Dual Policy Distillation

Kwei-Herng Lai* , **Daochen Zha*** , **Yuening Li** and **Xia Hu**
Department of Computer Science and Engineering, Texas A&M University
{khlai037, daochen.zha, yueningl, xiahu}@tamu.edu

Abstract

Policy distillation, which transfers a teacher policy to a student policy has achieved great success in challenging tasks of deep reinforcement learning. This teacher-student framework requires a well-trained teacher model which is computationally expensive. Moreover, the performance of the student model could be limited by the teacher model if the teacher model is not optimal. In the light of collaborative learning, we study the feasibility of involving joint intellectual efforts from diverse perspectives of student models. In this work, we introduce *dual policy distillation* (DPD), a student-student framework in which two learners operate on the same environment to explore different perspectives of the environment and extract knowledge from each other to enhance their learning. The key challenge in developing this dual learning framework is to identify the beneficial knowledge from the peer learner for contemporary learning-based reinforcement learning algorithms, since it is unclear whether the knowledge distilled from an imperfect and noisy peer learner would be helpful. To address the challenge, we theoretically justify that distilling knowledge from a peer learner will lead to policy improvement and propose a disadvantageous distillation strategy based on the theoretical results. The conducted experiments on several continuous control tasks show that the proposed framework achieves superior performance with a learning-based agent and function approximation without the use of expensive teacher models.

1 Introduction

Reinforcement learning (RL), especially deep reinforcement learning has achieved great success in various domains [Sutton and Barto, 2018], ranging from robotic control [Levine *et al.*, 2016], perfect information games [Silver *et al.*, 2017] to imperfect information games [Zha *et al.*, 2019a]. However, it usually requires a large number of interactions with the environment to obtain high-level performance [Salimans *et al.*, 2017]. Recently, works have been proposed to study how we can transfer knowledge from one or more teacher models to a student model so that we can train an agent based on a pre-trained expert model [Rusu *et al.*, 2016; Czarnecki *et al.*, 2019]. One of the simple yet effective techniques is called policy distillation [Rusu *et al.*, 2016], which uses supervised regression to train a student model to produce the same output distribution as the teacher model. Policy distillation has achieved great success and led to stronger performance in challenging domains [Teh *et al.*, 2017; Yin and Pan, 2017]. Unfortunately, it is computationally expensive to obtain a teacher policy since a well-performed pre-trained model is often not available. In addition, the performance of the student model could be restrained by the teacher model if the teacher model is sub-optimal.

In cognitive psychology, collaborative learning illustrates a situation in which a group of students works together to search for solutions [Dillenbourg, 1999]. Different from the traditional teacher-student relationship where students non-interactively receive information from teachers, collaborative learning involves joint intellectual efforts from diverse perspectives of students [Smith and MacGregor, 1992]. Motivated by this, we study the feasibility of collaborative learning on student policies without the use of pre-trained teacher models, and the methodology of extracting beneficial knowledge from a peer policy to accelerate learning, analogous to the human ability to learn from others.

In this paper, we introduce *dual policy distillation* (DPD), a student-student framework in which two policies operate on the same environment and extract knowledge from each other to benefit their learning. There are mainly two challenges in developing such dual learning framework upon contemporary learning-based RL agents. First, different from conventional policy distillation, which uses an expert policy, both policies in DPD are imperfect and may generate noisy outputs in the training process. It is unclear whether the regression to these noisy data is helpful. Second, contemporary RL algorithms usually use function approximators to approximate the policy function and the value function. The inaccurate estimations of the functions make it challenging to design practical algorithms that can be combined with learning-based agents.

To address the challenges above, apart from the original policy, we introduce another policy which is simultaneously trained in the same environment with different initialization. Each of the two policies iteratively optimizes its own RL

*These two authors contributed equally in this work

objective and updates a distillation objective which extracts knowledge from the other peer policy. One may find it surprising that the proposed framework tries to simultaneously encourage the uniqueness of the policy, and keep the two polices close with distillation objective. In the following sections, we demonstrate that this method is able to balance exploration and exploitation through parallelization and distillation, respectively, with two nice properties: (1) it does not require an expert policy as teacher signals in the sense that the two student policies explore different aspects of the environment and share knowledge with each other; (2) distilling knowledge from a peer policy has theoretical policy improvement and it can achieve satisfactory performance with a learning-based agent and function approximation in our empirical results.

Through addressing the challenges, in this paper, we make the following contributions:

- We introduce dual policy distillation (DPD), a student-student framework in which two polices extract beneficial knowledge from each other to help their learning.
- We provide a theoretical justification of the policy improvement of DPD. We show that in the ideal case, by distilling a hypothetical hybrid policy, each of the policies has guaranteed policy improvement.
- We propose a practical algorithm[1] based on our theoretical results. The algorithm uses a disadvantageous policy distillation strategy which prioritizes the distillation at disadvantage states and pushes each of the two policies towards the optimal policy. Experiments on several continuous control tasks demonstrate that the proposed DPD significantly enhances each of the two policies.

2 Preliminaries

In this section, we introduce reinforcement learning (RL), the background of policy distillation, and the notations used in this paper.

In the following, we consider standard reinforcement learning which is denoted by a sextuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}_T, \mathcal{R}, \gamma, p_0)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}_T : \mathcal{S}\times\mathcal{A}\to\mathcal{S}$ is the state transition function, $\mathcal{R} : \mathcal{S}\times\mathcal{A}\times\mathcal{S}\to\mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and p_0 is the distribution of the initial state. The interactions with the environment can be formalized as a Markov decision process: at each timestep t, an agent takes an action $a_t \in \mathcal{A}$ at state $s_t \in \mathcal{S}$ and observes the next state s_{t+1} with a reward signal r_t. This results in a trajectory τ which consists of a sequence of triplets of states, actions and rewards, i.e., $\tau = \{(s_t, a_t, r_t)\}_{t=1,...,T}$, where T is the terminal timestep. The objective of RL algorithm is to learn a policy $\pi : \mathcal{S}\to\mathcal{A}$ that maximizes the cumulative reward $R = \mathbb{E}[\sum_{t=1}^{\infty}\gamma^t r_t]$.

We use standard definitions of value function, state-action value function, and advantage function of a policy π, i.e., $V^\pi(s_t) = \mathbb{E}_{a_t,s_{t+1},...|\pi}[\sum_0^\infty \gamma^l r_{t+1}]$, $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1},a_{t+1},...|\pi}[\sum_{t=0}^\infty \gamma^l r_{t+1}]$, and $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$, where $a_t, s_{t+1}, ...|\pi$ and $s_{t+1}, a_{t+1}, ...|\pi$ denote the resulting trajectories from the environment if we follow policy π. We use $\rho_\pi(s)$ to denote discounted visitation frequencies of state s, that is , $\rho_\pi(s) = \sum_{t=0}^\infty \gamma^t p(s_t = s)$, where $p(s_t = s)$ is the probability of s being visited at timestep t.

[1] https://github.com/datamllab/dual-policy-distillation

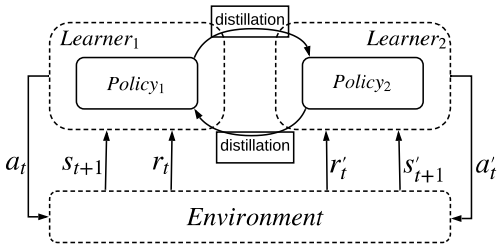


Figure 1: An overview of dual policy distillation (DPD) framework. The two learners interact with the same environment with different initialization. In each iteration, each of the two learners updates two objectives: the RL objective which optimizes the cumulative reward on the environment, and the distillation objective which conducts regression to its peer policy.

Policy distillation is a simple yet effective method of transferring knowledge from one or more action policies to an untrained network. We denote $\widetilde{\pi}$ as a teacher policy, i.e., a trained model that can generate expert data, and π_θ as an untrained parametric student policy. Policy distillation trains the student policy by conducting regression to the teacher policy, i.e., minimizing the following objective:

$$\mathcal{J} = \mathbb{E}_{s\sim\widetilde{\pi}}[D(\pi_\theta(\cdot|s), \widetilde{\pi}(\cdot|s))] \tag{1}$$

where $s \sim \widetilde{\pi}$ means s follows the distribution of $\rho_\pi(\cdot)$, and $D(\cdot, \cdot)$ is a kind of distance metric. The above description is a general form of policy distillation. There are multiple choices for the distance metrics, such as mean square error, KL divergence or log-likelihood loss [Rusu *et al.*, 2016]. In this paper, we use mean square error for its simplicity.

Note that, in our presented algorithm, we use $\widetilde{\pi}_\phi$ to denote a trainable peer policy with parameters ϕ. π_θ and $\widetilde{\pi}_\phi$ are both student models and are trained interactively.

3 Dual Policy Distillation

In this section, we present dual policy distillation (DPD), a framework that enables knowledge transfer between two student policies operating on the same environment. We consider two policies denoted as π and $\widetilde{\pi}$ respectively. We first theoretically justify that extracting knowledge from a peer policy will lead to policy improvement through a view of hypothetical hybrid policy. Then based on our theoretical results, we present a disadvantageous policy distillation objective that can be combined with learning-based RL algorithms. Figure 1 shows an overview of the proposed framework.

3.1 A View of Hypothetical Hybrid Policy

We first justify that π and $\widetilde{\pi}$ are theoretically complementary, and thus transferring knowledge between two policies will lead to policy improvement for both π and $\widetilde{\pi}$. Consider a hypothetical hybrid policy:

$$\pi^{hypo}(\cdot|s) = \begin{cases} \widetilde{\pi}(\cdot|s) & \xi^{\widetilde{\pi}}(s) > 0 \\ \pi(\cdot|s) & otherwise \end{cases} \tag{2}$$

where $\xi^{\widetilde{\pi}}(s) = V^{\widetilde{\pi}}(s) - V^{\pi}(s)$ represents the advantage of $\widetilde{\pi}$ over π at state s. That is, the hypothetical policy π^{hypo} selectively follows one of π and $\widetilde{\pi}$ depending on which policy has larger expected discounted reward at the state. In Proposition 1, we give that in ideal case the hypothetical hybrid policy is an improved policy compared to π or $\widetilde{\pi}$.

Proposition 1. *For the hypothetical hybrid policy π^{hypo} defined in Eq. 2, we have $\forall s \in \mathcal{S}$, $V^{\pi^{hypo}}(s) \geq V^{\pi}(s)$ and $V^{\pi^{hypo}}(s) \geq V^{\widetilde{\pi}}(s)$.*

Proof: Considering a state $s \in \mathcal{S}$, we prove that $V^{\pi^{hypo}}(s) \geq V^{\pi}(s)$. Define advantage policy at s:

$$\pi_s^{adv} = \begin{cases} \widetilde{\pi} & \xi^{\widetilde{\pi}}(s) > 0 \\ \pi & otherwise. \end{cases} \tag{3}$$

That is, π_s^{adv} is one of π and $\widetilde{\pi}$ such that it has higher value at state s. Note that whether $\pi_s^{adv}(\cdot|s')$ is π or $\widetilde{\pi}$ depends on s instead of s', where s' is a different state. It is straightforward to have

$$V^{\pi_s^{adv}}(s) \geq V^{\pi}(s), \tag{4}$$

$$V^{\pi_s^{adv}}(s) \geq V^{\pi_{s'}^{adv}}(s), \tag{5}$$

where s' is another state and $s \neq s'$.

We now consider an arbitrary state $s_i \in S$ and denote s_{i+1} as the next state. Define

$$V_n(s_i) = \begin{cases} \mathbb{E}_{(s_{i+1},r_i)\sim\pi^{hypo}(s_i)}[r_i + \gamma V_{n-1}(s_{i+1})] & n \geq 1 \\ V^{\pi_{s_i}^{adv}}(s_i) & n = 0, \end{cases} \tag{6}$$

where n is a positive integer. That is, the value in state s_i if we follow π^{hypo} for the first n steps and follow $\pi_{s_{i+n}}^{adv}$ afterwards.

When $n = 0$ we have $V_0(s_i) = V^{\pi_{s_i}^{adv}}(s_i)$. We first prove $\forall s_i \in S, V_1(s_i) \geq V_0(s_i)$.

$$\begin{aligned} V_1(s_i) &= \mathbb{E}_{(s_{i+1},r_i)\sim\pi^{hypo}(s_i)}[r_i + \gamma V_0(s_{i+1})] \\ &= \sum_{s_{i+1},r_i} p^{\pi^{hypo}}(s_{i+1}, r_i|s_i)[r_i + \gamma V_0(s_{i+1})] \\ &= \sum_{s_{i+1},r_i} p^{\pi^{hypo}}(s_{i+1}, r_i|s_i)[r_i + \gamma V^{\pi_{s_{i+1}}^{adv}}(s_{i+1})] \\ &\geq \sum_{s_{i+1},r_i} p^{\pi^{hypo}}(s_{i+1}, r_i|s_i)[r_i + \gamma V^{\pi_{s_i}^{adv}}(s_{i+1})] \\ &= \sum_{s_{i+1},r_i} p^{\pi_{s_i}^{adv}}(s_{i+1}, r_i|s_i)[r_i + \gamma V^{\pi_{s_i}^{adv}}(s_{i+1})] \\ &= V^{\pi_{s_i}^{adv}}(s_i) \\ &= V_0(s_i). \end{aligned} \tag{7}$$

Based on the equations above, when $k \geq 1$, given that $\forall s_i \in S, V_k(s_i) \geq V_{k-1}(s_i)$, we have

$$\begin{aligned} V_{k+1}(s_i) &= \mathbb{E}_{(s_{i+1},r_i)\sim\pi^{hypo}(s_i)}[r_i + \gamma V_k(s_{i+1})] \\ &= \sum_{s_{i+1},r_i} p^{\pi^{hypo}}(s_{i+1}, r_i|s_i)[r_i + \gamma V_k(s_{i+1})] \\ &\geq \sum_{s_{i+1},r_i} p^{\pi^{hypo}}(s_{i+1}, r_i|s_i)[r_i + \gamma V_{k-1}(s_{i+1})] \\ &= \mathbb{E}_{(s_{i+1},r_i)\sim\pi^{hypo}(s_i)}[r_i + \gamma V_{k-1}(s_{i+1})] \\ &= V_k(s_i). \end{aligned} \tag{8}$$

By induction, we can conclude that $\forall n \geq 0, \forall s \in \mathcal{S}, V_n(s) \geq V^{\pi_s^{adv}}(s) \geq V^{\pi}(s)$. Thus, $\forall s \in \mathcal{S}$, $V^{\pi^{hypo}}(s) \geq V^{\pi}(s)$. Similarly, we have $\forall s \in \mathcal{S}, V^{\pi^{hypo}}(s) \geq V^{\widetilde{\pi}}(s)$. The above proof is also applicable in continuous space if we replace sum operation with integration.

Directly following the well-known policy improvement theorem [Sutton and Barto, 2018], Proposition 1 suggests that, the hypothetical hybrid policy π^{hypo} defined in Eq. 2 is at least as good as π and $\widetilde{\pi}$. If $V^{\pi}(s) > V^{\widetilde{\pi}}(s)$ at some states and $V^{\pi}(s') < V^{\widetilde{\pi}}(s')$ at some other states, π^{hypo} will be strictly better than π and $\widetilde{\pi}$. Our empirical observation also supports this intuition (see Figure 3). Thus, it will lead to theoretical policy improvement if we let π and $\widetilde{\pi}$ conduct regression to the hypothetical hybrid policy π^{hypo}.

3.2 Disadvantageous Policy Distillation

Based on the above theoretical results, we introduce a practical dual distillation strategy which can be combined with learning-based RL agents.

For a practical algorithm, rather than building the hypothetical hybrid policy π^{hypo} at every step, we prefer to use an objective to train the policy. In Proposition 2, we give that under mild conditions, the distillation of π^{hypo} is equivalent to minimizing a simple objective.

Proposition 2. *The distillation to π from the hypothetical hybrid policy π^{hypo} defined in Eq. 2 is equivalent to minimizing the following objective:*

$$\mathcal{J} = \mathbb{E}_{s\sim\widetilde{\pi}}[D(\pi(\cdot|s), \widetilde{\pi}(\cdot|s))\mathbb{1}(\xi^{\widetilde{\pi}}(s) > 0)], \tag{9}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $D(\cdot,\cdot)$ is denoted as the distance metric.

Proof: Since the two policies π and $\widetilde{\pi}$ have similar state visiting frequency, the distillation to π^{hypo} defined in Eq. 1 can be rewritten as follow:

$$\begin{aligned} &\mathbb{E}_{s\sim\pi^{hypo}}[D(\pi(\cdot|s), \pi^{hypo}(\cdot|s))] \\ =& \sum_{s\sim\rho_{\pi^{hypo}}(\cdot)} D(\pi(\cdot|s), \pi^{hypo}(\cdot|s)) \\ =& \sum_{s\sim\rho_{\pi^{hypo}}(\cdot);\xi^{\widetilde{\pi}}(s)>0} D(\pi(\cdot|s), \widetilde{\pi}(\cdot|s)) + \sum_{s\sim\rho_{\pi^{hypo}}(\cdot);\xi^{\widetilde{\pi}}(s)\leq 0} D(\pi(\cdot|s), \pi(\cdot|s)) \\ =& \sum_{s\sim\rho_{\widetilde{\pi}}(\cdot);\xi^{\widetilde{\pi}}(s)>0} D(\pi(\cdot|s), \widetilde{\pi}(\cdot|s)) + \sum_{s\sim\rho_{\widetilde{\pi}}(\cdot);\xi^{\widetilde{\pi}}(s)\leq 0} D(\pi(\cdot|s), \pi(\cdot|s)) \\ =& \mathbb{E}_{s\sim\widetilde{\pi}}[D(\pi(\cdot|s), \widetilde{\pi}(\cdot|s))\mathbb{1}(\xi^{\widetilde{\pi}}(s) > 0)] \end{aligned} \tag{10}$$

Algorithm 1 DPD: dual policy distillation

1: **Input:** policy π_θ, peer policy $\widetilde{\pi}_\phi$, and maximum iteration number M
2: **for** iteration = $1, M$ **do**
3: Execute π_θ to generate trajectories and save them into buffer B_π
4: Update π_θ based on its RL objective
5: Update π_θ based on Eq. 11
6: Execute $\widetilde{\pi}_\phi$ to generate trajectories and save them into buffer $B_{\widetilde{\pi}}$
7: Update $\widetilde{\pi}_\phi$ based on its RL objective
8: Update $\widetilde{\pi}_\phi$ based on Eq. 12
9: **end for**

The difference between $\rho_\pi(\cdot)$ and $\rho_{\widetilde{\pi}}(\cdot)$ can be ignored in our dual learning setting, since the dual distillation will push the two policies to perform similar actions and hence will result in similar state visiting frequencies. Our empirical observations also support this assumption in that both the learning curves and the outputs of the policy network are very similar during the learning process (see Figure 3). The result of Eq. 9 suggests a simple intuition: the states at which the peer policy is advantageous will be more helpful in distillation; otherwise, maintaining the current policy at the state would be a better choice. We call this strategy *disadvantageous policy distillation* because it prioritizes the distillation of the states at which the current policy is more disadvantageous than the peer policy.

For now, we ignore the estimation error for the value functions which are usually approximated by deep neural networks. However, in the approximate setting, we have to consider the inaccurate estimations of value functions, which makes it difficult to optimize Eq. 9. In our preliminary experiments, we observed that directly update the policies based on Eq. 9 will misclassify many states and lead to sub-optimal performances. Therefore, we propose to soften Eq. 9 and introduce weighted objectives $\mathcal{J}^w_{\pi_\theta}(\theta)$ and $\mathcal{J}^w_{\widetilde{\pi}_\phi}(\phi)$ to update parametric policies π_θ and $\widetilde{\pi}_\phi$:

$$\mathcal{J}^w_{\pi_\theta}(\theta) = \mathbb{E}_{s\sim\widetilde{\pi}_\phi}[D(\pi_\theta(\cdot|s), \widetilde{\pi}_\phi(\cdot|s)) \exp(\alpha\xi^{\widetilde{\pi}_\phi}(s))], \quad (11)$$

$$\mathcal{J}^w_{\widetilde{\pi}_\phi}(\phi) = \mathbb{E}_{s\sim\pi_\theta}[D(\widetilde{\pi}_\phi(\cdot|s), \pi_\theta(\cdot|s)) \exp(\alpha\xi^{\pi_\theta}(s))], \quad (12)$$

where $\exp(\alpha\xi^{\widetilde{\pi}_\phi}(s))$ and $\exp(\alpha\xi^{\pi_\theta}(s))$ are confidence scores, and α controls the confidence level which should be chosen depending on how accurate the value function estimation is.

We now describe how we combine the objectives in Eq. 11 and 12 with a learning-based agent. Given two parametric policies π_θ and $\widetilde{\pi}_\phi$ operating on the same environment, we use an alternating update framework as follows. In the first step, policy π_θ is updated based on its own RL objective. In the second step, π_θ is updated based on the distillation objective. Specifically, we sample a mini-batch of transitions from the buffer of $\widetilde{\pi}_\phi$ and compute the advantage $\xi^{\widetilde{\pi}_\phi}(s)$ for each sampled state and update the policy based on the distillation objective. We then do the same updates to its peer policy $\widetilde{\pi}_\phi$. As a result, each policy learns to optimize its RL objective and simultaneously extracts useful knowledge from its peer policy to enhance itself. The algorithm is summarized in Algorithm 1.

3.3 Connection to Value Iteration

In this subsection, we justify that the proposed distillation objective will lead to similar effects as by classical value iteration [Sutton and Barto, 2018]. Let π^* be an optimal policy and $V^{\pi^*}(s)$ be the optimal values, i.e., the expected discounted future rewards if we start from s and follow the optimal policy. Value iteration method iteratively updates the state values as follow:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} \mathcal{P}_T(s'|s,a)[\mathcal{R}(s,a,s') + V_i(s')], \quad (13)$$

where $V_i(\cdot)$ and $V_{i+1}(\cdot)$ are the values of the current step and the next step respectively. The intuition of this update rule is to compute the maximum value by choosing the most valuable action in each iteration to push the values towards and finally converge to the optimal values. Define $\mathcal{S}_{\widetilde{\pi}}$ as the set of disadvantage states, i.e, $S_{\widetilde{\pi}} = \{s|V^{\widetilde{\pi}}(s) > V^{\pi}(s)\}$. Then it is straightforward to have

$$V^{\pi^*}(s) \geq V^{\widetilde{\pi}}(s) > V^{\pi}(s), \forall s \in \mathcal{S}_{\widetilde{\pi}}. \quad (14)$$

The distillation of the states in $S_{\widetilde{\pi}}$ has similar effects for π. Encouraging π to choose the actions from $\widetilde{\pi}$ for the states in $S_{\widetilde{\pi}}$ will push the values of π towards optimal values because these actions lead to larger values based on $V^{\widetilde{\pi}}(\cdot)$. Thus, both Eq. 11 and 12 will optimize the values of the two policies.

4 Experiments

In this section, we empirically evaluate the proposed dual policy distillation (DPD) framework. We develop two instances of DPD by two benchmark RL algorithms, and evaluate them on four continuous control tasks. Our experiments are designed to answer the following questions:

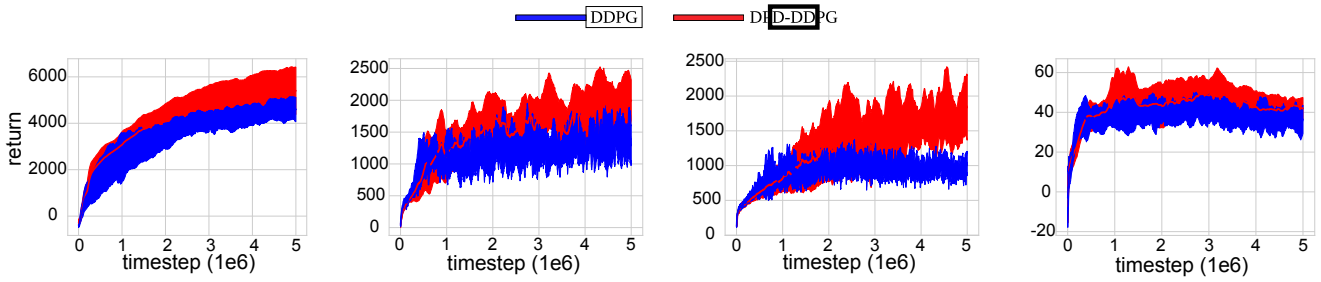
- **Q1:** Is DPD able to improve the performance in both on-policy and off-policy settings (Sec. 4.2)?
- **Q2:** How will the values and actions outputted by the models evolve during training (Sec. 4.3)?

4.1 Experimental Setting

Our experiments are implemented upon PPO [Schulman *et al.*, 2017] and DDPG [Lillicrap *et al.*, 2016], which are benchmark RL algorithms implemented in OpenAI baselines[2]. We follow all the hyper-parameters setting and network structures for our DPD implementation and all the baselines we considered. Since the policy network in DDPG is deterministic, we directly use the outputs of the Q-network in DDPG to estimate the state values. We use mean square error to compute the distance between the output actions of the two policies for the distillation objective. We consider the following baselines:

- **DDPG:** the vanilla DDPG without policy distillation.
- **PPO:** the vanilla PPO without policy distillation.

[2]https://github.com/openai/baselines



(a) HalfCheetah (b) Walker2d (c) Humanoid (d) Swimmer

Figure 2: Overall performance comparison on four continuous control tasks in off-policy setting. The shaded area represents mean $\pm$ standard deviation. For a fair comparison, each learner of DPD is run for 2.5×10^6 timesteps, and the X-axis (timestep) of DPD is stretched to 5.0×10^6 . Thus, all the algorithms are compared in the same condition with 5×10^6 timesteps in total. The learning curves are averaged over 10 random seeds. The performance is measured by the average return over episodes.

The experiments are conducted on several continuous control tasks from OpenAI gym[3] [Brockman *et al.*, 2016]: Swimmer-v2, HalfCheetah-v2, Walker2d-v2, Humanoid-v2.

4.2 Overall Performance

We conduct experiments on both on- and off-policy settings to validate the proposed disadvantageous distillation strategy is beneficial in general. Since DPD requires updating two policies simultaneously, we run vanilla DDPG and PPO for twice the number of timesteps as that of DPD for a fair comparison, i.e., we run DPD-DDPG and DPD-PPO for 2.5×10^6 and 1×10^7 timesteps; and vanilla DDPG and PPO for 5×10^6 and 2×10^7 timesteps. Figure 2 plot the learning curve of off-policy settings, and the result of on-policy settings is tabulated in Table 1.

Based on the result, we observed that, each of the two policies in DPD is significantly enhanced by the dual distillation, and DPD outperforms vanilla DDPG and PPO within the same running timesteps in the 4 tasks. Specifically, when comparing DPD and DDPG at 2.5×10^6 timesteps, the maximum return of DPD has an improvement of more than 15 percent in 3 out of 4 tasks. By comparing the performance of DPD at 2.5×10^6 timesteps with that of both PPO and DDPG at 5×10^6 timesteps, the maximum return of DPD has an improvement of more than 10 percent in the tasks. In our experiments, we only empirically explore α over a small set. It is possible to further improve DPD if exploring more α values. The results suggest that it is promising to exploit the knowledge from a peer policy and the proposed framework is generally applicable for both on- and off-policy algorithms.

4.3 Analysis of the Dual Distillation

We study how the Q-values and the actions outputted by the two learners evolve under DPD framework. We randomly sample some states from the rollouts performed by a pre-trained model which is obtained by training DDPG for 5×10^6 timesteps. We then feed these states to the DPD and DDPG models in different training stages. The outputted Q-values and actions are illustrated in Figure 3.

[3]https://gym.openai.com/

Mean	PPO	DPD-PPO
HalfCheetah	2947.17 $\pm$201.11	**3051.28 $\pm$190.51**
Walker2d	3694.79 $\pm$224.12	**3857.23 $\pm$143.58**
Humanoid	2164.36 $\pm$127.40	**2242.19 $\pm$230.72**
Swimmer	97.36 $\pm$0.57	**98.90 $\pm$2.50**
Max	PPO	DPD-PPO
HalfCheetah	4031.77 $\pm$3048.85	**6373.40 $\pm$941.01**
Walker2d	4738.16 $\pm$161.22	**5233.56 $\pm$286.84**
Humanoid	3518.00 $\pm$278.31	**3885.83 $\pm$261.89**
Swimmer	110.82 $\pm$0.83	**120.99 $\pm$8.01**

Table 1: Overall performance comparison on four continuous control tasks in on-policy setting. For a fair comparison, each learner of DPD is run for 1×10^7 timesteps, and the PPO learner is run for 2×10^7 timesteps. The result are averaged over 5 random seeds. The performance is measured by the mean return over episodes and mean of maximum return of each episode.

For the Q-values, we make two observations. First, in all the training stages, the Q-values outputted by each of the learners tend to be larger at some states and smaller at some other states compared with the other learner. The result supports our hypothesis that the two learners are complimentary so that we can find a hypothetical hybrid policy that has guaranteed policy improvement. Second, we can see that the Q-values of all the states tend to increase throughout the training process, and they increase faster than those of DDPG. The result suggests that the proposed dual distillation indeed pushes the values of the two learners towards the larger optimal values, as indicated by our theoretical justification.

For the actions, we calculate the Euclidean distance between the outputted actions of the two learners in DPD, and run two separate DDPG models, calculate the Euclidean distance between their outputted actions for comparison. We observe that the two learners in DPD tend to perform more similar actions than the two DDPG models. The average Euclidean distance for DPD decreases from the early stage (0.43) to the later stage (0.31), which suggests that the dual distillation may make the two learners slowly converge through the training.

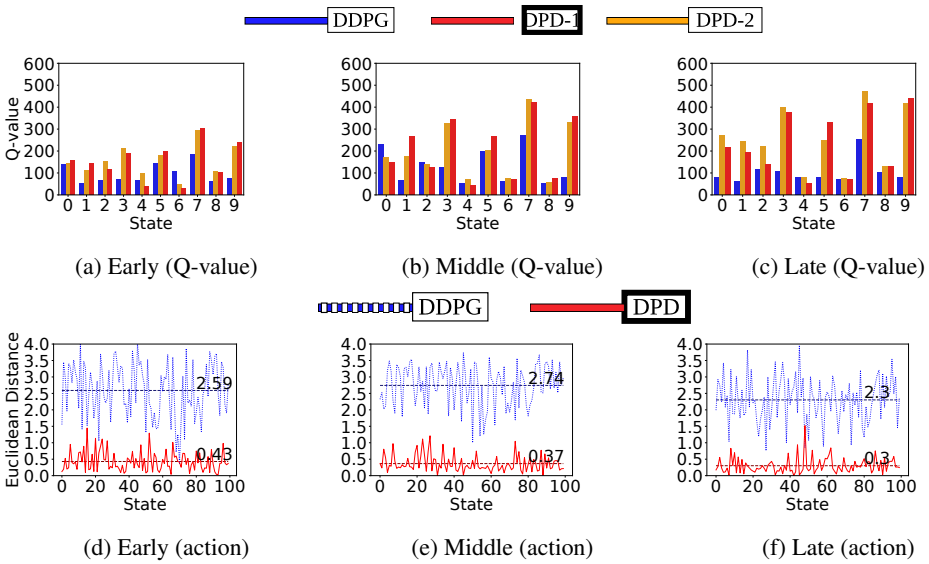


Figure 3: The evolution of Q-values (top) and actions (bottom) outputted by DPD and DDPG in Walker2d. Since DDPG uses a deterministic policy network, the Q-values can represent the state values. DPD-1 and DPD-2 denote the first learner and the second learner in DPD respectively. The dark lines are the average values of the two distances. The x-axis represents the state index. For Q-values evolution, we randomly sample 10 states from the rollouts performed by a pre-trained model which is obtained by training DDPG for 5×10^6 timesteps. From left to right, we plot the Q-values of these 10 states in early (5×10^5 timesteps), middle (1.5×10^6 timesteps) and later (2.5×10^6 timesteps) training stages. For actions evolution, we run DDPG for two separate runs with different random seeds and compute the Euclidean distance between the actions outputted by the two DDPG models as well as the actions outputted by the two learners in DPD. We plot the results of 100 randomly sampled states. Note that we sample 10 or 100 states for better visualization. We have sampled multiple times and observed similar results.

Since the two learners have similar policies, the result further supports our assumption that the two learners in DPD have similar state visiting frequencies.

5 Related Work

5.1 Teacher-student Framework

A typical teacher-student framework is imitation learning. [Abbeel and Ng, ; Finn *et al.*, 2016]. One representative method is behavior cloning [Bain and Sommut, 1999; Ho and Ermon, 2016; Rusu *et al.*, 2016], which directly trains a student policy based on demonstration data of experts. Previous work shows that it is promising to learn from imperfect demonstrations [Hester *et al.*, 2018] and exploiting own past good experiences will help exploration [Oh *et al.*, 2018]. Our framework also learns from imperfect demonstrations, but treats actions performed by a peer policy as demonstrations. Meta-learning methods is studied to make use of a teacher model to improve the sample efficiency [Xu *et al.*, 2018b; Xu *et al.*, 2018a; Zha *et al.*, 2019b]. Our work extends the traditional teacher-student setting and studies a student-student framework with two student models distilling knowledge from each other. Each model servers as both student and teacher and work together with its peer model to find the solution.

5.2 Multi-agent Reinforcement Learning

Multi-agent reinforcement learning [Littman, 1994; Tan, 1993; Shoham *et al.*, 2003] studies how a group of agents sharing the same environment learns to collaborate and coordinate with each other to achieve a goal. There are several studies on knowledge diffusion in multi-agent setting [Hadfield-Menell *et al.*, 2016; Da Silva *et al.*, 2017; Omidshafiei *et al.*, 2019]. Although our work also introduces two learners, our setting is significantly different from the multi-agent setting in that the two learners in the framework are independently deployed to two instances of the same single-agent environment. The two learners capture various aspects of the same environment and share beneficial knowledge during the training process.

6 Conclusion and Future Work

In this work, we introduce dual policy distillation, a student-student framework which enables two policies to explore different aspects of the environment and exploit the knowledge from each other. Specifically, we propose a distillation strategy which prioritizes the distillation at disadvantage states from the peer policy. The theoretical and empirical results show the promising of the proposed framework. In the future, we will investigate whether our framework can be extended to enable knowledge transfer among multiple tasks and explore the possibility of using our framework to combine the benefits of different RL algorithms.

Acknowledgements

This work is, in part, supported by NSF (#IIS-1750074, #IIS-1939716 and #IIS-1900990). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[Abbeel and Ng,] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. page 1.

[Bain and Sommut, 1999] Michael Bain and Claude Sommut. A framework for behavioural cloning. *Machine Intelligence*, pages 103–129, 1999.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Czarnecki *et al.*, 2019] Wojciech Marian Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. *arXiv preprint arXiv:1902.02186*, 2019.

[Da Silva *et al.*, 2017] Felipe Leno Da Silva, Ruben Glatt, and Anna Helena Reali Costa. Simultaneously learning and advising in multiagent reinforcement learning. In *AAMAS*, pages 1100–1108, 2017.

[Dillenbourg, 1999] Pierre Dillenbourg. *Collaborative learning: Cognitive and computational approaches. advances in learning and instruction series.* ERIC, 1999.

[Finn *et al.*, 2016] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, pages 49–58, 2016.

[Hadfield-Menell *et al.*, 2016] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *NeurIPS*, pages 3909–3917, 2016.

[Hester *et al.*, 2018] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018.

[Ho and Ermon, 2016] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, pages 4565–4573, 2016.

[Levine *et al.*, 2016] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, pages 1334–1373, 2016.

[Lillicrap *et al.*, 2016] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[Littman, 1994] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. 1994.

[Oh *et al.*, 2018] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *ICML*, pages 3878–3887, 2018.

[Omidshafiei *et al.*, 2019] Shayegan Omidshafiei, Dong-Ki Kim, Miao Liu, Gerald Tesauro, Matthew Riemer, Christopher Amato, Murray Campbell, and Jonathan P How. Learning to teach in cooperative multiagent reinforcement learning. In *AAAI*, pages 6128–6136, 2019.

[Rusu *et al.*, 2016] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR*, 2016.

[Salimans *et al.*, 2017] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[Shoham *et al.*, 2003] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. *Web manuscript*, 2003.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, page 354, 2017.

[Smith and MacGregor, 1992] Barbara Leigh Smith and Jean T MacGregor. What is collaborative learning, 1992.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*, pages 330–337, 1993.

[Teh *et al.*, 2017] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *NeurIPS*, pages 4496–4506, 2017.

[Xu *et al.*, 2018a] Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore via meta-policy gradient. In *ICML*, pages 5463–5472, 2018.

[Xu *et al.*, 2018b] Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *NeurIPS*, pages 2396–2407, 2018.

[Yin and Pan, 2017] Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *AAAI*, 2017.

[Zha *et al.*, 2019a] Daochen Zha, Kwei-Herng Lai, Yuanpu Cao, Songyi Huang, Ruzhe Wei, Junyu Guo, and Xia Hu. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

[Zha *et al.*, 2019b] Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. In *IJCAI*, pages 4243–4249, 2019.