Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 198 (2022) 472–479

Procedia Computer Science

www.elsevier.com/locate/procedia

International Workshop on Internet of Smart Things (IST2021)
November 1-4, 2021, Leuven, Belgium

Swarm Deep Reinforcement Learning for Robotic Manipulation

Xudong Zhu[a,*], Fan Zhang[a], Hui Li[b]

[a] *Xi'an University of Architecture and Technology (XAUAT), No. 13 Yanta Road, Xi'an and 710055, P.R. China*
[b] *Xidan University, No. 2 South Taibai Road, Xi'an and 710071, P.R. China*

Abstract

Deep reinforcement learning scheme, which combines both deep learning and reinforcement learning, enables robots to learn from exploration and flexibly performance in a range of different operational tasks under highly dynamic and complex environments encountered in daily life. However, robotic manipulation still face many serious threats due to inadequate data sharing between robots and concerns about data privacy and security. To privacy-protect the data of all owners, we propose a swarm reinforcement learning method, a decentralized deep reinforcement learning technology based on block chain. Specifically, each robotic agent controls the robot using actor-critic strategy optimization algorithm, and shares their learning experience (i.e. loss function gradient) through the blockchain network, and passes on a mature strategy model parameters to other agents. Experimental results indicate that our swarm reinforcement learning method can improve the learning process of several agents, and the more agents there are, the faster the learning speed will be.

Keywords: Robotic Manipulation; Deep Reinforcement Learning; Blockchain; Swarm Learning;

1. Introduction

Robots have been successfully applied in industrial environments to perform a variety of static and simple operational tasks. However, robots still face many challenges for the dynamic and complex tasks they often encounter in daily life [1]. Reinforcement Learning (RL) agents can give robots the ability to learn from exploration and flexibly perform a range of different manipulation tasks. In recent years, deep reinforcement learning has made amazing success in the field of machine control, such as excellent performance in Atari game [2], defeating the World Champion of Go[3].

In traditional DRL schema (see Fig. 1a), a sole robot agent only interacts with its local robot, the amount of data collected is limited, so it is difficult to quickly generate enough accurate deep reinforcement model, which is difficult to

* Corresponding author. Tel.: +80-029-8375-7896 ; fax: +80-029-8375-7896.
E-mail address: zhudongxu@vip.sina.com

1877-0509
10.1016/j.procs.2021.12.272

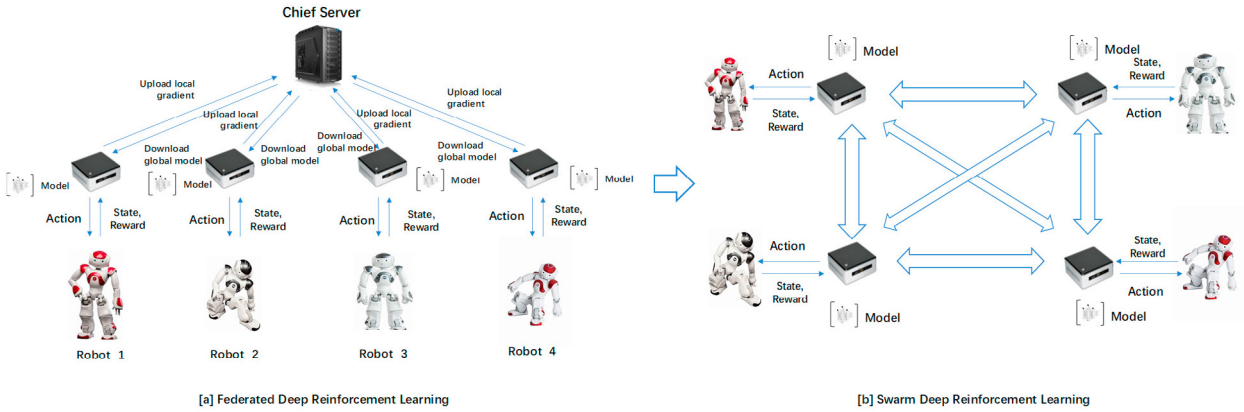

Fig. 1. Federated Deep Reinforcement Learning vs. Swarm Deep Reinforcement Learning.

be applied to real-time control of physical robots. To control several robots optimally, Federated Deep Reinforcement Learning (FDRL) has been proposed [11], in which DRL agents can not only control their own local robots, but also cooperate with other agents to control other robots, so as to learn their optimal control strategy more quickly. Federated learning can speed up the reinforcement learning process and reduce the instability of several robot training. FDRL is a distributed learning model which includes the robotic agents and the parameter server. The robot agent updates the local reinforcement learning model, trains its local models according to local private data and uploads its gradients. The parameter server aggregates these gradients to form a new global model.

However, since a central parameter server is still needed, FDRL still face some issues. Firstly, a centralized parameter server may form a single point of failure. Secondly, due to the need to aggregate the model gradient and broadcast the new model, the centralized parameter server needs high network bandwidth. Thirdly, the ”honest and curious” parameter server can collect the privacy data of the robot only through the model gradient. Finally, malicious parameter services can favor or ignore some robots to poison the global model.

In the article, we propose a novel Swarm Deep Reinforcement Learning (SDRL) – a decentralized, autonomous federated deep reinforcement learning approach based on blockchain – for controlling same type of multiple robots to solve these challenges (see Figure 1b). Specifically, the agent independently controls its local robot by Actor-Critic PPO method. And share model parameters with other agents through blockchain; Other agents use maturity model parameters to speed up their learning process. Our contributions are summarized below:

- First, we propose group deep reinforcement Learning (SDRL), a collaborative deep reinforcement learning framework based on blockchain, which allows agents to participate in deep reinforcement model training and control their own robots to solve potential problems such as single points of failure in traditional federated reinforcement learning systems.
- Second, we implemented a prototype of SDRL based on the Ethereum (V2.0) platform, demonstrating that SDRL provides a fully transparent distributed deep learning process with limited performance and network overhead when training multiple robots on control strategies to accelerate the learning process overall.

The rest of the paper is organized as follows. Firstly, we review Federated Learning (FL) and Federal Reinforcement Learning (FRL) in section 2. In section 3, we propose the Swarm Deep Reinforcement Learning (SDRL), a blockchain-based federated deep reinforcement learning approach, followed by performance evaluation in section 4. At last, we draw our conclusions in Section 5.

2. Related Work

Shokri et al. [6] proposed a method of multi-party federal trainning an accurate model for a given goal. In particular, each agent can store model parameters to shared memory and update model parameters randomly according to its own local training. Federated learning (FL) approaches [7] can build shared knowledge among a group of agents while preserving privacy and security of agents. Lalitha et al. [8] first proposed a completely distributed federated learning scheme in which agents updates their model by aggregating its neighbor information. Harris et al. [9] proposed and implemented a decentralized federated learning scheme based on blockchain. However, this scheme requires participants to publish private training data to the blockchain. Stefanie et al. [10] later proposed Swarm Learning (SL) method, a full distributed deep learning that fuses edge computing, peer-to-peer network and blockchain.

Federal reinforcement learning (FRL) is first proposed in [11], which integrates reinforcement learning and federal learning. This study demonstrates that FRL using collective observations of the environment is superior to DQN scheme only using partial observations of the same environment. FRL has been used to automatic pilot[12], and each participant used the knowledge learned by others to conduct steering control actions in a variety of different environments. In the field of robotic manipulation, FRL enables robots to quickly adapt to new environments by utilizing the knowledge and experience of other robots[13]. However, Swarm Reinforcement Learning which combines swarm learning and reinforcement learning has not been studied. In particular, there is a lack of a concrete and practical framework for implementing swarm reinforcement learning in both research and industry.

3. Proposed Swarm Reinforcement Learning

In the section, we propose the Swarm Deep Reinforcement Learning (SDRL) scheme, a decentralized, blockchain-based and privacy-protected federal reinforcement learning framework, which is able to allow multiple agents to control their local robots.

3.1. System Overview

As shown in Fig. 2, the SDRL is a common decentralized reinforcement learning scheme. The system consists of N workers and their interconnected point-to-point network. Each worker includes a robotic agent and a environment, in which the agent trains its Actor-Critic models to optimally manipulate the environment. The independent environment is divided into physical space and cyberspace. The robot is in the physical space. The agent can observe and control the state of robots through the cyberspace.

Unlike traditional federated reinforcement learning (FRL), a blockchain network replaces the place of the centralized parameter server. Smart contracts (SC) driven by blockchain transactions implements the function of a FDL's centralized parameter server. Without the need for a central parameter server, workers train reinforcement learning models based on its local data and share its model gradients with other workers via blockchain, which stores the global model parameters and local model gradients. Meanwhile, workers compete to add block rights through consensus mechanism (e.g., PoW). And the winning worker selects the gradient and generates a new block. SDRL consists of the following three processes:

Blockchain Bootstrapping. SDRL participants connect with each other to build blockchain networks and agree on initial settings for the global model.

Local Training. Workers interact with the robot and compute local model updates according to the latest global model parameters. Then the gradients of local model is encapsulated into messages that are broadcast to all workers.

Global Model Aggregation. Workers gain authority to attach new blocks to the blockchain by computing the hash value backwards. If a worker wins the competition, the global gradient is calculated from the local gradient received in the memory pool by the aggregation method updating the model parameter $\mathbf{w}$. The newly created block containing the relevant information is added to the blockchain. By computing local gradient computation and aggregating global gradient repeatedly, SDRL learns the global model. SDRL is summarized in Algorithm 1, which is described in detail below.

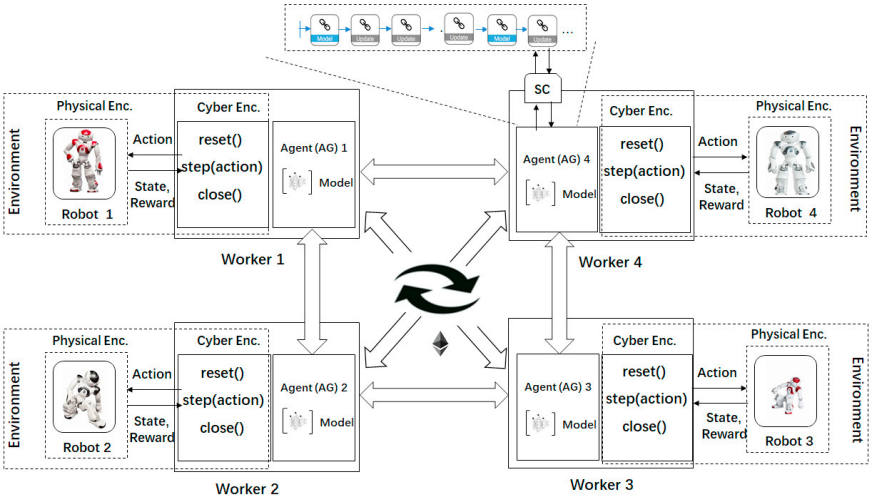

Fig. 2. The architecture for Swarm Deep Reinforcement Learning (SDRL).

3.2. Blockchain Bootstrapping

At this phase, we build a blockchain network and set initial parameters for training a deep reinforcement learning model. The phase consists of two steps:

Blockchain network setup. The workers are started and connected to each other to setup a blockchain network.

Initial parameter commitment. Before starting the iterative collaborative reinforcement learning, all workers need to agree on the initial values of the learning model parameters, like learning degree ξ, minimal batch size l, and model parameters $\mathbf{w}_0$. The worker which is randomly selected sets the initial value of the model parameter and broadcasts the parameter values to all workers.

3.3. Local Training

In this step, workers interact with the robot, calculate their local gradients, and propagate the gradients to all other workers. We now have the details as follows:

Pseudo-identity Creation. For protecting the worker's privacy in distributed learning framework, we use a pseudo-identity mechanism to generate pseudo-identities for workers. We denote the pseudo-indentity of worker C_k by C_k^*.

Local Training. Each worker C_k retrieves the current model $< g_\theta^{(t)}, g_\mu^{(t)} >$ from the latest block B_t on the chain. Each worker conducts individual training on an independent robot using the actor and critic PPO scheme.

The actor model π_θ has itself parameters θ. Based on the local environment the worker learns the Actor model which determines what actions to perform under the given state of robot. The local robot performs the actions predicted by the Actor model and returns the results to the worker. If achieving a positive response, the robot returned a positive reward to the worker, and vice versa. The critic model V_μ takes the reward and the parameter μ as input. The critic model learns to assess whether the action chosen by the actor model make robots in a good state and uses the feedback of the critic model to optimize the actor model.

For each round, each worker begins to interact with the local robot at $t = 0$, and ends the interaction with the local robot when $t = T$ or the end condition is met. In step t, agents receive environment's state s_t, choose to perform the action a_t, and in turn receive the reward r_{t+1} and the successor state s_{t+1}. In step t, the tuple $< s_t, a_t, r_{t+1}, s_{t+1} >$ is stored in the worker's trajectory memory as its experience.

The actor and critic PPO model can obtain a finite batch of experience tuples from the trajectory memory. In policy gradient reinforcement learning, the loss function F^P is defined as follows: $F^Q(\theta) = \widehat{E}[log\, \pi_\theta(a_t|s_t)\, \widehat{B}_t]$ where $\widehat{E}[\cdots]$ is the empirical expectation, and $\widehat{B}_t$ is the advantage function at step t. $\widehat{B}_t$ can be computed using a generalized advantage estimator (GAE) and the discount factor $\gamma \in [0, 1]$

Algorithm 1 Swarm Deep Reinforcement Learning

Blockchain Initialization:
(a) **Network Setup:** Workers $C_1, \cdots, C_M$ setup a P2P system.
(b) **Parameter Commitment:** Workers $C_1, \cdots, C_M$ set initial parameters ξ, l, and $< \overline{\theta}_0, \overline{g}_0 >$
Local Training:
for $1 \leq t \leq T$ and the model's performance did not reach the threshold **do**
  for $1 \leq k \leq M$ **do**
    C_k generates pseudo-identity C_k^*, and retrieves current $< \overline{g}_\theta^{(t)}, \overline{g}_\mu^{(t)} >$ from Block B_t
    Aggregate $\overline{g}_\theta^{(t)}$,$\overline{g}_\mu^{(t)}$ to set π_θ,V_μ by SGD scheme
    for $1 \leq t \leq T_{C_k}$ **do**
      Run π_θ for s_t and perform a_t
      Accept r_{t+1}, s_{t+1} from the local robot
      Save $< s_t, a_t, r_{t+1}, s_{t+1} >$ to experience memory.
    end for
    Update $\pi_{\theta_{old}} \leftarrow \pi_\theta$
    for $1 \leq j \leq K$ **do**
      Obtain the minimum batch B from experience memory.
      for $1 \leq t \leq U$ **do**
        Calculate $\widehat{B}_t$ by using the critic model V_μ and Equation (1)
        Get the value $V_\mu(s_t)$ and $\hat{V}_\mu^{target}(s_t)$
        Calculate $F_t^V(\mu)$ by using Equation (2) and $F_t^{CLIP}(\theta)$ by using the actor model π_θ and Equation (3)
      end for
      Calculate $g_\mu = \nabla F^V(\mu)$,$g_\theta = \nabla F^{CLIP}(\theta)$for average $(F_1^V(\mu), \cdots, F_U^V(\mu))$, $(F_1^{CLIP}(\theta), \cdots, F_U^{CLIP}(\theta))$ w.r.t. μ,θ
      Update V_μ,π_θ with g_μ,$g_\theta$through SGD
    end for
    C_k broadcasts the message $msg_k^t =< C_k^*, g_\theta, g_\mu, t >$
  end for
end for
Global Model Aggregation:
Worker C_j competes to calculate difficult math puzzles.
if C_j wins **then**
  C_j retrieves local gradients $(g_\theta^1, \cdots, g_\theta^m)$ and $g_\mu^1, \cdots, g_\mu^m$.
  C_j calculates the sum of all gradients using Equation $\overline{g}_\theta^{(t+1)} = \frac{1}{m}\sum_{i=1}^m g_\theta^i$ and $g_\mu^1, g_\mu^2, \cdots, g_\mu^m$
end if
C_j constructs the block $B_{t+1} = (< \overline{g}_\theta^{(t+1)}, \overline{g}_\mu^{(t+1)} >, [msg_{C_1}, \cdots, msg_{C_K}])$ and appends it to the main chain.

The generalized advantage estimator (GAE) is defined as:

$$\widehat{B}_t = \xi_t + (\gamma\alpha)\xi_{t+1}^V + (\gamma\alpha)^2\xi_{t+2}^V + \cdots + (\gamma\alpha)^{U-t+1}\xi_{U-1}^V \tag{1}$$

where $\alpha \in [0, 1]$ is the parameter, U is the batch size of samples, and $\xi_t = r_t + \gamma V_\mu(s_{t+1}) - V_\mu(s_t)$.

The loss function F^V is defined as:

$$F^V(\mu) = \widehat{E}[F_t^V(\mu)] = \widehat{E}[|\widehat{V}_\mu^{target}(s_t) - V_\mu(s_t)|] \tag{2}$$

where Time-Difference Error is defined as $\widehat{V}_{\mu}^{target}(s_t) = r_t + \gamma V_{\mu}(s_{t+1})$. The SGD algorithm updates the parameters of V_{μ} with the gradient $\triangledown F^V$: $\mu = \mu - \eta_{\mu} \triangledown F^V(\mu)$, where η_{μ} is the learning rate to optimize the critic model. By the importance sampling, the actor model calculates the average of samples from old policy $\pi_{\theta_{old}}$. The alternative loss function $F^{CPI}(\theta) = \widehat{E}[\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{B}]$ can be maximized.

Using the probability $R_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, we define the loss function of PPO F^{CLIP} as

$$F^{CPI}(\theta) = \widehat{E}[F_t^V(\mu)] = \widehat{E}[min(R_t(\theta), clip(R_t(\theta), 1-\delta, 1+\delta))\widehat{B}] \tag{3}$$

where δ is the shard parameter. The SGD algorithm updates the parameters of π_{θ} using the gradient $\triangledown F^{CLIP}$ for the negative of the shard loss function: $\theta = \theta - \eta_{\theta} \triangledown F^{CLIP}(\theta)$ where η_{θ} is the learning rate to optimize the actor model.

3.4. Global Model Aggregation

The local gradients which are broadcasted by workers are collected and stored in workers' memory pool. All workers compete for the authority to append new blocks to the blockchain through a consensus mechanism. The winner aggregates the local gradients to update the global model, and then uploads the new block to the blockchain. The phase consists of three steps:

Consensus Mechanism. All the workers in the P2P network copy and hold the blockchain. But only one worker is allowed to add new blocks to the blockchain at a time. In SDRL, we take advantage of PoW consensus protocol. The nonce will be incremented to the block until one worker C_j finds a value when the hash of the block meet requirements. And then the winner C_j can append a new block to the main chain and update the global model.

Gradient Aggregation. At the iteration t, the winner C_j aggregates all local gradients in itself memory pool correctly: $\overline{g}_{\theta}^{(t+1)} = \frac{1}{m}\sum_{i=1}^{m} g_{\theta}^{i}$, $\overline{g}_{\mu}^{(t+1)} = \frac{1}{m}\sum_{i=1}^{m} g_{\mu}^{i}$.

Block Constructing. Finally, the new model parameters and all local gradients are written in a block and appended to the blockchain by C_j. This block contains the following contents: $B_{t+1} = (< \overline{g}_{\theta}^{(t+1)}, \overline{g}_{\mu}^{(t+1)} >, [msg_{C_1}, \cdots, msg_{C_K}])$

4. Evaluation

In the section, we implement the proposed SDRL scheme and deploy it to real robots to evaluate its performance. The experiment proves the effectiveness of model knowledge sharing on reinforcement learning, and also verifies the influence of the number of workers on SDRL.

4.1. Evaluation Environment

To evaluate the SDRL performance, we set up an experimental system for controlling real robots. The experimental system includes three workers, and each worker controls its local robot. Each worker takes observed robot's status through a camera, and transfers them to the worker's agent. The agent implements the Actor-Critic PPO scheme by Python 3.4 and PyTorch 1.2. The multilayer perceptron used by the Actor-Critic model contains 3 hidden layer and 2 output layers. Each hidden layers contains 128 neurons. The Actor-Critic model parameters are set, in which shard parameter is set to 0.9, Adam optimizer is chosen as model optimization, GAE parameter is set to 0.99, learning rates is set to 0.001, experience memory size is set to 400 and size of the mini-batch is set to 128. In our evaluation, Ethereum Smart Contracts is used to implement the gradient aggregation scheme.

4.2. Effect Evaluation

And we utilized three OpenAI Gym's MoJoCo robot simulators simultaneously to simulate three completely independent robots. Each MoJoCo's 2-D simulator is actually a video game that controls a three-jointed robotic arm to a specific location. The arm of robot includes 3 joints and 4 links, in which joins constraints and the specifications of Boxter arm are satisfied. Observing the arm of robot through a camera with horizontal perspective, the play can issue

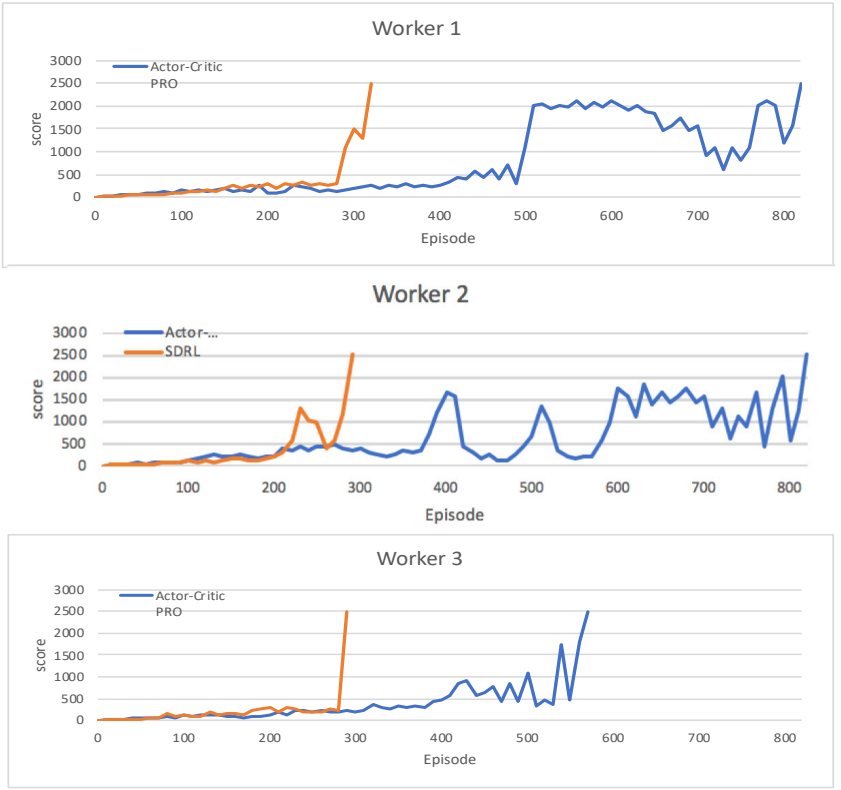

Fig. 3. Effectiveness of the proposed swarm deep reinforcement learning scheme.

specific commands to the joints to control the arm. So the simulator outputs the raw pixel image representing the state of the arm and has 9 action options, that is, 3 options for each joint: increase, decrease and maintain joint angle. The action reward is calculated according to the distance between the end of the robot arm and the target. If the distance becomes smaller, the reward value is set to 1; If becomes larger, is set to -1; Otherwise is set to 0 . When the total of three consecutive awards is less than -1, the whole process will be terminated.

In Fig. 3, we illustrate the proposed swarm deep reinforcement learning scheme is effective. The blue curve indicates the variation of actor model's score when workers perform the learning process alone, while the golden curve indicates the variation of the score when SDRL scheme is applied. When the SDRL scheme is used, all learning processes end at about 300 episodes. However, if worker 1, 2 and 3 perform the learning task alone, the overall process of worker 1 and 2 end at about 820 episodes and worker 3 ends at 570 episodes. Therefore, when the SDRL scheme is used, the learning speed of each robot gets much higher and the change of learning time is reduced. It is also found that in our experimental environment, the performance of SDRL will improve as the number of workers increases (i.e., from 1 to 8). Therefore, with SDRL, the quality of robotic manipulation can be greatly improved.

5. Conclusions

In the paper, we proposed a novel swarm deep reinforcement learning schema for robot control, called SDRL. With the help of blockchain smart contract, the centralized parameter server of Federated learning is no longer needed. Integrating swarm learning and reinforcement learning, we propose a population reinforcement learning scheme, in which reinforcement learning adopts the popular actor critical PPO model. The results prove that the proposed swarm reinforcement learning scheme can be successfully applied to the field of robot manipulation, and improve the speed

and accuracy of the model. The results also show that with the increase of the number of workers, the learning speed further increases. Compared to two workers, the learning speed of three workers increased by about 32%.

Acknowledgements

This work was supported by National Key Research and Development Program of China (2019YFD1100901), Natural Science Foundation of Shaanxi (Grant No. 2013JM8022).

References

[1] Zhang, F., Leitner, J., Milford, M., Upcroft, B., and Corke, P. (2015), "Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control, *arXiv:1511.03791*.
[2] Mnih, V., et al. (2015), "Human-level control through deep reinforcement learning", *Nature* **518** (7540): 529-533.
[3] Silver, D., et al. (2016), "Mastering the game of go with deep neural networks and tree search, *Nature* **529**(2016): 484-489.
[4] Zhuo, H.H., Feng, W., Xu, Q., Yang, Q., and Lin, Y. (2019), "Federated Reinforcement Learning", *ArXiv:1901.08277*.
[5] Crosby, M., Pattanayak, P., Verma, S., and Kalyanaraman, V.(2016), "Blockchain technology: Beyond bitcoin, *Applied Innovation*, **2**(2016): 6-10.
[6] Shokri, R., and Shmatikov, V.(2015), "Privacy-preserving deep learning", in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321. ACM.
[7] McMahan, B., and Ramage, D. (2017), "Federated learning: Collaborative machine learning without centralized training data", *URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html*.
[8] Lalitha, A., Shekhar, S., Javidi, T., and Koushanfar, F. (2018), "Fully decentralized federated learning", *Proceedings of third workshop on Bayesian Deep Learning (NeurIPS)*.
[9] Harris, J. D. and Waggoner, B. (2019), "Decentralized & collaborative AI on blockchain", *arXiv preprint arXiv:1907.07247*.
[10] Warnat-Herresthal, S., Schultze, H., Shastry, K.L. et al. (2021), "Swarm Learning for decentralized and confidential clinical machine learning", *Nature*, **594**(2021), 265–270.
[11] Zhuo, H.H., Feng, W., Xu, Q., Yang, Q., and Lin, Y. (2019), "Federated Reinforcement Learning", *ArXiv:1901.08277*.
[12] Liang, X., Liu, Y., Chen, T., Liu, M., and Yang, Q. (2019), "Federated Transfer Reinforcement Learning for Autonomous Driving", *ArXiv:1910.06001*.
[13] Liu, B., Wang, L., and Liu, M. (2019), "Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems", *IEEE Rob. Autom Lett*, **4**(2019), 45554562.