received the B.E. and M.S. degrees from the PLA University of Science and Technology, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with Army Engineering University. His research interests include machine learning and intelligent decision making.

**LEI CAO** received the B.S. degree from the China University of Science and Technology, in 1987, and the M.S. degree from the PLA University of Science and Technology, in 1990. He is currently a Professor with the Army Engineering University. His research interests include machine learning, command information systems, and intelligent decision making.

**XILIANG CHEN** received the B.S., M.S., and Ph.D. degrees from the PLA University of Science and Technology, in 2007, 2009, and 2019, respectively. He is currently an Associate Professor with the Army Engineering University. His research interests include reinforcement learning and intelligent decision making.