Letter

Competitive Meta-Learning

Boxi Weng, Jian Sun, Gao Huang, Fang Deng,
Gang Wang, and Jie Chen, *Fellow, IEEE*

Dear Editor,

This letter is concerned with developing meta-learning models for fast, stable, and effective few-shot learning across tasks over a few training samples. Nowadays, deep and reinforcement learning (RL) is widely used in autonomous intelligent systems (e.g., target recognition [1], path planning [2], and robot control [3], [4]). However, it is still challenging to learn quickly with a small number of data to obtain a workable model for a new task, also known as cross-task few-shot learning. Humans only need a few samples to learn new concepts or perform new tasks. In contrast, an autonomous system equipped with the most advanced AI technologies can hardly match the human rate of learning, and they generally call for significantly more data to reach a similar level of performance [5]. Such a gap can be bridged by e.g., effective transfer of learning experiences across tasks and rapid formation of skills from few samples, which are key ways to breaking the unmanned systems technology bottleneck.

Fortunately, meta-learning methods have been shown capable of effectively addressing cross-task few-shot learning. The main idea of meta-learning is to distill knowledge and experiences from historical tasks to guide the learning process in new tasks, or to equip itself with the ability of "learning to learn" through the past learning experience. The objective of meta-learning is to enhance the generalization performance of meta-learners at the task level. This is in contrast to transfer learning, which only emphasizes the ability to transfer from one task to another, and to pre-training, which concentrates on the model's immediate performance in various tasks without considering their adaptivity to subsequent tasks. A variety of meta-learning methods have been proposed, which have focused on different elements of "learning", including initial parameters, network structure, and optimization methods, to name just a few [6], [7].

Model-agnostic meta-learning (MAML) [6] is one of the most effective and widely used meta-learning methods thus far, which aims to find optimal initial model parameters based on gradient descent. This approach divides the update process into two stages: the task-specific inner-loop update and the cross-task outer-loop meta-update. Originally in [6], MAML is demonstrated via a first-order approximation, sacrificing the second-order derivatives for computational benefits. On this basis, Reptile [8] not only discards the second-order information but goes a step further by moving the initialized parameters of the meta-learner toward the temporary parameters trained on tasks, thus improving the computational efficiency. Meta-SGD [9], on the other hand, considers learning both the initial parameters, the update direction, and the learning rate, which endows the algorithm with strong ability for fast adaptation.

While MAML is now being widely used, it is not without flaws. It has been reported in [10] that the MAML training is sensitive to initial parameters, which incurs a high level of uncertainty and hinders its generalization. Further, MAML suffers from instability training issues too [11], resulting in sluggish convergence [12]. To address these challenges, we develop a competitive meta-learning (CML) method by considering the use of multiple meta-learners and inducing group (or collective) intelligence via a competition mechanism. CML incorporates information interaction into the parameter updating process of each meta-learner, thus enabling an effective exchange of the "best" learning experiences among meta-learners, and stimulating and accelerating each individual's learning process. This competition mechanism consists of a learning-adaptive aggregation of cross-task data information in the meta-learning process.

Problem formulation: Before formalizing the CML method, we first state the meta-learning setup, by describing a meta-learner in standard MAML as a running case followed by a meta-learner for RL later. Formally, let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ denote the inputs and labels, respectively, and define a model f_θ (e.g., a neural network) which is uniquely characterized by parameters θ to define the mapping $\mathcal{X} \to \mathcal{Y}$. MAML aims to find optimal initial parameters θ^* that achieve desired results on a new task with only a few updates. By measuring the performance of the model f_θ in terms of some loss ℓ, the MAML's training objective is to minimize the loss function

$$\min_{\theta} \sum_{\mathcal{T}_j \sim p(\mathcal{T})} \ell_{\mathcal{T}_j}(\theta) \tag{1}$$

where $\mathcal{T}_j$ represents the j-th task drawn from some distribution $p(\mathcal{T})$. Specifically, the parameter update in MAML consists of two stages: an inner-loop update and an outer-loop update. The data for cross-task learning can be split into meta-training set $\mathcal{D}$ and meta-testing set $\mathcal{D}'$. Each sampled task data $\mathcal{D}_{\mathcal{T}_j}$ consists of a support set $\mathcal{D}_{\mathcal{T}_j}^{\text{train}}$ and a query set $\mathcal{D}_{\mathcal{T}_j}^{\text{test}}$. Furthermore, the support set $\mathcal{D}_{\mathcal{T}_j}^{\text{train}}$ is used for training the model to adapt to a new task in the inner loop. Taking the one-step gradient update as an example, the inner-loop update in task $\mathcal{T}_j$ is obtained as follows:

$$\theta_j \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(\theta) \tag{2}$$

where θ_j is the adapted parameters in the j-th sampled task, θ is the initial parameters or the original parameters, α is the adaptation learning rate, and $\mathcal{L}_{\mathcal{T}_j}(\theta)$ denotes the loss function of θ in the j-th task. Based on the adapted parameters, the meta-gradient in task $\mathcal{T}_j$ can be found using the corresponding query set $\mathcal{D}_{\mathcal{T}_j}^{\text{test}}$ as follows:

$$g_{\mathcal{T}_j}^{\text{meta}} = \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(\theta_j). \tag{3}$$

After obtaining the meta-gradients corresponding to a batch of tasks, the parameters θ are updated in the outer loop as follows:

$$\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_j \sim p(\mathcal{T})} g_{\mathcal{T}_j}^{\text{meta}} \tag{4}$$

where β is the meta step size. In short, a learning epoch consists of an inner-loop update and an outer-loop meta-update. The meta-learner continuously samples tasks $\mathcal{D}_{\mathcal{T}_j}$ from the meta-training set $\mathcal{D}$, and updates its parameters until reaching the optimal initial parameters θ^*. Concretely, the meta-learning objective is rewritten as follows:

Corresponding author: Gang Wang.

Citation: B. X. Weng, J. Sun, G. Huang, F. Deng, G. Wang, and J. Chen, "Competitive meta-learning," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 9, pp. 1902–1904, Sept. 2023.

B. X. Weng, J. Sun, F. Deng, and G. Wang are with the National Key Laboratory of Autonomous Intelligent Unmanned Systems, Beijing Institute of Technology, Beijing 100081, and also with the Beijing Institute of Technology Chongqing Innovation Center, Chongqing 401120, China (e-mail: wengboxi@bit.edu.cn; sunjian@bit.edu.cn; dengfang@bit.edu.cn; gang wang@bit.edu.cn).

G. Huang is with the School of Automation, Tsinghua University, Beijing 100084, China (e-mail: gaohuang@tsinghua.edu.cn).

J. Chen is with Department of Control Science and Engineering, Tongji University, Shanghai 201804, and also with the National Key Laboratory of Autonomous Intelligent Unmanned Systems, Beijing Institute of Technology, Beijing 100081, China (e-mail: chenjie@bit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2023.123354

$$\min_{\theta} \sum_{\mathcal{T}_j \sim p(\mathcal{T})} \ell_{\mathcal{T}_j}(\theta) = \sum_{\mathcal{T}_j \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_j}(\theta)). \tag{5}$$

For RL, each task $\mathcal{T}_j$ contains a tuple $(\mathcal{S}, \mathcal{A}, \lambda, r, p_0, P)$, where $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, λ the discount factor, $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, p_0 the initial state distribution, and $P(s_{t+1}|s_t, a_t)$ the transition probability distribution from state s_t to s_{t+1} after taking action a_t. Hence, we model each task as a Markov decision process (MDP). A neural network policy $f_\theta: \mathcal{S} \to \mathcal{A}$ is a mapping from a state to an action. The loss $\mathcal{L}_{\mathcal{T}_j}(\theta)$ for task $\mathcal{T}_j$ corresponds to the negative expected reward

$$\mathcal{L}_{\mathcal{T}_j}(\theta) = -\mathbb{E}_{s_t, a_t \sim f_\theta, p_{0\mathcal{T}_j}, P_{\mathcal{T}_j}} \Big[\sum_{t=0}^{H-1} \lambda^t r_j(s_t, a_t) \Big] \tag{6}$$

where H is the horizon. Specifically, for a sampled task $\mathcal{T}_j$, we generate several MDP trajectories using policy f_θ to form the support set $\mathcal{D}_{\mathcal{T}_j}^{\text{train}}$. Next, the policy is updated to an adapted model f_{θ_j} by means of (2). We sample new trajectories to construct the query set $\mathcal{D}_{\mathcal{T}_j}^{\text{test}}$ according to f_{θ_j}. After obtaining the support and query sets, we can train the RL meta-learner as in the supervised learning setting.

Competitive meta-learning: Inspired by swarm intelligence or more specifically competitive swarm optimization [13], [14], we deploy multiple meta-learners and introduce a novel competition mechanism for their meta-training.

In CML, we refer to the set of meta-learners as a meta-learning swarm, and consider each meta-learner as a standard meta-learner in MAML. Suppose there are m meta-learners in a swarm, and the parameters of those meta-learners are randomly initialized as $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(m)}$. The training tasks in $\mathcal{D}$ are randomly reshuffled to generate the different meta-training sets $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots, \mathcal{D}^{(m)}$, which are assigned to different meta-learners, respectively. It is important to mention that the meta-training data for each meta-learner is of the same size, that is, $|\mathcal{D}| = |\mathcal{D}^{(1)}| = \cdots = |\mathcal{D}^{(m)}|$.

In our competitive meta-training, a competition step is added to the standard meta-update at every meta-learning epoch. First, each meta-learner in the meta-learning swarm performs a meta-update independently. For the i-th meta-learner, the original parameter $\theta^{(i)}$ is updated using a batch of tasks data sampled from the corresponding meta-training set $\mathcal{D}^{(i)}$ as follows:

$$\theta^{(i)} \leftarrow \theta^{(i)} - \beta \nabla_{\theta^{(i)}} \sum_{\mathcal{T}_j \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(\theta_j^{(i)}). \tag{7}$$

After each learner finishes a meta-update, the performances of all meta-learners are compared using a preset metric (e.g., testing accuracy, training accuracy or loss, etc.). In this epoch, the best meta-learner is labeled as winner, while the rest in the swarm are all labeled as losers. We believe that a winner has a better learning effect in this meta-training epoch, which motivates us to leave the parameters θ_w unchanged as a reward for winning this round of competition. Since the loser loses this competition, it should learn from the winner by updating θ_l toward θ_w. The i-th meta-learner in a round of competition updates its parameters as follows:

$$\text{Winner}: \ \theta_w^{(i)} \leftarrow \theta_w^{(i)} \tag{8}$$

$$\text{Loser}: \ \theta_l^{(i)} \leftarrow \theta_l^{(i)} + \gamma(\theta_w^{(i)} - \theta_l^{(i)}) \tag{9}$$

where γ is the competitive learning rate. Exact implementation of MAML requires the second-order information of the loss function $\mathcal{L}(\theta)$, so the computational complexity of MAML per iteration is $O(d^2)$, where d is the problem dimension. Therefore, the computational complexity of CML is $O(md^2)$. In general, we have $d >> m$, where m is the number of meta learners in CML, and oftentimes a constant number of swarm learners yields an affordable increase in computational complexity relative to MAML. A pictorial description of our proposed CML method is given in Fig. 1, with the algorithm tabulated in Algorithm 1.

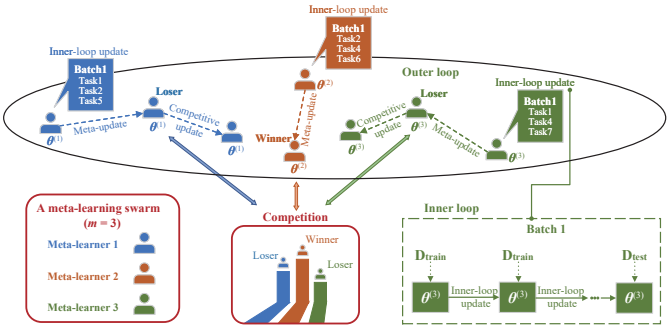

Fig. 1. A pictorial description of CML.

Algorithm 1 Competitive Meta-Learning (CML)

Require: Cross-task dataset $\mathcal{D}$
Require: Step size hyperparameters α, β, γ
Randomly initialize $\theta^{(1)}$, $\theta^{(2)}, \ldots, \theta^{(m)}$
Randomly shuffle tasks in $\mathcal{D}$ to generate $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \ldots, \mathcal{D}^{(m)}$
while not done **do**
  for all $\theta^{(i)}$ **do**
    Sample batch of tasks $\mathcal{T}_j^{(i)} \sim p(\mathcal{T})$, each task data $\mathcal{D}_{\mathcal{T}_j}^{(i)}$ in $\mathcal{D}^{(i)}$ contains some $\mathcal{D}_{\mathcal{T}_j}^{\text{train}(i)}$ and $\mathcal{D}_{\mathcal{T}_j}^{\text{test}(i)}$
    for all $\mathcal{T}_j^{(i)}$ **do** Evaluate $\nabla_{\theta^{(i)}} \mathcal{L}_{\mathcal{T}_j^{(i)}}(\theta^{(i)})$ using $\mathcal{D}_{\mathcal{T}_j}^{\text{train}(i)}$
      Compute adapted parameters by gradient descent via $\theta_j^{(i)} \leftarrow \theta^{(i)} - \alpha \nabla_{\theta^{(i)}} \mathcal{L}_{\mathcal{T}_j^{(i)}}(\theta^{(i)})$
      Evaluate $g_{\mathcal{T}_j}^{\text{meta}(i)} = \nabla_{\theta^{(i)}} \mathcal{L}_{\mathcal{T}_j^{(i)}}(\theta_j^{(i)})$ using $\mathcal{D}_{\mathcal{T}_j}^{\text{test}(i)}$
    end for
    Update parameters $\theta^{(i)} \leftarrow \theta^{(i)} - \beta \sum_{\mathcal{T}_j^{(i)} \sim p(\mathcal{T})} g_{\mathcal{T}_j}^{\text{meta}(i)}$
  end for
  Mark $\theta^{(i)}$ with highest preset competition metric as winner, and the rest as loser
  for all $\theta^{(i)}$ **do**
    if $\theta^{(i)}$ is winner **then**
      $\theta_w^{(i)} \leftarrow \theta_w^{(i)}$
    else
      $\theta_l^{(i)} \leftarrow \theta_l^{(i)} + \gamma(\theta_w^{(i)} - \theta_l^{(i)})$
end while
Return $\theta^{(i)}$

Experiment results: Since our CML can adopt general meta-learning algorithms as base learners to form a swarm, we evaluate the performance of MAML-based CML over diverse tasks, including classification and RL. To demonstrate the efficacy of the competition mechanism in CML, we consider a CML swarm of m learners against m independent MAML learners (referred to as a MAML swarm yet without any mutual communication and interaction). Specifically, we randomly generate initial parameters $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(m)}$, assigned to the m meta-learners in both CML and MAML swarms. We next utilize CML and standard MAML algorithms to train the two meta-learning swarms in the same environment, which comprises the meta-training task data $\mathcal{D}$, randomly initialized parameters θ, and hyper-parameters α, β. The only difference between the CML and MAML swarms is that the latter uses MAML to update the parameters directly, whereas the former performs a competition after each meta-update. Through substantial tests, we observe that CML with a competition mechanism can effectively boost the learning (e.g., stability, cross-task generalization) performance.

To examine the cross-task few-shot classification performance of CML, we call for the benchmark datasets MiniImagenet. Adopting the models in [6], we use a CNN architecture consisting of 4 modules with 3×3 convolutions, followed by batch normalization, ReLU nonlinearity, and 2×2 max-pooling. Our experiments focus on the N-way K-shot classification task; that is, we sample N classes from the meta-training set, train the model with K images from each of the N classes, then test the model by classifying new images from the N classes. Specifically, during training, we draw a batch of tasks for

one outer-loop update. Each task consists of N classes, with K images as the support data and 15 new images as query data. And we take the testing accuracy as the competition metric.

The MiniImagenet dataset [15] contains 60 000 color images from 100 categories. For MiniImagenet, we consider 5-way 1-shot classification tasks. We set $m = 3$ for both two methods, and the batch size is 4 for 5000 epochs. The results are shown in Fig. 2. Compared with MAML, CML exhibits significantly improved performance. It is worth mentioning that the testing accuracy of CML at epoch 1000 is already close to that of MAML at epoch 3000. This illustrates the efficiency of CML at the early stage, and can, to a certain extent, dispel the doubts about the low utilization of computational resources of the method.

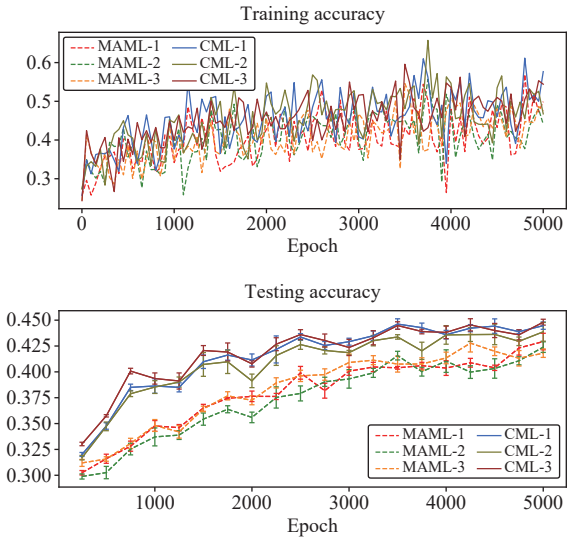

Fig. 2. Average accuracy on MiniImagenet. Error bars represent ±1 standard deviation.

In the RL experiment, we evaluate our CML method on the standardized continuous control task HalfCheetah-Dir. Our experimental environments are implemented in OpenAI Gym [16] with the MuJoCo [17] physics simulator. Following the models used in [6], we consider a neural network policy having two hidden layers both of size 100 along with the ReLU activation function. The only change made relative to [6] is that the meta-optimizer employed in our experiments is the proximal policy optimization (PPO) algorithm [18]. We set $m = 2$ for both two methods, and the batch size is 20. Furthermore, for all RL experiments, We take the training reward after inner-loop updates to be the competition metric.

Each task corresponds to moving forward or backward, consisting of a total of 2 tasks. The directions are drawn from a Bernoulli distribution with parameter 0.5 on {–1, 1}, where 1 and –1 represent the forward and backward directions, respectively. The cheetah aims to run in a particular direction in this experiment, and the reward is the magnitude of the velocity in the goal direction. As shown in Fig. 3, there is a meta-learner trained with MAML that shows very harsh learning performance. However, the meta-learners trained with the CML method yield (much) higher training and testing returns even from the same initialization. This experiment demonstrates that CML can effectively accelerate and enhance the stability of meta RL.

Conclusion: In this letter, we presented a novel method termed CML for cross-task few-shot learning. CML penetrates the idea of particle swarm optimization into meta-learning. By introducing a competition mechanism among a swarm of meta-learners, CML features improved stability, better generalization, and faster learning speed. Convincing experiments were performed to demonstrate its merits over both image classification as well as RL tasks.

Acknowledgments: The work was supported in part by the National Key R&D Program of China (2021YFB1714800), the National Natural Science Foundation of China (62173034, 61925303, 62025301, 62088101), and the CAAI-Huawei MindSpore Open Fund.

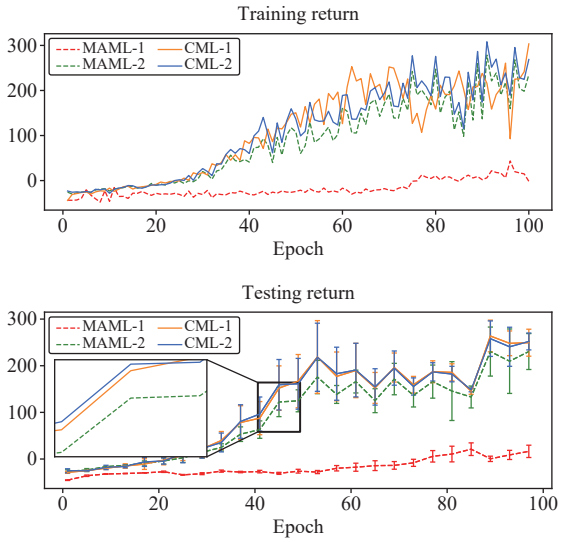

Fig. 3. Average return on HalfCheetah-Dir. Error bars represent ±1 standard deviation.

References

[1] J. Yin, D. Zhou, L. Zhang, J. Fang, C.-Z. Xu, J. Shen, and W. Wang, "ProposalContrast: Unsupervised pre-training for lidar-based 3D object detection," in *Proc. Eur. Conf. Comput. Vision*, 2022, pp. 17–33.

[2] M. Mazouchi, S. Nageshrao, and H. Modares, "Conflict-aware safe reinforcement learning: A meta-cognitive learning framework," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 3, pp. 466–481, 2021.

[3] J. Chen, J. Sun, and G. Wang, "From unmanned systems to autonomous intelligent systems," *Eng.*, vol. 12, pp. 16–19, 2022.

[4] Y. Li, X. Wang, J. Sun, G. Wang, and J. Chen, "Data-driven consensus control of fully distributed event-triggered multi-agent systems," *Sci. China Inf. Sci.*, vol. 66, no. 5, p. 152202, 2023.

[5] X. Wang, J. Sun, G. Wang, F. Allgöwer, and J. Chen, "Data-driven control of distributed event-triggered network systems," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 1, pp. 1–14, 2023.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[7] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.

[8] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," arXiv preprint arXiv: 1803.02999, 2018.

[9] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-SGD: Learning to learn quickly for few-shot learning," arXiv preprint arXiv: 1707.09835, 2017.

[10] A. Antoniou, A. Storkey, and H. Edwards, "How to train your MAML," in *Proc. Int. Conf. Learn. Rep.*, 2019, pp. 1–11.

[11] H. Liu, R. Socher, and C. Xiong, "Taming MAML: Efficient unbiased meta-reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4061–4071.

[12] H. S. Behl, A. G. Baydin, and P. H. Torr, "Alpha MAML: Adaptive model-agnostic meta-learning," arXiv preprint arXiv: 1905.07435, 2019.

[13] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, 2014.

[14] B. Xin, J. Chen, J. Zhang, H. Fang, and Z.-H. Peng, "Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: A review and taxonomy," *IEEE Trans. Syst. Man Cyb., Part C*, vol. 42, no. 5, pp. 744–767, 2011.

[15] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Rep.*, 2017, pp. 1–11.

[16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," arXiv preprint arXiv: 1606.01540, 2016.

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for modelbased control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv: 1707.06347, 2017.