Demonstration Guided Actor-Critic Deep Reinforcement Learning for Fast Teaching of Robots in Dynamic Environments

Liang Gong, Te Sun, Xudong Li, Ke Lin, Natalia Díaz-Rodríguez, David Filliat, Zhengfeng Zhang, Junping Zhang

Demonstration Guided Actor-Critic Deep Reinforcement Learning for Fast Teaching of Robots in Dynamic Environments

Liang Gong * **Te Sun** * **Xudong Li** * **Ke Lin** *
Natalia Díaz-Rodríguez ** **David Filliat** ** **Zhengfeng Zhang** ***
Junping Zhang ***

* *School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China (e-mail: gongliang_mi@sjtu.edu.cn)*
** *Computer Science and System Engineering Department and Inria Flowers Team, ENSTA, Institut Polytechnique Paris, Palaiseau, France*
*** *Shanghai Key Laboratory of Intelligent Information Processing and School of Computer Science, Fudan University, Shanghai, China*

Abstract:
Using direct reinforcement learning (RL) to accomplish a task can be very inefficient, especially in robotic configurations where interactions with the environment are lengthy and costly. Instead, learning from expert demonstration (LfD) is an alternative approach to gain better performance in an RL setting, which also greatly improves sample efficiency. We propose a novel demonstration learning framework for actor-critic based algorithms. Firstly, we put forward an environment pre-training paradigm to initialize the model parameters without interacting with the target environment, which effectively avoids the cold start problem in deep RL scenarios.Secondly, we design a general-purpose LfD framework for most of the mainstream actor-critic RL algorithms that include a policy network and a value function like PPO, SAC, TRPO, A3C. Thirdly,we build a dedicated model training platform to perform the human-robot interaction and numerical experimentation. We evaluate the method in six Mujoco simulated locomotion environments and our robot control simulation platform. Results show that several epochs of pre-training can improve the agent's performance over the early stage of training. Also, the final converged performance of the RL algorithm is also boosted by external demonstration. In general the sample efficiency is improved by 30% with the proposed method. Our demonstration pipeline makes full use of the exploration property of the RL algorithm, which is feasible for fast teaching robots in dynamic environments.

Keywords: deep learning, deep reinforcement learning, learning from demonstration (LfD), actor-critic framework, robotics

1. INTRODUCTION

The actor-critic (AC) framework has been extensively used in reinforcement learning (RL) algorithms. This framework combines policy gradient methods and value functions, namely the actor and the critic. However, most actor-critic algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016) heavily rely on agent-environment interaction to improve its performance. Especially during the early stage of learning, the model development requires large amounts of exploration experience, which can be very expensive and time-consuming. Over the course of training from scratch, a reinforcement learning model could not internalize the expert knowledge and human behavioral prior which leads to useless and expensive exploration at early stages, which ends up usually reaching the sub-optimum at much later stage. Thus, there has been plenty of research focusing on the combination of RL algorithms with expert demonstrations. Pre-training the policy network and critic network may be the most common way to leverage expert demonstrations. However, in the case of limited demonstrations, it is prone to overfit during the course of training, resulting in ill-conditioned network parameters and poor performance.

In this study, we proposed a novel strategy (Fig. 1) for learning from demonstration (LfD, Schaal (1997)) that is available for all actor-critic based algorithms, accelerating the convergence of reinforcement learning and improving the final performance. We first construct two separate demonstration environments aimed at pre-training the actor network and the critic network. We then create an auxiliary value function to assist the pre-training of the policy network. After the demonstration process, we use the pre-trained parameters as model initialization, and perform the standard reinforcement learning procedure until convergence in the target environment.

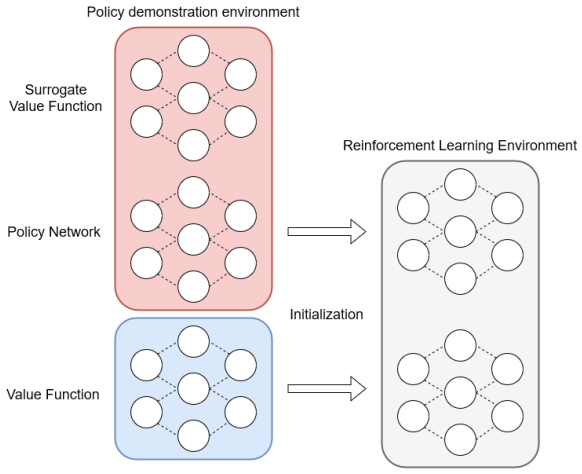

Fig. 1. Demonstration Learning procedure: Two pre-training environments are constructed based on expert replay data. We pre-train the "actor" and "critic" network separately in these two demonstration environments. We train the policy network along with the "Surrogate value function" to fit the policy environments.The final RL model is initialized with the pre-trained network afterwards.

We aim at solving tasks in various environments, combining human demonstration with reinforcement learning. We investigate the performance of our method both in standard simulation benchmark environments and in simulated humanoid robotic configuration. The results show that our method has improved sample efficiency and achieved better performance in benchmark environments. The main contributions of our work can be summarized as:

- The demonstration process augments the episode rewards at early stage during training, i.e., the algorithm costs less samples to reach convergence. The sample efficiency is improved in most benchmark environments.
- Learning behaviors through reinforcement learning is more stable via demonstration even without careful hyperparameter tuning.
- The model with demonstration can achieve higher episode rewards that cannot otherwise be acquired through simple exploration by the RL model itself. The knowledge is distilled from demonstration data through pre-training.

In the following sections, we will first present reinforcement learning, background and related LfD work. Then we will explain our demonstration framework. Next, we will show our experimental configuration and results that manifest the improvement. The article concludes with a brief discussion and possible future works.

2. BACKGROUND

Actor-critic framework Traditional reinforcement learning methods can generally be divided into value-based methods and policy-based methods (Arulkumaran et al., 2017). The former includes the classical Q-Learning (Watkins and Dayan, 1992) algorithm and its deep learning based variants like DQN (Mnih et al., 2013) and Double-DQN (van Hasselt et al., 2015). These methods have better sample efficiency and can update parameters for every time step,but value-based methods often cannot handle high dimensional or continuous control environments. The latter approaches parameterize the policy model and use policy gradient based optimization to guide the agent to explore the environment. These methods present more stable behavior during training and can be easily implemented to solve continuous, high-dimensional problems. However, policy-based methods usually suffer from low sample efficiency since the model can only update parameters after collecting enough interactions.

The actor-critic algorithm (Konda and Tsitsiklis, 2000) is proposed to combine the strong points of policy-based and value-based methods. At each training step, it applies an actor model to choose an action at a given environment state. The critic part aims at estimating the expectation of accumulated rewards at specific states. The actor continuously iterates to increase the probability of choosing a suitable action in different states, and the critic keeps iterating to give a more accurate value assessment. Hence, the actor-critic framework maintains the characteristics of single-step updates and fast learning, and can be applied to the continuous action space. Most state-of-the-art model free reinforcement learning algorithms such as TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018), DDPG (Lillicrap et al., 2015) and PPO (Schulman et al., 2017) are based on the actor-critic framework.

Reinforcement Learning with LfD Reinforcement learning algorithms rely on trial and error mechanisms to leverage the agent exploration and exploitation (Sutton et al., 1998) of the environment. Then collected interaction experiences will be used to optimize the model. However, sparse reward signal can cause the inefficient and random attempts during the early stage of training (Parisi et al., 2020), which will slow down the training of the model.

When acquiring new skills, human beings usually start with learning from expert demonstrations. Inspired by this idea, learning from demonstration is proposed to accelerate robot learning process. Within LfD framework, the robot decision policy learns from examples or demonstrations that are provided by a teacher (Argall et al., 2009). LfD has been applied to improve robot performance in real life (Safavi and Zadeh, 2017), (Konidaris et al., 2012). At the same time, this kind of technique is also studied as an alternative against low sample efficiency of reinforcement learning in some cases. (Hester et al., 2017)) use demonstration data to pre-train the Q-function network in a DQN model. (Vecerík et al., 2017) seek to use priority replay buffer to leverage the ratio of demonstration data for training the network. By using demonstration dataset as part of replay experiences reflects the off-policy characteristic of a model being able to learn from other experiences. However, we need a large amount of demonstration data to maintain the performance. Some approaches to reduce this burden train a student network from several teaching networks in a continuous learning manner through policy distillation (Traoré et al., 2019) or use intrinsic motivation to LfD (Nguyen et al., 2011). Other studies evaluate the artificial expert providing help on demand Bennetot et al. (2020).

(Zuo et al., 2017) and (Lin et al., 2020) both modify the objective function of reinforcement learning to evaluate the similarity of sampled data and demonstration data. The algorithms motivate the agent to act as expert when there are similar frames in the demonstration dataset. However, these approaches may fail in high dimensional observation spaces. On the one hand, demonstration data becomes sparser due to the curse of dimensionality (Bellman, 1966), which renders data less representative. On the other hand, enumerating the demonstration dataset becomes infeasible with large size dataset. High dimensional states will invalidate this approach through the curse of dimensionality.

Works in (Cruz et al., 2017) and (Nair et al., 2017) apply ideas of imitation learning to fit the demonstration dataset in a supervised manner. (Xiang and Su, 2019) use the mean episode rewards in demonstration data as a baseline to amplify the incentive in high reward zone. Another work (Vasan and Pilarski, 2017) trains mechanical prostheses with AC framework agents to imitate complex movements through demonstration methods. However, the idea behind this implementation is to fit the demonstration trajectory using a RL model. However, this approach may degrade the RL's exploration behaviors into a supervised way. The model in (Zhao et al., 2017) projects demonstration data into a nonlinear control model to assist actor-critic learning. Despite the promising results, the method is not easily applicable into other non-robotic domains.

Our demonstration pipeline makes full use of the exploration property of the RL algorithm, which equips the model with strong robustness. Instead of forcing the agent to fit the demonstration data, we create two demonstration environments from expert experience to simulate natural RL learning process. We will explain our method in detail in the next section.

3. METHODOLOGY

3.1 Pre-training Environments

When performing learning from demonstration under the actor-critic RL framework, two artificial demonstration environments are needed during the pre-training phases. The first, *actor demonstration environment*, is for training the policy network. In the common actor-critic framework, the actor model includes a policy network that takes current sate as input and outputs a corresponding action. Thanks to the *Markovian* assumption within the RL scope, only the current state is sufficient for the agent to take an action.

Actor demonstration environment is built as follows: We firstly load all demonstration data which preserves sequential time order of observation during expert replay. Then we recreate this demonstration scenario in the pre-training environment. Under the *reset* command, the environment will randomly choose one episode from the dataset. It will continuously feed the model with time step frames during demonstration, formatted as $\{s_t^{\mathcal{D}}, s_{t+1}^{\mathcal{D}}, r_t^{\mathcal{D}}, a_t^{\mathcal{D}}\}$. $\mathcal{D}$ denotes data coming from demonstrations. Regardless of the action that the model takes, the feeding order of time step data remains the same in order to simulate the demonstration process. After receiving actions, the demonstration environment returns a reward signal r^{π} that describes the distance in probability between the action given by the model and the action taken during demonstration. The reward is also leveraged by the reward signal collected in demonstration dataset. This encourages the agent to imitate expert behavior. We reformulate the actor demonstration environment $\mathcal{E}_{\pi}$as:

$$\begin{aligned} \mathcal{E}_{\pi}(a_t|s_t^{\mathcal{D}}) &\rightarrow \{s_{t+1}^{\mathcal{D}}, r_{t+1}^{\mathcal{D}}, a_{t+1}^{\mathcal{D}}\} \\ r_{t+1}^{\pi} &= \mathbb{P}_{\pi'(s_{t-1}^{\mathcal{D}})}(a_t^{\mathcal{D}}) \times r_t^{\mathcal{D}} \end{aligned} \tag{1}$$

where $\mathbb{P}$ is the probability to choose the action $a_t^{\mathcal{D}}$ at the given state $s_{t-1}^{\mathcal{D}}$. π denotes the expert policy. And we note r^{π} the reward function in $\mathcal{E}_{\pi}$. The second demonstration environment is dedicated to pre-training the "critic" part of the model. The value function takes an environment state as input and outputs the expectation of future accumulated rewards based on the input. The *critic demonstration environment* interacts with the model in the following way: At first, the environment chooses randomly an episode from the demonstration dataset. Data frames can be provided either in expert replay order or in random order. That is because the value function fits the states to episode returns and can be trained in a supervised way. Since the value function generates no action, the model will continuously receive states and rewards from the demonstration environment while the environment runs alone by itself. The interaction in the critic demonstration environment $\mathcal{E}_v$ can be formulated as:

$$\begin{aligned} \mathcal{E}_v(\emptyset|s_t^{\mathcal{D}}) &\rightarrow \{s_{t+1}^{\mathcal{D}}, r_{t+1}^{\mathcal{D}}\} \\ r_{t+1}^{v} &= r_t^{\mathcal{D}} \end{aligned} \tag{2}$$

where r_t^v denotes the reward function in $\mathcal{E}_v$ at time step t.

3.2 Pre-training Method

The reward function during demonstration is not the same as in the RL setting, which demands for another value function $v'_{\pi'}$ to assist the pre-training of the policy network. This is because the policy network training requires a value function as baseline guidance. Yet, the reward signal will change along with the switch from pre-training with LfD to training in the target environment and thus, this auxiliary value function will be discarded once the pre-training is finished. Different from the original PPO clipped loss function, we use this assistant value function to compute the GAE (Generalized Advantage Estimation) (Schulman et al., 2015) style advantage.

The policy π' will be optimized by minimizing the following loss function in original PPO algorithm:

$$L_{\pi'}^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(R'_t(\theta)\hat{A}'_t, \text{clip}(R'_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}'_t)] \tag{3}$$

where,

$$R'_t = \frac{\pi'(a_t|s_t^{\mathcal{D}})}{\pi'_{old}(a_t|s_t^{\mathcal{D}})} \tag{4}$$

$$\hat{A}'_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l (r_{t+l}^{\pi} + \gamma v'_{\pi'}(s_{t+l+1}^{\mathcal{D}}) - v'_{\pi'}(s_{t+l}^{\mathcal{D}})) \tag{5}$$

R'_t denotes the ratio between the probability of current action under the updated and the old policy distribution. And $\hat{A}'_t$ represents the advantage evaluated with $v'_{\pi'}$. As for the value function pre-training, the parameters are optimized in a supervised way. The value estimation adapts

in a way of minimizing $\mathcal{L}_{v'}$ in order to fit the incremental rewards. After all pre-training, the reinforcement learning model will be initialized with the pre-training parameters.

$$\mathcal{L}_{v'} = ||v'(s_t^{\mathcal{D}}) - \sum_{i=t}^{T} \gamma^{T-i} r_i^v||^2 \tag{6}$$

3.3 Algorithm

Proximal Policy Optimization (PPO, Schulman et al. (2017)) algorithm has achieved state-of-the-art performance in many control and gaming tasks. And it adopts actor-critic structure in the model. Although we develop our demonstration learning algorithm mainly based on PPO framework, our method is applicable for most actor-critic reinforcement learning algorithms that include a policy network and a value function like SAC, TRPO, A3C *etc.*

Firstly, we initialize our student model, which has similar structure with PPO, with random parameters. Besides, we initialize our auxiliary value function network to provide states estimation for policy pre-training.

Secondly, we follow the steps described in subsection 3.1 to create demonstration environments. Then, LfD training is based on pre-training π' and v' in the constructed environments.

Third, we initialize PPO model with pre-trained parameters and perform the standard PPO training procedure until the return converges.

We summarize our demonstration algorithm as Algorithm 1

Algorithm 1 LfD training algorithm

Input:Initialize parameters for demonstration model π', v'. Initialize parameters for auxiliary value estimation network $v'_{\pi'}$. Load demonstration data $\{s_t^{\mathcal{D}}, s_{t+1}^{\mathcal{D}}, r_t^{\mathcal{D}}, a_t^{\mathcal{D}}\}$;
Procedure:

1: Create $\mathcal{E}_\pi$ and $\mathcal{E}_v$ with r_t^π and r_t^v as their step rewards:

$$r_{t+1}^\pi = \mathbb{P}_{\pi'(s_{t-1}^{\mathcal{D}})}(a_t^{\mathcal{D}}) \times r_t^{\mathcal{D}}$$

$$r_{t+1}^v = r_t^{\mathcal{D}}$$

2: //Perform LfD training for π' with $v'_{\pi'}$ in $\mathcal{E}_\pi$
3: **for** $k_1 = 1, \ldots, K_\pi$ **do**
4: Update π' and $v'_{\pi'}$ by gradient descent to minimize $\mathcal{L}_{PPO}(\pi', v'_{\pi'})$
5: **end for**
6: //Perform demonstration v' in $\mathcal{E}_v$
7: **for** $k_2 = 1, \ldots, K_v$ **do**
8: Update v' to minimize $\mathcal{L}_{v'}$, where

$$\mathcal{L}_{v'} = ||v'(s_t^{\mathcal{D}}) - \sum_{i=t}^{T} \gamma^{T-i} r_i^v||^2$$

9: **end for**
10: Initialize parameter for PPO model π, v with pre-trained parameters stored in π', v'
11: Perform standard PPO training procedure in target environment $\mathcal{E}$ until convergence

Output: Trained PPO model and parameters

4. EXPERIMENTS

This section will mainly answer three core questions in the following:

(1) Can our proposed method outperform the original RL algorithm in most environments with few expert demonstration episodes? And how much better results can we expect?
(2) Under what circumstances does our proposed method exhibit better performance?
(3) Is our demonstration framework applicable to real robotic setup in an opening scene?

We will present our experimental results to investigate these questions in order.

4.1 Simulated robotic experiments

Results on standard benchmark environments First, we examine the performance of our method across 6 OpenAI gym environments Brockman et al. (2016) with Mujoco implementation Todorov et al. (2012) (Fig. 2: Ant, HalfCheetah, Hopper, Humanoid, Swimmer and Walker. Five of them involve 2D locomotive control problem and Humanoid is a 3D task where the robot aims to walk as far as possible without falling. For every benchmark

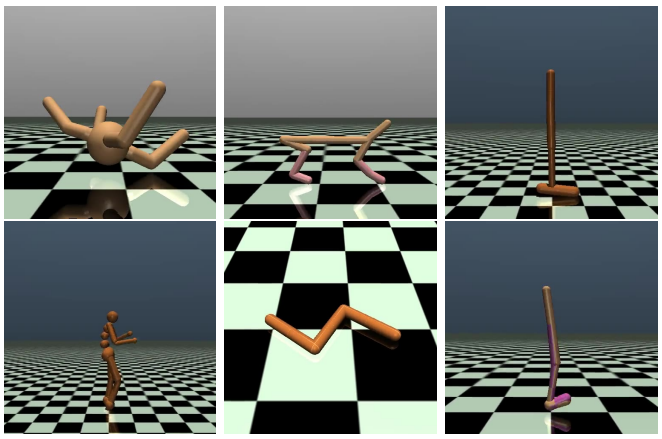

Fig. 2. 2D and 3D robot models used as benchmark environments: Ant-v2, HalfCheetah-v2, Hopper-v2, Humanoid-v2, Swimmer-v2, Walker2d-v2

environment, an expert agent is obtained by training an SAC algorithm until the convergence. We collect demonstration data by replaying the expert model during 20 episodes. The demonstration frame in dataset is recorded as $\{s_t^{\mathcal{D}}, s_{t+1}^{\mathcal{D}}, r_t^{\mathcal{D}}, a_t^{\mathcal{D}}\}$. We then create the demonstration environments described in Section 3.1. To reset one demonstration environment episode, we bootstrap by sampling data from the demonstration dataset.

We choose PPO algorithm as the target algorithm to be trained from scratch; we pre-train a PPO model with the demonstration dataset and train standard PPO in the target environment. We ran each RL experiments with 10 different random seeds per environment. Other hyperparameter information can be found in Table A.1 and the learning curve in Fig. 3:

When the learning curve bends towards top-left, the RL model needs less interactions with the environment to

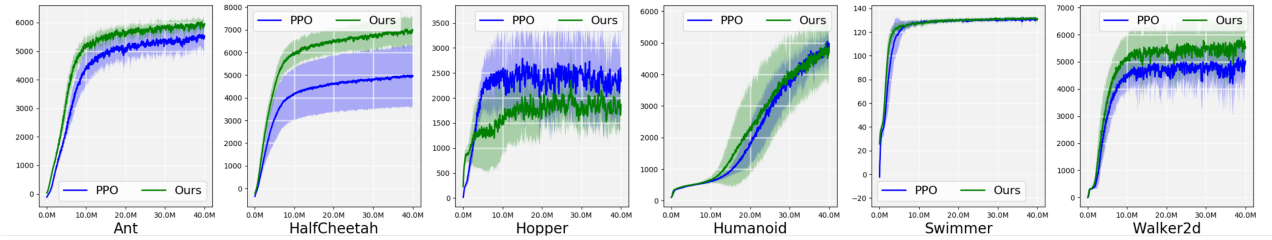

Fig. 3. Learning curve of RL with PPO and LfD training(ours). X-axis describes the interaction steps; y-axis shows the cumulative episode rewards.

Table 1. Time steps to reach the same level of performance (ratio between the time steps of our approach and PPO). Each row represents the time needed for our method to reach $x\%$ of the maximum rewards PPO gets.

	Ant-v2	**HalfCheetah-v2**	**Hopper-v2**
80%	0.71	0.54	5.27
50%	0.77	0.69	1.50
30%	0.78	0.70	0.13
	Humanoid-v2	**Swimmer-v2**	**Walker2d-v2**
80%	0.95	0.70	0.79
50%	0.92	0.74	0.87
30%	0.87	0.14	0.81

gain similar performance, what manifests better sample efficiency. Table 1 also shows that our algorithm can boost sample efficiency by 30% except for Hopper-v2, which might suffer from over-fitting. Especially at early training stages, the efficiency gain is even stronger. At this stage, training in Hopper-v2 benefits 87% of improvement in terms of sample efficiency. From Fig. 3, we see that demonstration can boost performance in terms of sample efficiency and final reward.

Table 2, Table 3 along with Fig. 4 compare mean episode rewards from early training stages and after the algorithm's convergence. Thanks to our approach, the model *warms up* through the expert demonstration dataset and thus, the performance at the very beginning of training is improved. In the meanwhile, we also improve RL's final performance with the help of the demonstration dataset. The *know-how* is distilled from the demonstration dataset and internalized by the PPO model. The model acquires performance that can not be achieved by simple exploration. Therefore, the expert demonstration can guide the original model out of sub-optima and so achieve better performance.

Table 2. Initial rewards per episode.

	Ant-v2	**HalfCheetah-v2**	**Hopper-v2**
PPO	-112 ± 14.0	-346 ± 53.3	14.1 ± 5.48
Ours	35.1 ± 20.4	-215 ± 80.9	243 ± 95.1
	Humanoid-v2	**Swimmer-v2**	**Walker2d-v2**
PPO	106 ± 8.03	-2.14 ± 17.4	1.59 ± 2.50
Ours	109 ± 7.12	25.0 ± 1.72	7.92 ± 7.41

In summary, we boost the performance of the PPO model on three aspects:

(1) Rewards at initial stage of training are raised.
(2) The final episode rewards becomes higher.
(3) Stability and sample efficiency become stronger.

Table 3. Final rewards per episode.

	Ant-v2	**HalfCheetah-v2**	**Hopper-v2**
PPO	5682 ± 356	4648 ± 1749	2785 ± 708
Ours	5891 ± 130	7027 ± 611	2362 ± 604
	Humanoid-v2	**Swimmer-v2**	**Walker2d-v2**
PPO	5193 ± 463	132 ± 1.3	5203 ± 1143
Ours	4985 ± 806	132 ± 0.7	6296 ± 1136

4.2 Ablation study

In our ablation study, we investigate the question: how the demonstration process can influence the initial episode rewards and the final rewards? We test our method in the *Ant-v2* environment since the training both with PPO and our method presents stable behaviors.

Table 4. Influence of pretraining epochs on episode rewards at different RL training stages in the Ant-v2 environment. Results over 5 runs using different random seeds

	20 Eps	**50 Eps**	**100 Eps**
Early Reward	35.1 ± 20.4	319 ± 102	463 ± 122
Final Reward	5891 ± 130	$\mathbf{5916 \pm 157}$	3835 ± 89
	200 Eps	**500 Eps**	
Early Reward	581 ± 73	$\mathbf{623 \pm 77}$	
Final Reward	3004 ± 212	1879 ± 47	

Table 4 shows that increasing the number of pre-training episodes can enhance the model's initial performance. Demonstrations before the actual RL training can help the agent to imitate the expert behavior, leading to higher rewards at early training stages. Nevertheless, excessive pre-training leads to overfitting. This overfitting problem traps the RL algorithm within a local optimum, which restricts the rewards at a low level.

Experiments demonstrate the fact that the initial performance tends to be better with more pre-training, but the model may also overfit the demonstration dataset with too many demonstration epochs. The overfitting causes a 68% drop in terms of the final performance (Table 4). Hence, a trade-off exists between the pre-training epochs, the model initial and final performances. Although the model's performance is sensitive to the number of pretraining epochs, the choice of 20-50 epochs works robustly for all kinds of environments within our experimental scope.

Results in simulated robotic configuration We also inspect the performance of our approach in a simulated humanoid robotic scenario. The environment (Fig. 5) consists of a robot that aims to reach out the fruit in front of it. The robot model is a re-designed open-source InMooV

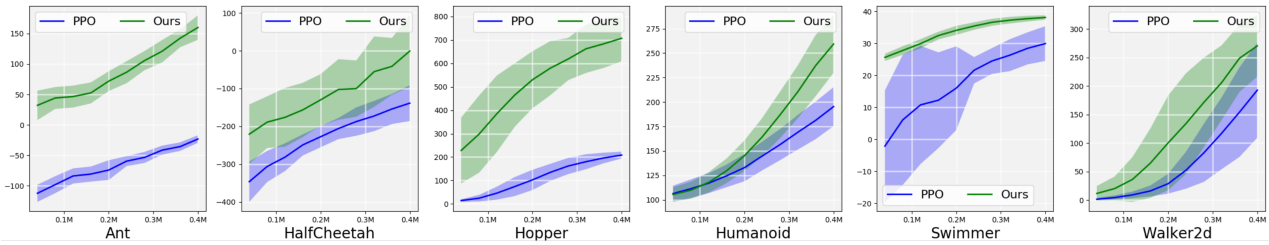

Fig. 4. Learning curve in 6 environments during first $0.4M$ time steps out of total $40M$. The figures represent a zoom-in for the Fig 3.

humanoid robot Langevin (2014)[1]. InMoov is a life-size robot with 87 joints structure that can be produced by 3D printing. We partly re-designed the robot structure for human-robot synchronized control system (Gong et al., 2017), of which 53 joints can be actively controlled. This robotic simulation is built based on PyBullet (Coumans and Bai, 2016–2019), an open source real-time physics simulation engine. And we developed the RL interface through the idea of Raffin et al. (2018).

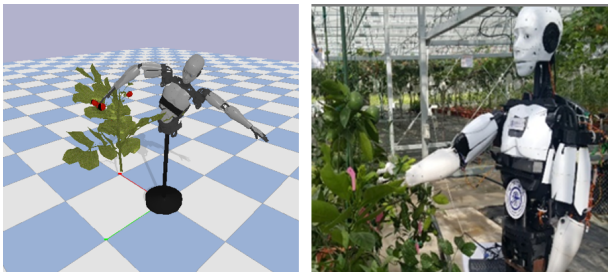

Fig. 5. Robotic fruit picking task in simulation and real life.

The state given by environment consists of 53 joint angles that are limited by the robot intrinsic functionality. The reward signal depends on the distance between the end of robot's index finger and the center of the fruit. The action generated by the model provides a control signal for all robot joints and is characterized by 53 dimensions as well. Therefore, states and actions are presented as a 53 dimension vector, and the reward is the minus distance to motivate robot to approach the target. Thus, the problem can be reformulated as:

$$s_t \in \mathbb{R}^{53},\ a_t \in \mathbb{R}^{53},\ r_t = -d(finger, fruit) \in \mathbb{R} \qquad (7)$$

In our robotic simulation configuration, the robot has to figure out the dynamics mechanism beneath the states, actions and the rewards, which involves a high dimensional inverse problem for robotic control. The reward signal only provides information about the distance between the robot and the target. This converts the problem in a "learning to coordinate" task.

For the experiments in this part, we firstly obtain our expert model by training a well converged PPO model. We reload the model that is trained from demonstrations and save the replay experiences into the demonstration dataset. Implementation details and hyper-parameters, pretraining epochs and network structure, are the same as described in Section 4.1.1.

[1] `https://inmoov.fr` Open source 3D printed life-size robot

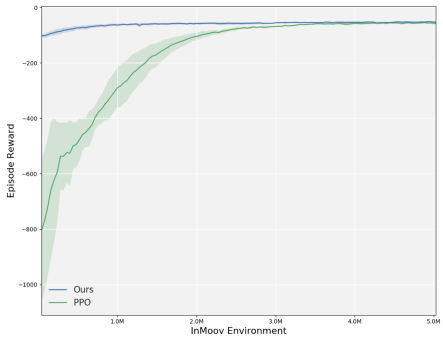

Fig. 6. Learning curve for RL in the InMoov environment

From Fig. 6, LfD training exhibits both strong stability and sample efficiency. This phenomenon consolidates our conclusion in previous sections.

5. CONCLUSION

To summarize our contributions, we present a novel pipeline for learning from demonstration based actor-critic RL models. In six standard Mujoco locomotion simulation environments , our method could improve agent learning speed in a more stable manner and achieve better sample efficiency in complex behaviors. The demonstrations can guide the model towards better performance both at early stage and final stage of training. We also perform experiments on *InMoov* open source robot , which shows that our method can as well be effective in the real life setting.

In terms of future work, we would like to test the performance of our demonstration strategy when the environment states consist of raw pixels, both in end to end approaches and state representation learning Lesort et al. (2018). Implementation of demonstration learning across other actor-critic RL algorithms could also be helpful to boost the current performance.

6. ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China (Grant No.51775333), the Science Foundation of Shanghai Municipal Commission of Science and Technology (Grant No.18391901000) and National Key Research and Development Project (2018YFB1305104).

REFERENCES

Argall, B.D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5), 469–483.

Arulkumaran, K., Deisenroth, M.P., Brundage, M., and Bharath, A.A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731), 34–37.

Bennetot, A., Charisi, V., and Díaz-Rodríguez, N. (2020). Should artificial agents ask for help in human-robot collaborative problem-solving? In *ICRA Brain-PIL Workshop - New advances in brain-inspired perception, interaction and learning*. arXiv preprint arXiv:2006.00882. URL `https://brain-pil.github.io/icra2020/`.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Coumans, E. and Bai, Y. (2016–2019). Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

Cruz, G.V.D.L., Du, Y., and Taylor, M.E. (2017). Pre-training neural networks with human demonstrations for deep reinforcement learning. *CoRR*, abs/1709.04083. URL `http://arxiv.org/abs/1709.04083`.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477. URL `http://arxiv.org/abs/1802.09477`.

Gong, L., Gong, C., Ma, Z., Zhao, L., Wang, Z., Li, X., Jing, X., Yang, H., and Liu, C. (2017). Real-time human-in-the-loop remote control for a life-size traffic police robot with multiple augmented reality aided display terminals. In *2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, 420–425.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905. URL `http://arxiv.org/abs/1812.05905`.

Hester, T., Vecerík, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J.Z., and Gruslys, A. (2017). Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732. URL `http://arxiv.org/abs/1704.03732`.

Konda, V.R. and Tsitsiklis, J.N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. (2012). Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3), 360–375.

Langevin, G. (2014). Inmoov. open source 3d printed robot platform. `www.inmoov.fr/project`.

Lesort, T., Díaz-Rodríguez, N., Goudou, J.F., and Filliat, D. (2018). State representation learning for control: An overview. *Neural Networks*, 108, 379 – 392. doi: https://doi.org/10.1016/j.neunet.2018.07.006.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *Computer Science*, 8(6), A187.

Lin, K., Gong, L., Li, X., Sun, T., Chen, B., Liu, C., Zhang, Z., Pu, J., and Zhang, J. (2020). Exploration-efficient deep reinforcement learning with demonstration guidance for robot control. *arXiv preprint arXiv:2002.12089*.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783. URL `http://arxiv.org/abs/1602.01783`.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2017). Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089. URL `http://arxiv.org/abs/1709.10089`.

Nguyen, S.M., Baranes, A., and Oudeyer, P.Y. (2011). Bootstrapping intrinsically motivated learning with human demonstrations. *arXiv preprint arXiv:1112.1937*.

Parisi, S., Tateo, D., Hensel, M., D'Eramo, C., Peters, J., and Pajarinen, J. (2020). Long-term visitation value for deep exploration in sparse reward reinforcement learning. *ArXiv*, abs/2001.00119.

Raffin, A., Hill, A., Traoré, R., Lesort, T., Díaz-Rodríguez, N., and Filliat, D. (2018). S-RL Toolbox: Environments, Datasets and Evaluation Metrics for State Representation Learning. *arXiv preprint arXiv:1809.09369*.

Safavi, A. and Zadeh, M.H. (2017). Teaching the user by learning from the user: personalizing movement control in physical human-robot interaction. *IEEE/CAA Journal of Automatica Sinica*, 4(4), 704–713.

Schaal, S. (1997). Learning from demonstration. In *Advances in neural information processing systems*, 1040–1046.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347. URL `http://arxiv.org/abs/1707.06347`.

Sutton, R.S., Barto, A.G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033.

Traoré, R., Caselles-Dupré, H., Lesort, T., Sun, T., Cai, G., Díaz-Rodríguez, N., and Filliat, D. (2019). DISCORL: Continual reinforcement learning via policy distillation. *arXiv preprint arXiv:1907.05855*.

van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461. URL `http://arxiv.org/abs/1509.06461`.

Vasan, G. and Pilarski, P. (2017). Learning from demonstration: Teaching a myoelectric prosthesis with an intact limb via reinforcement learning. *IEEE, International Conference on Rehabilitation Robotics :*, 2017, 1457–1464. doi:10.1109/ICORR.2017.8009453.

Vecerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and

Riedmiller, M.A. (2017). Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *CoRR*, abs/1707.08817. URL `http://arxiv.org/abs/1707.08817`.

Watkins, C.J.C.H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4), 279–292.

Xiang, G. and Su, J. (2019). Task-oriented deep reinforcement learning for robotic skill acquisition and control. *IEEE Transactions on Cybernetics*, PP, 1–14. doi:10.1109/TCYB.2019.2949596.

Zhao, D., Xia, Z., and Zhang, Q. (2017). Model-free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [research frontier]. *IEEE Computational Intelligence Magazine*, 12(2), 56–69.

Zuo, S., Wang, Z., Zhu, X., and Ou, Y. (2017). Continuous reinforcement learning from human demonstrations with integrated experience replay for autonomous driving. doi:10.1109/ROBIO.2017.8324787.

Appendix A. HYPERPARAMETER TABLE

Table A.1. Hyperparameters for our PPO implementation and RL from demostration.

Hyperparameter	Value
Policy learning rate	3×10^{-4}
Value function learning rate	10^{-3}
γ	0.99
λ	0.97
Clip ratio (ϵ)	0.2
Target KL to clip	0.01
Update interval	80