PiCor: Multi-Task Deep Reinforcement Learning with Policy Correction

Fengshuo Bai[1, 2], **Hongming Zhang**[3], **Tianyang Tao**[4], **Zhiheng Wu**[1, 2], **Yanna Wang**[2], **Bo Xu**[2, 5*]

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]Institute of Automation, Chinese Academy of Sciences
[3]University of Alberta
[4]Université Paris-Saclay
[5]Nanjing Artificial Intelligence Research of IA
{baifengshuo2020, boxu}@ia.ac.cn

Abstract

Multi-task deep reinforcement learning (DRL) ambitiously aims to train a general agent that masters multiple tasks simultaneously. However, varying learning speeds of different tasks compounding with negative gradient interference makes policy learning inefficient. In this work, we propose PiCor, an efficient multi-task DRL framework that splits learning into policy optimization and policy correction phases. The policy optimization phase improves the policy by any DRL algothrim on the sampled single task without considering other tasks. The policy correction phase first constructs a performance constraint set with adaptive weight adjusting. Then the intermediate policy learned by the first phase is constrained to the set, which controls the negative interference and balances the learning speeds across tasks. Empirically, we demonstrate that PiCor outperforms previous methods and significantly improves sample efficiency on simulated robotic manipulation and continuous control tasks. We additionally show that adaptive weight adjusting can further improve data efficiency and performance.

Introduction

Reinforcement learning (RL) (Sutton and Barto 2018) combining with deep neural networks (LeCun, Bengio, and Hinton 2015) shows powerful capabilities on various single-task decision-making problems, including board games (Silver et al. 2017, 2018), video games (Mnih et al. 2015; Vinyals et al. 2019), robotic control (Levine et al. 2016; Rajeswaran et al. 2018; Kalashnikov et al. 2018), and some industrial applications (Mao et al. 2019; Tang et al. 2019). Despite the learning algorithm is general, the solution is not; each agent can only solve the one task it was trained on. To make the RL agent more general, many new methods focus on multi-task learning settings that solve multiple tasks at once. However, current approaches still suffer from sample inefficiency, this makes multi-task DRL difficult to train a general agent for practical applications.

One of the most straightforward methods is to continually train the policy on various tasks leveraging existing single-task learning algorithm. Unfortunately, neural networks are proven to suffer from inevitable catastrophic forgetting (Kirkpatrick et al. 2017). The training of subsequent tasks can make the knowledge of previously learned tasks abruptly lost. To mitigate catastrophic forgetting, alternate training on multiple tasks becomes a general way (Teh et al. 2017; Arora et al. 2018). However, alternate policy training means gradient interference among different tasks. Policy can easily get distracted by certain tasks due to varying learning speeds (Chen et al. 2018; Hessel et al. 2019), which causes the algorithm to focus on those salient tasks at the expense of generality. Some works provide solutions based on knowledge transfer and representation sharing (Teh et al. 2017; Xu et al. 2020; D'Eramo et al. 2020; Yang et al. 2020). However, these methods require expert-level policies or high-quality representations among all tasks, which are difficult and impractical.

To overcome the gradient interference and improve learning efficiency, we propose PiCor, an efficient framework for multi-task DRL. The critical insight is to alternately optimize the policy on sampled tasks while also correcting the negative interference on other tasks. Specifically, PiCor splits the learning into two phases, **P**olicy Opt**i**mization and Policy **Cor**rection. The policy optimization phase samples a task from the task distribution and performs policy learning with task-specific experience replay using any DRL algorithm. The policy correction phase first constructs a performance constraint set that approximates the performance improvement area considering all tasks. The constraint set is dynamically adjusted by *adaptive weight adjusting* to balance the learning speeds and guarantee a lower bound of average performance among tasks. The intermediate policy learned in the policy optimization phase is corrected by a probability metric (e.g, KL divergence) to keep it in the constraint set which controls the negative gradient interference. Our contribution is three-fold:

- We propose PiCor, a novel and efficient multi-task DRL algorithm framework, which mitigates gradient interference and balances learning speeds among different tasks.
- We enhance this framework with adaptive weight adjusting and task-specific experience replay, to further balance the learning progress among tasks. Both techniques are pivotal for the empirical success of PiCor.
- Experiments demonstrate that our method PiCor outperforms other state-of-the-art multi-task DRL methods

*These authors contributed equally.

and substantially improves sample efficiency on various robotic simulated manipulation tasks from Meta-world (Yu et al. 2020b) and continuous control tasks from the MuJoCo suite (Henderson et al. 2017).

Related Work

Deep reinforcement learning (DRL) becomes a powerful tool for solving sequential decision-making problems. There are many practical algorithms for continuous control tasks. Specifically, we choose Soft Actor-Critic (SAC) (Haarnoja et al. 2018) as the algorithm backbone in our work. However, our framework is general, and can be easily applied to other typical RL algorithms, such as TRPO (Schulman et al. 2015), PPO (Schulman et al. 2017) and TD3 (Fujimoto, van Hoof, and Meger 2018).

In recent years, many methods are proposed to solve multi-task learning problems. Previous work may boil down to three directions: knowledge transfer, representation sharing across tasks, and gradient interference mitigation.

For knowledge transfer-based methods, Arora et al. (2018) combines knowledge distillation with a single-task RL algorithm to learn the multi-task policy. Actor-Mimic (Parisotto, Ba, and Salakhutdinov 2016) and Distral (Teh et al. 2017) apply the policy distillation to reinforcement learning for learning the common behavior of different tasks into the shared policy network. KTM-DRL (Xu et al. 2020) employs offline knowledge transfer to train a policy network and leverages online learning to improve policy performance further. These methods mitigate gradient interference to some extent. However, they usually require expert-level policies, which are difficult and expensive to provide. In addition, teacher models have an important impact on the performance of multi-task policy, which restricts performance improvement.

Structure sharing methods focus on reusing network modules and sharing representation, which tries to find architectural solutions to the multi-task learning problem. Devin et al. (2017) takes advantage of a modular neural network to share information among tasks to solve the multi-task learning problem. D'Eramo et al. (2020) presents a shared network algorithm that can extract common representations across various tasks and efficiently generalize the knowledge for better performance. Router network (Rosenbaum, Klinger, and Riemer 2018) proposes a general architecture to jointly train the router and function blocks by employing a collaborative multi-agent reinforcement learning algorithm. Soft modularization (Yang et al. 2020) is a learning method that learns basic policy modules and automatically generates soft combinations probability via a routing network (Rosenbaum, Klinger, and Riemer 2018), which are trained in an end-to-end manner.

Several methods focus on overcoming gradient interference in multi-task DRL. Suteu and ke Guo (2020) proposes a method of correcting task gradients that can force gradients of each task to be nearly orthogonal to minimize the interference between tasks. Gradnorm (Chen et al. 2018) is a gradient normalization algorithm capable of dynamically tuning the magnitudes of the task gradients to balance learning speed during training. G-Surgery (Yu et al. 2020a) is a gradient correction method that can project the gradient of a task onto the normal plane of any other task to mitigate gradient interference among different tasks.

We summarize the differences between our method and previous work. Compared with knowledge transfer and representation sharing, our method does not depend on expert-level teacher models and automatically learns the shared knowledge among tasks. Furthermore, prior gradient manipulation-based methods optimize the policy with an implicit objective. In contrast, our method design a performance constraint set that directly focuses on the the performance maximization over all tasks.

Preliminaries

In this section, we recap the formalism of the Markov Decision Process (MDP) for RL, SAC algorithm, and policy performance measure.

Markov Decision Process (MDP)

A specific task can be modeled as a finite horizon Markov Decision Process (MDP), a 5-tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ represents the stochastic dynamics, which determines the probability of transferring to s' given state s and action a. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function and $\gamma \in (0, 1)$ is the discount factor. The policy $\pi(a|s)$ is a mapping from state space to action space.

Soft Actor-Critic (SAC)

SAC (Haarnoja et al. 2018) is an actor-critic method based on the maximum entropy RL framework, which optimizes policy in an off-policy way. This method encourages exploration by maximizing entropy, which accelerates learning. It optimizes parameters by alternating soft policy evaluation and soft policy improvement. For soft policy evaluation, the parameter θ of the soft Q-function is optimized by minimizing the soft bellman error:

$$\begin{aligned} \mathcal{L}(\theta) &= \mathop{\mathbb{E}}_{\tau_t \sim \mathcal{B}} \left[\left(Q_\theta(s_t, a_t) - r_t - \gamma \bar{V}(s_{t+1}) \right)^2 \right], \\ \bar{V}(s_t) &= \mathop{\mathbb{E}}_{a_t \sim \pi_\phi} \left[Q_{\bar{\theta}}(s_t, a_t) - \alpha \log \pi_\phi(a_t|s_t) \right], \end{aligned} \tag{1}$$

where $\tau_t = (s_t, a_t, r_t, s_{t+1})$ is the transition at time step t, $\mathcal{B}$ is a replay buffer, $\bar{\theta}$ is the parameter of target soft Q-function and α is temperature parameter which is learnable and controls the item of entropy. For soft policy improvement, the parameter ϕ of the policy is optimized by minimizing the following loss:

$$\mathcal{L}(\phi) = \mathop{\mathbb{E}}_{\substack{s_t \sim \mathcal{B} \\ a_t \sim \pi_\phi}} \left[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t) \right]. \tag{2}$$

SAC has good sample efficiency and excellent performance in solving continuous control tasks. Therefore, we use SAC as the backbone RL algorithm in this work.

Policy Performance Measure

In standard RL, we express the expected discounted cumulative reward as

$$J(\pi) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi} \\ a\sim\pi, s'\sim P}} \left[R(s,a,s')\right], \quad \text{with} \quad d^{\pi} = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi), \tag{3}$$

where the discounted future state distribution is denoted as d^{π} and P is the stochastic transition dynamics.

Kakade and Langford (2002) provide an explicit expression of the difference in performance between two policies π and π':

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}} \left[A^{\pi}(s,a)\right], \tag{4}$$

where A is the advantage function.

PiCor

In this section, we introduce PiCor, which includes two crucial phases for each update, the policy optimization phase and the policy correction phase. In addition, we introduce the objective for multi-task DRL and the practical implementation of our method.

Objective

In standard RL, the goal of the agent is to learn a policy that maximizes the expected return on a specific task. In multi-task DRL, a general agent is expected to learn multiple tasks simultaneously. The agent's goal is to maximize the average performance across all tasks, which can be formulated as

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{z\sim p(\mathcal{T})} \left[\mathop{\mathbb{E}}_{\tau\sim\pi} \left[R_z(\tau)\right] \right], \tag{5}$$

where $\mathcal{T}$ denotes a task random variable and $p(\mathcal{T})$ presents the task distribution.

Overview

PiCor is a general and efficient learning framework for multi-task DRL. Since SAC (Haarnoja et al. 2018) presents excellent sample efficiency and performance on continuous control tasks, we consider using SAC as the backbone for our method in this work. Each learning iteration in PiCor consists of two phases, the policy optimization phase and the policy correction phase. During the policy optimization phase, PiCor samples a task $z \sim p(\mathcal{T})$ and optimizes the policy to maximize the cumulative reward on this sampled task. A detailed description of the policy optimization phase can be found in subsection Policy Optimization Phase. However, the intermediate policy learned in the policy optimization phase does not consider the negative impact on other tasks in the optimization. Therefore, policy updates need to be modified to avoid harmful interference with other tasks. During the policy correction phase, the intermediate policy is constrained to avoid a significant drop in the average performance of the policy. In addition, when PiCor constructs the performance constraint set, it not only considers the mitigation of gradient interference among tasks but also leverages the adaptive weight adjusting technique to balance the learning progress of different tasks, which are described in subsection Policy Correction Phase and subsection Adaptive Weight Adjusting. Since the first phase can use any single-task DRL algorithm, the generality of our learning framework is guaranteed. At the same time, the second phase alleviates gradient interference and balances the learning progress between tasks to ensure the data efficiency of PiCor.

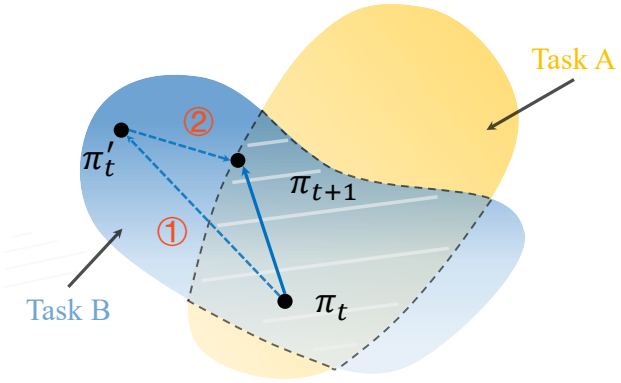


Figure 1: The illustration for two phases of policy learning in PiCor. ① denotes Policy Optimization Phase. The policy π_t is improved on the sampled task B. ② denotes Policy Correction Phase. The intermediate policy π_t' is constrained to the performance constraint set.

To illustrate the update process of PiCor, we take a simple case with two tasks for example. As shown in Figure 1, the yellow and blue regions denote the policy improvement area for task A and task B at time step t. The constraint set is constructed based on the overlapping area with adaptive weight adjusting. ① and ② denote the two phases in PiCor.

Policy Optimization Phase

The policy optimization phase updates policy without considering the harmful interference on other tasks. First, PiCor samples a task from the distribution $p(\mathcal{T})$. Second, it optimizes the policy to improve the performance on this sampled task. As for a specific task z, it optimizes parameters by alternately performing soft policy evaluation and soft policy improvement. For soft policy evaluation, the parameters θ of the soft Q-function are optimized by minimizing the soft Bellman residual:

$$J_Q(\theta; z) = \mathop{\mathbb{E}}_{\tau_t\sim\mathcal{B}_z} \left[\left(Q_\theta(s_t, a_t; z) - r_t - \gamma \bar{V}(s_{t+1})\right)^2 \right], \tag{6}$$

where $\tau_t = (s_t, a_t, r_t, s_{t+1})$ is a transition from the replay buffer $\mathcal{B}_z$ of task z.

After the updating of the soft Q-function, it performs the soft policy improvement step. The parameters ϕ of the policy

are optimized by minimizing the following loss:

$$J_\pi(\phi; z) = \mathop{\mathbb{E}}_{\substack{s_t \sim \mathcal{B}_z \\ a_t \sim \pi_\phi}} \left[\alpha_z \log \pi_\phi(a_t|s_t; z) - Q_\theta(s_t, a_t; z) \right], \tag{7}$$

where α_z is the temperature parameter of task z. The above optimization process improves the performance of policy π_t on task z. We use π'_t to denote an intermediate policy after the policy optimization phase at time step t.

Policy Correction Phase

The policy correction phase constrains the intermediate policy π'_t to alleviate the gradient interference and balance the learning process among all tasks. First, we construct a suitable constraint set that can represent the weighted performance bound across tasks. Inspired by (Kakade and Langford 2002), we utilize formula (4) to measure the performance changes when the policy is optimized. Specifically, policy optimization causes the average task performance (i.e., our objective) to change during the first phase. The performance difference between π_{t+1} and π_t is in proportion to the advantage, which is the average advantage under the discounted future state distribution $d^{\pi_{t+1}}$. However, it is difficult and inefficient to calculate it. Therefore, we use d^{π_t} to approximate $d^{\pi_{t+1}}$. In addition, to control the balance of various tasks, we set up different weights for distinct tasks adaptively, which is introduced in section . We incorporate this manipulation into the constraint set construction. Second, we project the intermediate policy to the constraint set by minimizing the Kullback-Leibler (KL) divergence between π'_t and π. This process can be formalized as the following optimization problem with performance constraints on N tasks sampled from $p(\mathcal{T})$:

$$\begin{aligned} &\pi_{t+1} = \arg\min_{\pi \in \Pi} D_{KL}(\pi'_t \parallel \pi), \\ &s.t. \sum_{z=1}^{N} w_z \mathop{\mathbb{E}}_{\substack{s \sim d^{\pi_t} \\ a \sim \pi}} \left[A^{\pi_t}(s, a) \right] \geq c(1-\gamma), \end{aligned} \tag{8}$$

where c is a negative number approximate to 0, w_z is the adaptive weight for task z which represents the importance of task z, γ is the discounted factor. This phase explicitly corrects the intermediate policy to mitigate interference among tasks, which guarantees sample efficiency.

Adaptive Weight Adjusting

Tasks with different difficulties lead to inconsistent convergence speeds in policy learning, which is detrimental to multi-task optimization. Therefore, we control the learning speeds during training among tasks by setting up different weights for each task. In the SAC algorithm, the temperature parameter α represents the importance of the entropy term. Large temperature parameter α encourages policy to explore the environment, making the policy more stochastic. To some extent, the temperature parameter α reflects the policy learning progress on the task. As for a specific task, when the α is large, the policy tends to explore, which means that the policy is not confident about the current optimal action. On the other hand, when the α is small, the policy is encouraged to exploit, which indicates that the policy is deterministic in the optimal action. Specifically, we provide the independent temperature parameter for different tasks, facilitating policy optimization more flexibly to balance exploration and exploitation across various tasks. Furthermore, we associate the temperature parameters of tasks with task weights. It is easy to achieve adaptive adjustment of task weights by leveraging the learnability of the temperature parameter. The adaptive weight for tasks can be calculated by

$$\boldsymbol{w} = \text{softmax}(\boldsymbol{\alpha}/\tau), \tag{9}$$

where τ is a hyperparameter to control the smoothness of the output distribution.

By adaptively adjusting the weights of each task, the policy learning process is more stable and policy can achieve higher performance, which is demonstrated in subsection Ablation Study.

Algorithm 1: PiCor

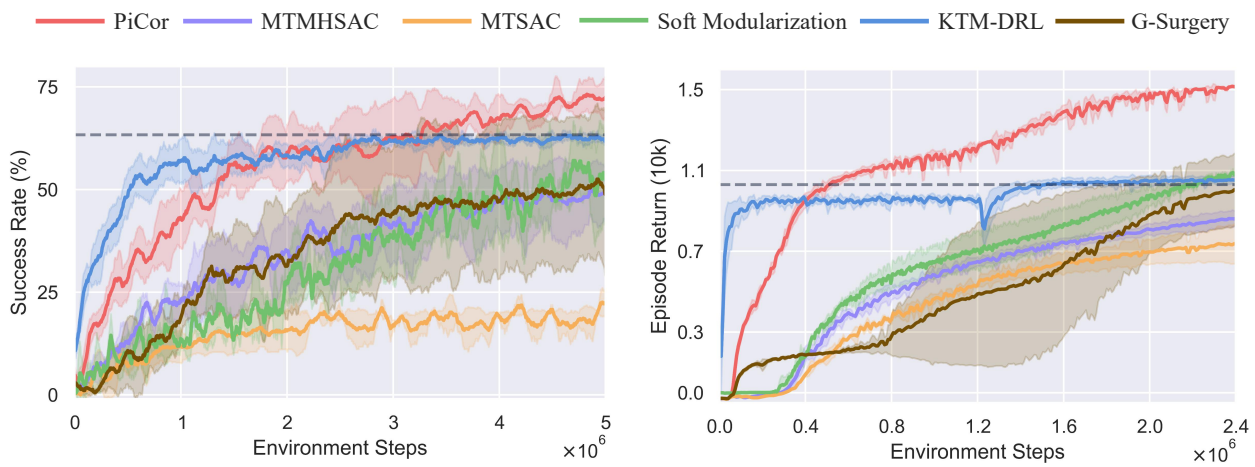
Input: the frequency K of sampling task
Input: the distribution of tasks $p(\mathcal{T})$
Parameter: the parameters θ for soft Q-function
Parameter: the parameters ϕ for policy
Output: a conditioned policy π_ϕ

```
Initialize replay buffer B_i ← ∅
for each iteration do
    if iteration % K == 0 then
        Sample a task z from p(T)
    end if
    Take action a_t ~ π_φ(a_t|s_t, z)
    Collect s_{t+1} ~ P(s_{t+1}|s_t, a_t; z)
    Store transitions B_z ← B_z ∪ {(s_t, a_t, r_t, s_{t+1})}
    for each gradient step do
        // Policy Optimization Phase
        Sample a mini-batch transition from B_z
        Optimize J_Q(θ; z) in (6) with respect to θ
        Optimize J_π(φ; z) in (7) with respect to φ
        // Policy Correction Phase
        Calculate the adaptive weight according to (9)
        Update φ by minimizing L(φ) in (10)
    end for
end for
```

Practical Implementation

As for the policy optimization phase, the intermediate policy π'_t can be easily calculated by alternating soft policy evaluation and soft policy improvement via framework of SAC. As for the policy correction phase, it can be solved efficiently if it is a convex programming problem with a reasonable approximation (Schulman et al. 2015; Achiam et al. 2017).

Although we can provide a formula that explicitly presents the update rule for the policy correction phase, optimizing a large neural network policy with many parameters is impractical. Therefore, we utilize another more straightforward and practical solution. We rewrite the above opti-



(a) Meta-world MT10 (b) HalfCheetah Task Group

Figure 2: Training curves of various methods on two benchmarks. The solid line presents the mean values, and the shaded area denotes the standard deviations over five runs. The grey dashed line indicates the average performance of individual policies across all tasks for each benchmark. Robotic simulated manipulation tasks from Meta-world MT10 are measured on success rate and continuous control tasks from HalfCheetah Task Group are measured on episode return.

Benchmark	PiCor	MTSAC	MTMHSAC	Soft Modularization	KTM-DRL	G-Surgery
Meta-world MT10	**73.3%**	23.3%	46.0%	46.6%	66.6%	63.3%
HalfCheetah Task Group	**15200**	7080	8170	10830	10750	10084

Table 1: Comparisons of all methods on average success rates for Meta-world MT10 and average episode returns for HalfCheetah Task Group. Meta-world (Yu et al. 2020b) reports the success rates of MTSAC and MTMHSAC for MT10 are 24% and 44%, respectively.

mization problem and give the following objective:

$$\mathcal{L}(\phi) = D_{KL}(\pi_\phi \parallel \pi'_t) + \frac{1}{2}\lambda\, \tilde{c}^2(\phi),$$

with

$$\tilde{c}(\phi) = \left(c(1-\gamma) - \sum_{z=1}^{N} w_z \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi_t} \\ a\sim\pi_\phi}} [A^{\pi_t}(s,a)] \right)^+, \tag{10}$$

where the parameter λ controls the size of the second item. For notation simplicity, we give a definition $(\cdot)^+ \equiv \max(\cdot, 0)$.

In addition, we use a task-specific replay buffer to store transitions that interact with the environment to improve data efficiency further. The complete procedure of PiCor is presented in Algorithm 1.

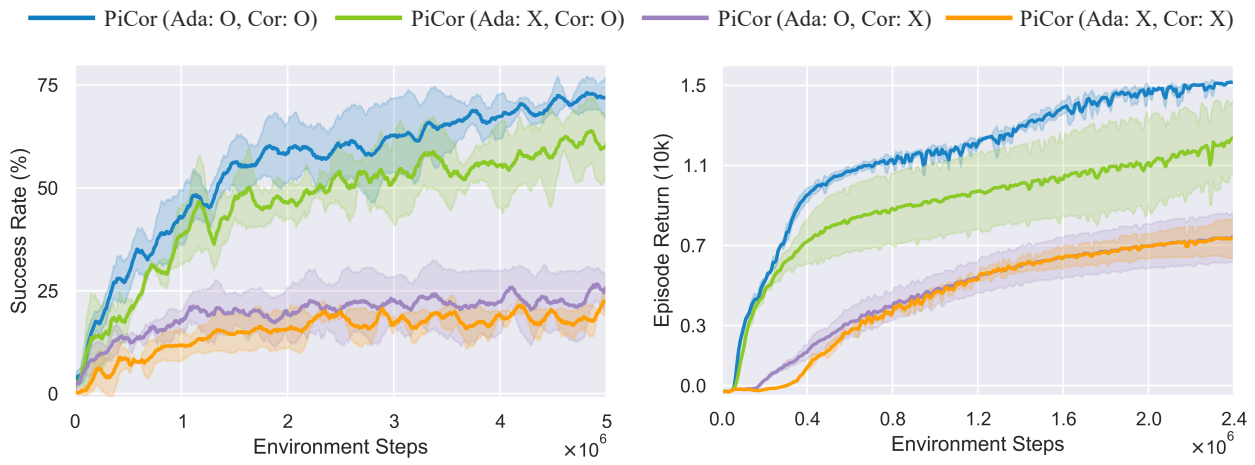
Experiments

In this section, we conduct experiments to compare PiCor with five existing approaches on various robotic simulated manipulation tasks from Meta-world (Yu et al. 2020b) and continuous control tasks from the MuJoCo suite (Henderson et al. 2017). More specifically, we use two multi-task benchmarks in our experiments. The first benchmark is MT10, which is challenging and requires the agent to learn ten manipulation tasks with fixed goals simultaneously. The second is the HalfCheetah Task Group which includes eight similar locomotion tasks.

Baseline Methods

We choose multi-task SAC, multi-head multi-task SAC, and three recent state-of-the-art multi-task DRL methods.

- Multi-task Soft Actor-Critic (MTSAC): MTSAC is an off-policy actor-critic algorithm, which adapts SAC algorithm (Haarnoja et al. 2018) to the multi-task setting.
- Multi-task Multi-headed Soft Actor-Critic (MTMHSAC): MTMHSAC is a variant of MTSAC, which emphasizes the importance of shared representation. It uses a shared network as a backbone but independent output layer for each task.
- Soft Modularization (Yang et al. 2020): a method that learns basic policy modules and automatically generates soft combinations probability via a routing network for multi-task DRL.
- KTM-DRL (Xu et al. 2020): a knowledge transfer-based method that combines offline knowledge learning and online performance improvement.
- G-Surgery (Yu et al. 2020a): a gradient correction-based method that projects conflicting gradients onto the normal plane of the gradient of each task.



(a) Meta-world MT10 (b) HalfCheetah Task Group

Figure 3: Ablation study for policy correction on two benchmarks. Analyze the contribution of policy correction and adaptive weight adjusting technique in our method,which are denoted as Cor and Ada, respectively. O means PiCor uses this technique, and X means PiCor without. The results present the mean and standard deviation averaged over five runs.

Benchmark	PiCor (Ada: O, Cor: O),	PiCor (Ada: X, Cor: O),	PiCor (Ada: O, Cor: X),	PiCor (Ada: X, Cor: X)
Meta-world MT10	**73.3%**	60.0%	24.0%	23.3%
HalfCheetah Task Group	**15200**	12400	7518	7080

Table 2: Comparisons of different techniques in PiCor on two benchmarks.

Experimental Settings

In the experiments, we identify different tasks through one-hot task encoding and assume the task distribution is uniform. We evaluate all methods over five runs independently, and the mean and standard variance of the results are reported for each benchmark. For MT10, all algorithms are evaluated based on the success rate well-defined in Meta-world (Yu et al. 2020b). And for the HalfCheetah Task Group, policies are measured by episode return. Since the task difficulty and the number of tasks are different between the two benchmarks, we train all methods with 5 million and 2.4 million time steps on Meta-world MT10 and HalfCheetah Task Group, respectively. For fair comparisons, baseline methods train with the same network structure except for MTMHSAC and soft modularization (Yang et al. 2020). Each neural network contains two hidden layers with a width of 512 neurons. Soft modularization needs to use a specially designed network structure, so we use its variant called the deep version with the same number of parameters for comparison. In addition, MTMHSAC utilizes a multi-head policy. Furthermore, all algorithms use hierarchical experience replay proposed in KTM-DRL (Xu et al. 2020), which means that transitions of each task are stored independently in the task-specific replay buffer. For the implementation, Soft Modularization [1], KTM-DRL [2], G-Surgery [3] are implemented by using their released code.

[1]https://github.com/RchalYang/Soft-Module
[2]https://github.com/xuzhiyuan1528/KTM-DRL
[3]https://github.com/WeiChengTseng/Pytorch-PCGrad

Main Results

Meta-world MT10. The Meta-world consists of fifty robotic simulated manipulation tasks well-designed to learn various manipulation skills. We consider MT10 from Meta-word to investigate the performance and sample efficiency of PiCor. Figure 2(a) shows the learning curves of average success rate over policy optimization of PiCor and baselines on the Meta-world MT10. From the learning curves, the performance of PiCor outperforms all baselines and presents a significant sample efficiency. We also observe that PiCor exceeds all other methods, using only half of all samples. Moreover, Figure 2(a) also shows that the learning curve of PiCor is steeper in the early stage, which indicates that PiCor has excellent data efficiency. It is worth emphasizing that KTM-DRL seems to learn faster, but the fact is that this benefits from knowledge transfer from expert-level teacher agents. For further analysis, we compare the performance of PiCor with the average performance of the agents, which are trained independently on each task. Specifically, each independent agent is trained through the SAC algorithm and keeps the same experimental settings as PiCor. We note that PiCor exceeds the average performance of independent agents with 70% of total samples, which means PiCor can automatically learn the shared knowledge among tasks leading to better performance. These results demonstrate that PiCor can train a general agent to master multiple tasks efficiently.

HalfCheetah Task Group. As for the benchmark of the HalfCheetah Task Group, it contains eight similar environments modifying specific body parts through morphological modification. The training curves of the average episode return of PiCor and baselines on the HalfCheetah Task Group are in Figure 2(b). The results demonstrate that PiCor shows an obvious advantage on locomotion tasks compared to baselines. Given the same samples, the performance of PiCor exceeds baselines by 40.4%. On the other hand, PiCor reaches the same performance as baselines with only 12% of the total sample. We observe that the knowledge transfer-based method leverages offline knowledge transfer to improve performance at the beginning rapidly, but online policy learning has slightly improved. Compared with the average performance of the independent agents, PiCor also shows an improvement by a large margin. These results again demonstrate that PiCor improves the performance and sample efficiency of multi-task DRL methods on various complex tasks. The results of both benchmarks are displayed in the table 1.

Ablation Study

Policy Correction. In order to investigate the advantages of using policy correction, we conduct extended ablation experiments. PiCor utilizes SAC as the backbone algorithm. Therefore, MTSAC can be considered PiCor with only the policy optimization phase. However, we notice that MTSAC has poor performance on both benchmarks, which shows slow performance improvement and has a large gap with PiCor. It can also be noticed that policy correction can significantly improve multi-task asymptotic performance because it effectively mitigates the harmful gradient interference among tasks. Figure 3(a) and 3(b) shows that PiCor reaches the same performance as MTSAC with less than 3% of total samples. Moreover, we remark that the speed of performance improvement of PiCor is noticeable by leveraging policy correction. These results demonstrate that policy correction can substantially improve sampling efficiency and play an essential role in our learning framework.

Adaptive Weight Adjusting. A detailed analysis is provided to evaluate the effect of adaptively balancing the learning speed across tasks. To investigate the benefits of adaptive adjusting of learning progress between tasks, we compare the performance of PiCor with adaptive weight adjusting and with a constant weight on two benchmarks. Other settings are maintained the same. Figure 3(a) and Figure 3(b) show the training curves on two benchmarks and report the mean and the standard variance over three seeds. And the final performance is given in table 2. By additionally utilizing adaptive weight to balance the learning progress of each task dynamically, the data efficiency and average performance are improved. However, when PiCor removes the adaptive weight adjust strategy, the performance drops by 16% and 18% on the Meta-world MT10 and HalfCheetah Task Groups, respectively. At the same time, we also note that adaptive weight adjustment cannot provide significant benefits without policy correction from the table 2.

To present the effect of learning rate balance among tasks, we plot the weight change of each task on MT10 during learning in Figure 4. We train an independent agent on each task to measure the task difficulty. From the learning curves of single-tasks, drawer-close and window-open are easy to solve, reach and push are difficult. By comparing the difficulty of tasks, we find that the effect of adaptive weight adjusting is in line with the expectation. For example, Figure 4 shows that the more difficult tasks have larger weights to keep the knowledge from losing during the training, while the simple tasks have smaller weights.

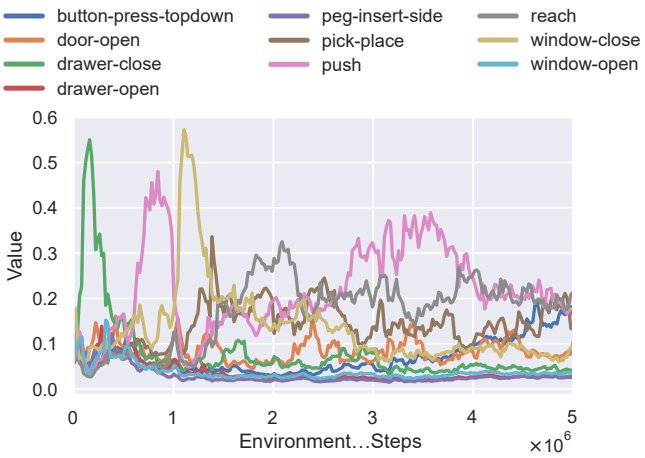


Figure 4: Weight adaptive adjusting process of ten tasks from Meta-world MT10. Various tasks show different weights according to the difficulty of tasks.

Conclusion

In this paper, we propose PiCor, an efficient and novel algorithm framework for training a general agent capable of mastering multiple tasks. PiCor mitigates gradient interference and balances learning speeds among different tasks via two-phase optimization. Specifically, PiCor provides a simple and general solution to multi-task DRL, which can apply to a wide range of domains for mastering diverse skills or handling different tasks. We conduct comprehensive experiments on simulated robotic manipulation and continuous control benchmarks for empirical evaluation and analysis. From experimental results, we conclude that 1) PiCor leads to across-the-board improvements in the performance and sample efficiency on both benchmarks comparing with recent state-of-art methods. 2) PiCor can significantly improve sample efficiency and automatically share knowledge among tasks. 3) Adaptive weight adjusting can balance the learning rate between tasks and is an important component in PiCor. As for future work, we would like to explore how to leverage expert knowledge to assist in the constraint set construction to reduce the computational complexity of policy correction and further improve the sample efficiency of the algorithm. Finally, we hope our work can attract more researchers to pay attention to the sample efficiency of multi-task optimization leading to practical applications.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained Policy Optimization. In Precup, d.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, 22–31. PMLR.

Arora, H.; Kumar, R.; Krone, J.; and Li, C. 2018. Multi-task Learning for Continuous Control. arXiv:1802.01034.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, 794–803. PMLR.

D'Eramo, C.; Tateo, D.; Bonarini, A.; Restelli, M.; and Peters, J. 2020. Sharing Knowledge in Multi-Task Deep Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

Devin, C.; Gupta, A.; Darrell, T.; Abbeel, P.; and Levine, S. 2017. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2169–2176.

Fujimoto, S.; van Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, 1587–1596. PMLR.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, 1861–1870. PMLR.

Henderson, P.; Chang, W.; Shkurti, F.; Hansen, J.; Meger, D.; and Dudek, G. 2017. Benchmark Environments for Multitask Learning in Continuous Domains. arXiv:1708.04352.

Hessel, M.; Soyer, H.; Espeholt, L.; Czarnecki, W.; Schmitt, S.; and van Hasselt, H. 2019. Multi-Task Deep Reinforcement Learning with PopArt. In *AAAI Conference on Artificial Intelligence*, volume 33, 3796–3803.

Kakade, S. M.; and Langford, J. 2002. Approximately Optimal Approximate Reinforcement Learning. In Sammut, C.; and Hoffmann, A. G., eds., *Proceedings of the 19th International Conference on Machine Learning (ICML)*, 267–274. Morgan Kaufmann.

Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; and Levine, S. 2018. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In Billard, A.; Dragan, A.; Peters, J.; and Morimoto, J., eds., *Proceedings of The 2nd Conference on Robot Learning (CoRL)*, volume 87, 651–673. PMLR.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521(7553): 436–444.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17(39): 1–40.

Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S. B.; Meng, Z.; and Alizadeh, M. 2019. Learning Scheduling Algorithms for Data Processing Clusters. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 270–288. New York, NY, USA: Association for Computing Machinery.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Parisotto, E.; Ba, L. J.; and Salakhutdinov, R. 2016. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

Rajeswaran, A.; Kumar, V.; Gupta, A.; Vezzani, G.; Schulman, J.; Todorov, E.; and Levine, S. 2018. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*. Pittsburgh, Pennsylvania.

Rosenbaum, C.; Klinger, T.; and Riemer, M. 2018. Routing Networks: Adaptive Selection of Non-Linear Functions for Multi-Task Learning. In *International Conference on Learning Representations (ICLR)*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust Region Policy Optimization. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, 1889–1897. Lille, France: PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. arXiv:1707.06347.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354–359.

Suteu, M.; and ke Guo, Y. 2020. Regularizing Deep Multi-Task Networks using Orthogonal Gradients. In *International Conference on Learning Representations (ICLR)*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tang, X.; Qin, Z.; Zhang, F.; Wang, Z.; Xu, Z.; Ma, Y.; Zhu, H.; and Ye, J. 2019. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the*

25th ACM SIGKDD international conference on knowledge discovery & data mining (KDD), 1780–1790.

Teh, Y.; Bapst, V.; Czarnecki, W. M.; Quan, J.; Kirkpatrick, J.; Hadsell, R.; Heess, N.; and Pascanu, R. 2017. Distral: Robust multitask reinforcement learning. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Xu, Z.; Wu, K.; Che, Z.; Tang, J.; and Ye, J. 2020. Knowledge Transfer in Multi-Task Deep Reinforcement Learning for Continuous Control. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 15146–15155. Curran Associates, Inc.

Yang, R.; Xu, H.; WU, Y.; and Wang, X. 2020. Multi-Task Reinforcement Learning with Soft Modularization. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 4767–4777. Curran Associates, Inc.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020a. Gradient Surgery for Multi-Task Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 5824–5836. Curran Associates, Inc.

Yu, T.; Quillen, D.; He, Z.; Julian, R.; Hausman, K.; Finn, C.; and Levine, S. 2020b. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100, 1094–1100. PMLR.