Server-Client Collaborative Distillation for Federated Reinforcement Learning

WEIMING MAI, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China
JIANGCHAO YAO*, CMIC, Shanghai Jiao Tong University & Shanghai AI Laboratory, China
CHEN GONG, School of CSE, Nanjing University of Science and Technology, China
YA ZHANG, CMIC, Shanghai Jiao Tong University & Shanghai AI Laboratory, China
YIU-MING CHEUNG, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China
BO HAN*, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China

Federated Learning (FL) learns a global model in a distributional manner, which does not require local clients to share private data. Such merit has drawn lots of attention in the interaction scenarios, where Federated Reinforcement Learning (FRL) emerges as a cross-field research direction focusing on the robust training of agents. Different from FL, the heterogeneity problem in FRL is more challenging, because the data depends on the policy of agents and the environment dynamics. FRL learns to interact under the non-stationary environment feedback, while the typical FL methods aim at handling the constant data heterogeneity. In this paper, we are among the first attempts to analyze the heterogeneity problem in FRL and propose an off-policy FRL framework. Specifically, a student-teacher-student model learning and fusion method, termed as *Server-Client Collaborative Distillation* (SCCD), is introduced. Unlike the traditional FL, we distill all local models on the server side for model fusion. To reduce the variance of the training, a local distillation is also conducted every time the agent receives the global model. Experimentally, we compare SCCD with a range of straightforward combinations between FL methods and RL. The results demonstrate that SCCD has a superior performance in four classical continuous control tasks with non-iid environments.

CCS Concepts: • **Computing methodologies** → *Cooperation and coordination*; *Reinforcement learning*.

Additional Key Words and Phrases: Federated learning, collaborative learning, heterogeneous environment.

ACM Reference Format:
WEIMING MAI, JIANGCHAO YAO, CHEN GONG, YA ZHANG, YIU-MING CHEUNG, and BO HAN. 2023. Server-Client Collaborative Distillation for Federated Reinforcement Learning. *ACM Trans. Knowl. Discov. Data.* 1, 1 (June 2023), 22 pages. https://doi.org/XXXXXXX.XXXXXXX

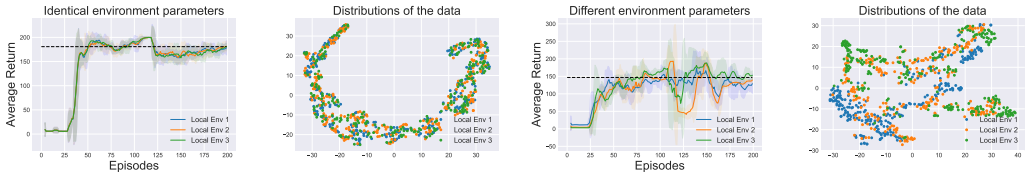
1 INTRODUCTION

Deep Reinforcement Learning (DRL) has achieved great success in video games and robotic control recently. However, DRL always requires lots of computation resources to master one specific

*Correspondence authors.

Authors' addresses: WEIMING MAI, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China, w.m.mai@tudelft.nl; JIANGCHAO YAO, CMIC, Shanghai Jiao Tong University & Shanghai AI Laboratory, Shanghai, China, Sunarker@sjtu.edu.cn; CHEN GONG, School of CSE, Nanjing University of Science and Technology, Nanjing, China, chen.gong@njust.edu.cn; YA ZHANG, CMIC, Shanghai Jiao Tong University & Shanghai AI Laboratory, Shanghai, China, ya_zhang@sjtu.edu.cn; YIU-MING CHEUNG, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China, ymc@comp.hkbu.edu.hk; BO HAN, Department of CS, Hong Kong Baptist University, Hong Kong SAR, China, bhanml@comp.hkbu.edu.hk.

1556-4681/2023/6-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX



(a) (b) (c) (d)

Fig. 1. A toy example to illustrate the affection of the environment heterogeneity. (a) shows the performance of FedAvg in three identical environments. The black dashed line indicates the average reward of the global model performed in each environment. Figure (b) shows the distribution of the sample data from each agent's local replay buffer after processing by t-SNE. The experiment of Figures (c) and (d) was conducted among three environments with different physical coefficients (e.g. the mass and length of the pole, the mass of the cart, and gravity). We can see that the learning curve and data distribution of the left column are more consistent. While there would be a distribution shift of the data in the right column because the policy of each agent is inconsistent.

task in the simulator, and there exists a huge gap between simulated environments and practical environments. Specifically, in many real-world scenarios, data privacy and the communication budget are usually the main concerns, which limit the application domain of DRL. For instance, in autonomous driving, when we leverage multi-agent techniques to train the vehicle, the agents need to share the observations with each other, which is usually disallowed due to the intrinsic privacy limitation. Fortunately, Federated Learning (FL) techniques offer users the ability to collaboratively train a machine learning model without frequently sending local models or gradients to a central server, thus preserving the privacy of their data. This has led to the development of Federated Reinforcement Learning (FRL). FL has been extensively studied and applied in various domains, including recommendation systems [45–47] and transportation [48], among others. Furthermore, the application of FL in DRL for decision-making and industrial robotic control [36] shows great promise and is an appealing direction to explore.

Recently, numerous techniques have been proposed in FL to speed up distributed training and tackle the issue of data heterogeneity. For example, Federated Averaging (FedAvg) [5] intends to increase the number of local updates to reduce the computational cost. However, it usually achieves poor performance when the local data distribution is heterogeneous. [14, 18] have explored the possibility of using reinforcement learning (RL) to optimize FL frameworks and facilitate edge computing. Meanwhile, others have proposed the addition of regularization to the objective function. For example, FedProx [23] deals with this issue with a l_2-norm regularizer, which minimizes the distance between the local and the global models in the local updates. However, FedProx has a lower convergence rate compared to FedAvg. Another approach is MOON [22] utilizes the similarity model representations to correct the training on the local database and get a good performance on image datasets. FedDF [27] and FedMD [21] perform the knowledge distillation to exchange knowledge between server and client-side through the public dataset and demonstrate the model distillation is superior to the naive model averaging in terms of highly heterogeneous local data. Nevertheless, the above methods are not suitable for RL due to two reasons: 1) there is no public dataset for RL agents to make use of for distillation; 2) their methods both utilize the average logits that cannot be exploited in the task with continuous action space.

On the other side, placing RL into the FL scenarios raises the intrinsic challenge from the non-stationary environments. In the conventional simulation of RL, we always assume the training

environment of the RL agent is identical to the testing environment, and there exists a unique state transition function across different environments. However, this assumption is not always satisfied in real-world FL applications. For example, the local agents in autonomous driving may face different weather conditions and different agents may have heterogeneous equipment. This kind of heterogeneity in FRL would be different from the objective heterogeneity in supervised FL. Hence, a more efficient and robust model fusion scheme is required to handle the non-stationary problem in FRL. The formal analysis would be illustrated in Section 3. Figure 1 shows the performance of FedAvg with three clients in identical local environments *CartPole* and the non-identical ones. Apparently, the traditional model averaging has a performance degradation in terms of the heterogeneous environment.

Although several FRL works [7, 28, 43, 49] have considered training the agents federally without sharing the data, most of them study the data heterogeneity problem caused by the local agent's policy and rarely consider the different dynamics (*i.e.*, local environments could have varied transition functions) of the local environments, which could bring the objective heterogeneity [31] issue into the training. To address the aforementioned objective

The early explorations in FL cannot always perform well in the reinforcement learning setting. As long as the value network is trained on the local data, its estimation on different environments would be unreliable and thus leading to poor performance of the actor network. Naively constraint the distribution between the local model and global model it's hard for the RL-based algorithm to converge in the early stage of training, resulting in a high communication cost issue.

In this paper, we proposed an FRL approach that can be aligned with traditional FL frameworks and study how to incorporate the FL techniques to deal with the environmental heterogeneity problem. Specifically, our goal is to train the agents locally with the aid of the server so that the agents perform well in similar environments but with different dynamics.

Inspired by the work in [30] and [27], we utilize the knowledge distillation for the model fusion, but unlike FedDF and FedMD, we make use of the representation generated from a Gaussian distribution for distillation [30] instead of the public dataset.

Besides, we conduct distillation in both the server and client side to reduce the problem of negative transfer [35] and remain feature extractor on the local side to obtain personalization. The role of the transmitted model between server and client would switch from both sides in order to deal with the dynamically changing heterogeneity. Specifically, we find such kind of iterative model distillation could be more efficient than the traditional model aggregation and hence the agents could learn better. Our contribution can be summarized as follows:

- We propose a federated learning framework for the off-policy DRL algorithm TD3 [13] named `FedTD3`. It is proved that it is equivalent to the centralized TD3 under some specific conditions. With this framework, the methods in traditional FL methods can be easily transferred to the FRL setting. Besides, this framework could be easily transferred to the other off-policy or actor-critic-based RL algorithm.
- We propose a model fusion method called Server-Client Collaborative Distillation. This method intends to solve the environment heterogeneity by distilling knowledge from the local critic's output layer so that it could have a lower communication workload while learning more local information. Besides, the critic network in each environment would have a specific feature extractor to better identify the distribution of the local data.
- We generate the non-identical environments for the local agents and explore how the policies of the local agents and the varied transition functions affect the federated training. The results show that SCCD could perform well even in some highly heterogeneous environments.

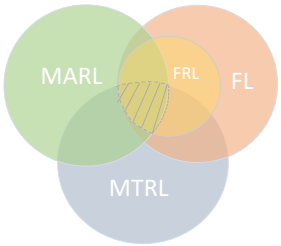


Fig. 2. The relationship between FL, FRL, MARL, and MTRL. Our work locates in their interaction zone.

2 RELATED WORK

2.1 Reinforcement Learning

Reinforcement learning is one of the basic machine learning paradigms. Generally, RL could be divided into off-policy reinforcement learning and on-policy reinforcement learning. The mainstream off-policy RL is developed based on the representative algorithm DQN [32], which was proposed in 2015 and achieved the human player level in the Arcade Learning Environment. After that, DDPG [26] and Twined Delayed DDPG [12] have been proposed to tackle the continuous control problem. The on-policy RL aims to update the agent's policy based on the real-time data collected from the environment and maximize its accumulated reward. REINFORCE [41] is a kind of on-policy RL based on the policy gradient methods. With this foundation, Konda et al. [19] proposed the famous Actor-Critic RL framework, which utilizes the neural network to estimate the value function. In this work, we mainly focus on the off-policy RL and choose TD3 as our FRL base algorithm because of its higher sampling efficiency.

2.2 Federated Learning

FL is a learning framework that requires multiple devices to perform training and simultaneously protects the privacy of the device. FedAvg [5] has been a practical federated learning approach. In FedAvg, the server sends the global model to the clients and then performs stochastic gradient descent to update the local models. After that, the local models are sent back to the server for aggregation. However, FedAvg is prone to falling into the local optima as the local optimums are far away from each other. FedProx [23] tries to deal with this issue by introducing a l_2 norm regularizer which limits the difference between the local model and the global model in the local updates. Karimireddy et al. [17] proposed Scaffold reduces the variance caused by the local updates and has a higher convergence rate than the former methods. MOON [22] utilizes the model representations to correct the training on the local database and achieve a good performance on image datasets. Luo et al. [30] proposed a post-calibration strategy CCVR to improve the classification performance by virtual representations. Other than performing the model aggregation, Lin et al. [27] presented the ensemble distillation (FedDF) to distill the knowledge from local models and increase the flexibility of the model's structure.

2.3 Knowledge Transfer MARL

Multi-Agent Reinforcement Learning (MARL) aims to organize multiple agents within a partially observable environment and to optimize the total reward. The parameters of each agent should be reserved and the goal of the agents maintains the same. Note that, we especially distinguish MARL from Multi-Task Reinforcement Learning (MTRL), which could be either multi-agent or single agent tries to learn a unique policy that could be adapted to different tasks. There have been some

Table 1. A summary of related works in FL and FRL. We compare our work with the others from the perspective of category, learning type, transmission, and whether consider the local personalization and non-stationary environment with different dynamics. The first subtable contains classic FL methods while the second subtable lists recent works of FRL. V and H denote the approach categorized as vertical FRL or horizontal FRL.

Methods	Category	Learning Type	Transmission	Personalization	Non-stationary Environment
FedAvg[5]	H	\	full network	×	\
FedProx[23]	H	\	full network	×	\
FedMD[21]	H	\	average logits	×	\
FedDF[27]	H	\	average logits	×	\
Dec-POMDP[34]	V	off-policy	original experience	×	×
FCRL[49]	V	off-policy	model outputs / full network	×	×
LFRL[28]	H	off-policy	score matrix / full network	×	✓
FTRL[25]	H	off-policy	scaled experience / full network	×	✓
MT-FedRL[2]	H	on-policy	full network	✓	×
FRD[7]	H	off/on-policy	proxy experience	×	×
FEMRL[43]	H	on-policy	fictitious experience / full network	×	×
SCCD (ours)	H	off-policy	statistic of representation / partial network	✓	✓

works that leverage knowledge distillation to transfer the knowledge between agents to accelerate the learning procedure of the agents. In [20], the authors introduced a student-student framework in which two agents explore in the same environment and share the knowledge with each other by policy distillation[37]. Another objective is to tackle the non-stationary environment, in which the environment dynamics are changing over time [1, 8, 29]. However, the works mentioned above don't consider protecting private data but just directly shares the observation of each agent. The following section would introduce the works that incorporate privacy protection into these two learning paradigms and relate them to federated reinforcement learning.

2.4 Federated Reinforcement Learning

In this section we will introduce Federated Reinforcement Learning (FRL) techniques that allow agents to learn collectively without sharing their collected data, ensuring privacy. In [36], FRL techniques are categorized as Horizontal Federated Reinforcement Learning (HFRL) and Vertical Federated Reinforcement Learning (VFRL) to maintain consistency with FL. In HFRL, the local environments of the agents are independent, which means that the behavior of the agent would not affect the other agents [4, 7, 33, 43]. Under this setting, Sherine et al. [4, 33] proposed to combine FL with RL from the perspective of game personalizing. The authors defined the personalized metric as the interaction between the human player and the agent, that is, the different players could have different skill levels. To protect the players' privacy and make a better gaming experience, they built up global models for each player group. Fan et al. [10] proposed the FedPG-BR framework by sending the local gradients to the server and introduced a filtering method to tackle the Byzantine General Problem. In [43], the authors have presented a model-based FRL framework based on TRPO [38] to learn the environment model (*i.e.* the transition function) in a decentralized manner. In their framework, each agent is trained according to the environment model from the server. Recently, Jin et al. [16] proposed two algorithms, QAvg and PAvg, which are federated extensions of Q-Learning and policy gradient, respectively. They exhaustively analyzed the convergence of these algorithms.

In VFRL, the agents explore in the same environment but the observation space is limited, they need to work with each other to complete the task. MARL would be more consistent with this

setting. Most other works [34, 40, 42] are related to this field. However, they intend to improve the collaboration of each agent and did not consider privacy protection. In the work of Zhou et al.[49], they proposed an FRL framework integrated with DQN and exploit the Gaussian differentials to encrypt the outputs of the value network. Since there is a central Q network to distill the knowledge of local models and the observation space of agents is limited, it can be considered as the classic VFRL.

In addition to the HFRL and VFRL taxonomies, some research in FRL overlaps with transfer learning, multi-task reinforcement learning, and meta reinforcement learning. For example, Liang et al.[25] present an online federated RL process that transfers both the local model and scaled local data to the server, tailored for car steering control. Anwar et al. [2] analyze adversarial attacks in multi-task RL where local environments have different action and observation spaces. Liu et al. [28] concurrently train agents to obtain a universe meta-model and enhance lifelong learning adaptability. FEMRL [43] aims to learn environment dynamic models federally and construct a federated model-based RL framework, but the real-world domain's dynamic model can be highly complicated and difficult to learn. Table 1 summarizes and compares these FL and FRL research works. Note that metric personalization considers whether the client model preserves partial original model information rather than being directly updated from the server model. Figure 2 depicts the relationship between different areas.

In our work, we adopted the off-policy model-free FRL framework and tries to train the Q network [32] from the previous experiences stored in the replay buffer which can be less affected by the data heterogeneity caused by the policies of different agents. Roughly speaking, the methods proposed in this paper can be categorized as the HFRL, but different from the work mentioned above. We introduce a slight perturbation to the local environment to accord with the non-stationary real-world environments. In our setting, the distribution of the data collected by the agent would be influenced by two factors: the agent's policy and the transition function. The details would be discussed in the following sections.

3 PRELIMINARY

3.1 Federated Learning

FL aims to train a global model without sharing the data of local devices. The major challenges are the data heterogeneity and the communication cost between servers and clients. Formally, the objective function of FL is defined as

$$\mathcal{L} = \min_{w} F(w) \triangleq \sum_{k=1}^{K} p_k F_k(w), \tag{1}$$

where F_k is the loss function over the local data in the k-th client, K is the client number. $p_k = n_k / \sum_{k=1}^{K} n_k$, where n_k is the sample batch size of client k. By increasing the local update iterations, FedAvg improves communication efficiency and simultaneously encourages convergence. FedProx [23] introduces a normalization for the loss function to calibrate the local training. In [44], Wang et al. proposed FedNova to normalize local model updates when averaging. Li et al. [22] proposed MOON which combines model-contrastive learning with the training procedure and outperforms the aforementioned FL algorithms.

3.2 Policy Distillation

Policy distillation is proposed to solve the knowledge transfer and model compression problems in reinforcement learning. It is also used in multi-task learning where expert policies can be combined into a single multi-task policy. Following the convention [37], $\mathcal{D}^{\mathrm{T}} = \{s_i\}_{i=0}^{N}$ is the set of observations

generated by the teacher model, where $\mathcal{N}$ is the number of the samples, the objective function can be defined as

$$\mathcal{J} = \mathbb{E}_{s\sim\mathcal{D}^{\mathrm{T}}}[D(Q_w(\cdot|s), Q_{\tilde{w}}(\cdot|s))], \tag{2}$$

where $D(\cdot|\cdot)$ is a proper distance measure of the output between the student Q-network $Q_w(\cdot|s)$ and the teacher Q-network $Q_{\tilde{w}}(\cdot|s)$, which can be the mean square error, KL divergence, or log-likelihood. In [20], the authors introduced a dual policy distillation framework for two independent agents explored in the same environment. The agents can perceive different aspects of the environment such that they can complement each other by sharing knowledge, which reveals the possibility of the knowledge transfer among multiple agents and acceleration of the learning process by policy distillation.

3.3 Non-IID Environments

The environments vary under different spatial and temporal characteristics. Therefore, the data collected in each environment is from a different distribution. Suppose $x = (s_t, a, r_t, s_{t+1})$ is the collected state action tuple, where r_t is the instant reward and s_{t+1} is the next state. Formally we have

$$\begin{aligned} P(x) &= P(s_t, a_t, r_t, s_{t+1}) \\ &= P(s_t) \underbrace{P_\phi(a_t|s_t)}_{\text{agent policy}} \underbrace{P_\theta(s_{t+1}|s_t, a_t)}_{\text{environment dynamic}} . \end{aligned} \tag{3}$$

From this equation, we can see the data collected from exploration is mainly affected by two factors, namely, the environment dynamic and the current policy of the agent. The local agents continuously learn from the replay buffer which contains the heterogeneous data collected by different policies, and thus the effect of the second term in Equation (3) can be minimized by off-policy RL. Hence, we mainly focus on how to handle the heterogeneous environment dynamic in FRL.

4 FEDERATED OFF-POLICY REINFORCEMENT LEARNING

4.1 Problem Statement

Similar to FL, the objective of FRL is to find the optimal value estimation function $Q_{w*}(s_t, a_t)$ over the state action pairs that draw from the local replay buffer of the agents. The local function $Q_{w*}(s_t, a_t)$ could be defined by the optimal Bellman equation:

$$Q_{w^*}(s_t, a_t) \triangleq \mathbb{E}_{S_{t+1}\sim p(\cdot|s_t,a_t)}[r_t + \gamma \max_{a\in A} Q_{w^*}(S_{t+1}, a)|S_t = s_t, A_t = a_t], \tag{4}$$

where r_t is the instant reward the agent received after taking action a_t at the current state s_t. Random variable S_{t+1} drawn from the state transition probability distribution function $p(s_{t+1}|s_t, a_t)$. The local objective function $F_k(w)$ in (1) is formulated as follows:

$$\begin{aligned} F_k(w) &= \frac{1}{n_k} \sum_j l(w; x_{kj}), \\ l(w; x_{kj}) &= \left(Q_w(s_t^{kj}, a_t^{kj}) - (r_t^{kj} + \gamma \max_a Q_w(s_{t+1}^{kj}, a))\right)^2 . \end{aligned}$$

Where the superscript kj denotes the jth sample collected from the kth agent. Supposing the amount of the exploration data n_k of each agent is all the same, the term $p_k = \frac{1}{K}$ can be eliminated. Therefore, the final objective in (1) can be rewritten as

$$\mathcal{L} = \mathbb{E}_{(s_t,s_{t+1})\sim D^+}[Q_w(s_t, a_t) - (r_t + \gamma \max_a Q_w(s_{t+1}, a))]^2, \tag{5}$$

where $D^+ = D_1 \cup \cdots \cup D_K$, the union of all local agent data.

ALGORITHM 1: FedTD3 (server-side)

Input: number of clients K, total communication round R
Output: Global value model $Q_{\bar{w}}$ and policy $\pi_{\bar{\phi}}$

Server Update:
Initialize Q-network parameters w^0, actor network parameters ϕ^0
for *r=1:R* **do**
  for *i=1:K* **do**
    send global model w_0, ϕ_0 to agent A_i
    $w_i^t, \phi_i^t \leftarrow \text{ClientUpdate}(w^t, \phi^t)$
  end
  $\bar{w}^{t+1}, \bar{\phi}^{t+1} = \text{aggregate}(w_i^t, \phi_i^t), i = 1, \ldots, K$
end

4.2 Federated Twin Delayed DDPG

In this section, we provide an instance FedTD3, a federated learning counterpart of TD3 which has been a state-of-the-art off-policy RL algorithm [12] by integrating the actor-critic diagram in the traditional DQN. TD3 is the benchmark off-policy RL algorithm to solve the continuous control task and compare to traditional DDPG, it is more stable and overcomes the value function overestimation issue.

In the framework of FedTD3, there is an individual state transition probability function $P_{\theta_k}(s_{t+1}|s_t, a_t)$ for each of the local environments. Besides, there are two types of communication in FedTD3, the transmission of the policy net and that of the Q-value net, between the server and the local agents. To maintain consistency with the single-agent TD3 framework, we have three processes to update the models: Firstly, the Q-value net does the local updates N times based on the current policy that is synchronized from the server and then sends it back to the server and waits for aggregation. Every M times updates of the Q-value network, conduct the delayed policy updates. Once the number of policy updates is achieved L, send the policy model to the server for aggregation. Both the target Q-value network and the target policy network are updated based on the model received from the server. Given the total exploration step T in each local update, the communication rounds are calculated by $R = T/N + T/(M \times L)$. Algorithm 1 and 2 present the detailed procedure of server update and client update in FedTD3. Each agent waits for the server for the new model after sending the local one in a synchronous manner. The *aggregate* function in Algorithm 1 adopts the naive weighted aggregation method for simplicity, and the loss functions could be constrained by different kinds of regular terms in order to deal with the heterogeneity, namely:

$$\begin{aligned} L(w_i^t) &= L_{\text{TD}}(w_i^t) + \beta l_{\text{reg}}(w_i^t, \bar{w}^t), \\ J(\phi_i^t) &= -\mathbb{E}[Q_{w_i^t}(s, \pi_{\phi_i^t}(s))] + \beta l_{\text{reg}}(\phi_i^t, \bar{\phi}^t). \end{aligned} \tag{6}$$

The first term $L_{\text{TD}}(w_i^t)$ is the TD loss function in Equation (5) and l_{reg} constrains the distance between the server model and the client models during the local training. In the baseline methods, the l_2 norm would be adopted in FedProx [23] and the model contrastive loss l_{con} is adopted in MOON [22] in the experiments. These terms can improve local training in a slightly heterogeneous environment, however, as the heterogeneity increase, the performance of these methods cannot be guaranteed. Moreover, empirically they are prone to slowing down the convergence rate in the RL setting. The results of the baselines will be shown in Section 5.

ALGORITHM 2: FedTD3 (client-side)

Input: number of local exploration T, communication frequency of value model N, communication frequency of policy model L, delayed policy updates frequency M

Output: Local model $Q_{w_i^t}$ and $\pi_{\phi_i^t}$

Client Update:
Synchronized w^t, ϕ^t from server
Set $count = 0$
for t=1:T **do**
  Explore in the local environment to get (s_t, a_t, r_t, s_{t+1}), save to replay buffer. Sample mini-batch data, compute gradient g_i^t with respect to w_i^t, update $Q_{w_i^t}$.
  if t *mod* N **then**
    send w_i^t to server.
    $\bar{w}^t \leftarrow$ ServerUpdate(w_i^t)
  end
  if t *mod* M **then**
    $\phi_i^t \leftarrow$ DelayedPolicyUpdate
    $count \leftarrow count + 1$
    if $count$ *mod* L **then**
      agent send ϕ_i^t to server.
      $\bar{\phi}^t \leftarrow$ ServerUpdate(ϕ_i^t)
    end
    agent updates target net ϕ_i^t based on local ϕ_i.
    agent updates target net w_i^t based on local w_i.
  end
end

4.3 Server-Side Distillation

Previous work [27] proposed knowledge distillation performed on the server side for the model fusion. However, they utilized the pre-trained GAN or unlabeled data for distillation, which is impractical in the RL setting as it requires the agent to interact with the environment to gather the data. In this section, we introduce our server-side distillation (SSD), which enables the local critic models to transfer their knowledge to the central student model, thus reducing the degradation caused by the previous model aggregation. On the other hand, we generate fictitious data from the statistic of the representations in the local replay buffer, and the global model can directly learn from these data, which allows the model to learn more information about the local critic and in the meantime protect the privacy. Figure 3 depicts the procedure of client-side distillation and the details of the server-side distillation.

Here, since the critic learned from all the environments is more reliable with the guidance of the critic, the actor can be trained more effectively. Therefore, we only focus on the value network distillation in this paper. Inspired by the concept in multi-task RL policy distillation [37], the Q network is constructed by a feature extractor $h_w(\cdot)$ and a predictor $f_w(\cdot)$. which is both implemented by the fully connected multi-layer perceptron (MLP). The extractor is to find a better representation of the input data. For the predictor, we send it to the server for model distillation and send it back to the agents. The choice of MLPs as function approximators is quite common in reinforcement learning [12, 39] due to their sample efficiency and ease of training. And also it is worth noting that for image data, a sophisticated CNN can also be used as the feature extractor. In order to obtain as much global information as possible, all the agents share the parameters of

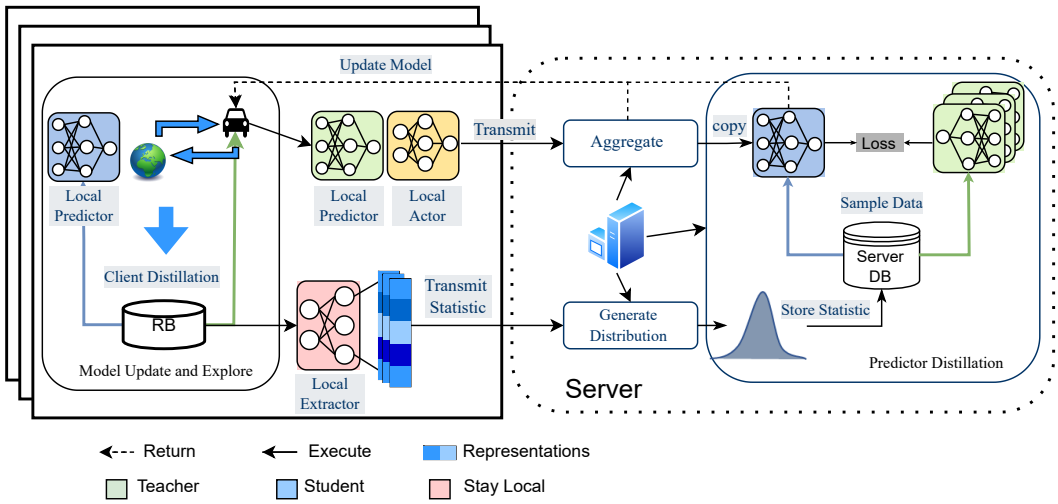


Fig. 3. The diagram of SCCD. The global server collects the actor parameters and the predictor parameters to do the model fusion. The representations of the local data are acquired on the local side, and the statistics are sent to the server for generating pseudo data. The local predictor would be fused as a single model based on the data generated by local statistics and sent back to the client. Next step on the client side, the local value models are updated via distillation from the global predictor.

the predictor of the value network $f_w(\cdot)$. The server-side distillation would usually conduct a few epochs, before the distillation we average the model parameters of the predictor to get a good initial parameter base. During each communication round, the local agent would send either the policy network or the partial value network to the server for model fusion.

For the data we use for the model distillation, their corresponding distribution from each local environment follows its own transition function $p_{\theta_k}(s_{t+1}|s_t, a_t)$. After extracting the representation of the observations, we calculate the statistic of the representation and send it back to the server for generating the pseudo data. To minimize the effect of the Gaussian distribution assumption, in each round, the local agents only calculate the statistic of a mini-batch of data, and the data are randomly sampled from the local replay buffer. Before conducting the model distillation, the server collects all the local model and do the aggregation. After obtaining the aggregated parameters, the model would do the distillation based on the parameters, in order to combine the knowledge of each critic in different environments.

Here, we term the partial value networks of the local agents as the teacher networks and the value network on the server side as the student network. We adopt the mean-square-error loss function $l(w; b)$ to train the server student network. Assume z_i is the representations of the i-th agent's input x, it can be obtained by $z_i = h_{w_i}(x)$. After getting z_i, the server utilizes its statistic to generate data as the input of the teacher network, such that we can get the distillation dataset $\mathcal{D} = \{(\tilde{z}_i, q_i)\}_{i=0}^{K}$, where each sample contains the pseudo representation $\tilde{z}_i$ and the output Q value q_i as the label. The network we use for the critic and actor are both MLP, while the number of layers of the critic network and actor network is different. One of the advantages of sending partial parameters of the critic network is to reduce the workload that is transferred to the server. In the RL setting, an MLP with few layers is sufficient to learn from the state and action pair. Theoretically, as long as the capacity of the predictor of the critic network is enough to learn from the representations, there is no need to send the whole Q network to the server [15].

4.4 Client-Side Distillation

In FedDF [27], the authors claim that only performing server-side distillation is sufficient to address the heterogeneity problems. Nevertheless, this cannot be guaranteed in a multi-task learning framework because of the issue of negative transfer [35]. Therefore, extra fine-tuning on the client side is necessary for local personalizing. In our method, from the server-side distillation, the global model can obtain knowledge from each local environment. After that, the server sends back the model to update the local model. Traditionally, the local model is directly updated from the global model, however in the early stage of training, the distilled global model is not completely reliable, and the update might delay the evolution of the critic network. Therefore in our method, two versions of the predictors would send back to the local clients: an aggregation one and a distillation one.

The local predictor would be updated based on the aggregated predictor and then distill the knowledge from the distillation predictor. To stabilize the local training process, the distillation loss function consists of two parts: the first part is the TD loss of the local replay buffer; the second part is the distillation loss function between the local Q network and the distilled global predictor:

$$\mathcal{L}(w_i, \bar{w}) = \alpha L_{\mathrm{TD}}(w_i) + (1-\alpha)\mathcal{J}(w_i, \tilde{w}), \tag{7}$$

where w_i, $\bar{w}$ are the parameters of the local critic network and server critic network. Loss function $\mathcal{J}$ can be referred to Equation (2). Note that the hyper-parameter α can be varied in different experiments. Algorithm 3 describes the procedure of SCCD, which is based on the FedTD3 framework in Algorithms 1 and 2 and only the model fusion and the transmission are changed. Besides, there is another way to conduct the client-side distillation in Equation (7), which is regarding the second term as the regularizer. This leads to no extra local update when the client side receives the model from the server if we choose the second type of distillation.

4.5 Theoretical Analysis

In this Section, we are going to analyze the convergence of our proposed method and compare it with FedAvg by exploring the upper bound. We begin with two standard assumptions:

Assumption 1. *There exists an optimal value function $Q_{w^*}(s,a)$ to estimate the value of the state-action pair from environments with heterogeneous transition dynamic.*

Assumption 2. *The local TD loss function F_k and server-side distillation loss $\mathcal{L}$ are L-smooth and μ-strongly convex.*

We can define the degree of heterogeneity Γ as follows:

$$\Gamma = \left| F_{w^*} - \sum_k p_k F_{w_n^*} \right|. \tag{8}$$

to describe the Non-IID characteristics of the environments, where F_{w^*} is the global minimum of the TD loss function over all the environments and $F_{w_n^*}$ is the local minimum of the n th environment. When Γ falls within an acceptable range, i.e., $0 < \Gamma < \epsilon$, Assumption 1 holds. It is worth noting that this index is commonly used to describe the Non-IID characteristics of environments [43]. Regarding Assumption 2, we follow the general spirit of [24], and consider both the TD loss and distillation loss functions as the mean square error, which satisfies the practical hypothesis on the loss space and is appropriate in designing SCCD. We now need to verify two key aspects:

- If Q_{w^*} exists, the local Q function Q_{w_n} could finally converges to Q_{w^*}.
- The global policy π_{glob} can be improved monotonically.

ALGORITHM 3: SCCD

Input: number of global distillation epochs N, local distillation iteration M, communication round R, number of agents K

Output: parameters of the global actor $\bar{\phi}$ and predictor $\bar{w}$

Initialize w^0, ϕ^0
for *r=1:R* **do**
  for *i=1:K* **do**
    send global model w_0, ϕ_0 to agent A_i
    $w_i^t, \phi_i^t, \mu_i, \sigma_i \leftarrow \text{ClientUpdate}(w^t, \phi^t)$
  end
  $\bar{w}^{t+1}, \phi^{t+1} = \text{aggregate}(w_i^t, \phi_i^t), i = 1, \ldots, K$
  generate data $\mathcal{D}$ based on μ_i and σ_i
  $\bar{w}^{t+1} \leftarrow \text{SSD}(\mathcal{D}, \bar{w}^{t+1})$
  clients conduct CSD based on $\bar{w}^{t+1}$.
  agents explore the local environments.
  sending local predictor w_i or actor ϕ_i to server for model fusion.
end
SSD:
for *each epoch from 1 to N* **do**
  for *each batch* $b \in \mathcal{D}$ **do**
    $L(\bar{w}; b) = \frac{1}{|b|} \sum_{\tilde{z} \in b} \|f_{\bar{w}}(\tilde{z}) - q_i)\|_2^2$
    $\bar{w} \leftarrow \bar{w} - \eta \nabla L(\bar{w}; b)$
  end
end
CSD:
for *j = 1:M* **do**
  Sample a batch of data b from the local replay buffer
  $\mathcal{L}(w_i, \bar{w}; b) = \alpha L_{\text{TD}}(w_i; b) + (1 - \alpha)\mathcal{J}(w_i, \bar{w}; b)$
  $w_i \leftarrow w_i - \eta \nabla \mathcal{L}(w_i, \bar{w}; b)$
end

Suppose w_n is the local parameters and w^* is the optimal global parameters. According to the Bellman Optimality Equation [3], in each round of local update, the local value function should be closer to the optimal value function, *i.e.,*

$$\left\| w_n^{t+E} - w^* \right\| \leq \left\| w_n^t - w^* \right\|. \tag{9}$$

In our method, both the global predictor and the actor network are the aggregation of the local models:

$$\begin{aligned} f^t &= \frac{1}{N} \sum_n f_n^t, \\ \pi_{glob} &= \frac{1}{N} \sum_n \pi_n^t, \end{aligned} \tag{10}$$

where f^t is the predictor on the server side and π_{glob} is the estimated global policy based on each local agent policy π_n. Following the proof in [24], the convergence of the actor model can be guaranteed by the following theorem.

Theorem 4.1. *If Assumtion 1 and 2 hold and the learning rate* $\eta \leq \frac{1}{4L}$*, we have the following convergence analysis for FedAvg:*

$$\mathbb{E}\left\|\bar{w}_{t+1} - w^*\right\|^2 \leq (1-\eta\mu)\mathbb{E}\left\|\bar{w}_t - w^*\right\|^2 + \eta^2 B. \tag{11}$$

Based on this theorem in [24], we compare the bound of FedAvg with server-side distillation in the following. Suppose $\tilde{w}_t$ is the distillation result of $\bar{w}_t$. If Assumption 2 holds, the distillation loss function is strongly convex, and the global minimum of the distillation loss $\mathcal{L}$ can be obtained by the sufficient update of SGD. Thus, $\tilde{w}_t$ can be obtained by $\tilde{w}_t = \bar{w}_t - \sum_{i=1}^{T} \alpha_i g^{(i)}$, where T is the total steps for SGD optimization, $g^{(i)}$ is the gradient over all the fictitious data at iteration step i and α_i is the learning rate. When $i = 1$, $g^{(1)} = \nabla\mathcal{L}(\bar{w}_t; \mathcal{D}_t)$, suppose the learning rate is fixed, we have $\| \sum_{i=1}^{T} \alpha g^{(i)} \| \leq \sum_{i=1}^{T} \alpha^2 \|g^{(i)}\| \leq \alpha^2 T \|g^{(1)}\|$. Then, by applying the Triangle inequality, we obtain the following upper bound for server-side distillation:

$$\begin{aligned}
\mathbb{E}\left\|\tilde{w}_t - w^*\right\|^2 &= \mathbb{E}\left\|\tilde{w}_t - \bar{w}_t + \bar{w}_t - w^*\right\|^2 \\
&\leq \mathbb{E}\left\|\bar{w}_t - w^*\right\|^2 + \mathbb{E}\left\|\tilde{w}_t - \bar{w}_t\right\|^2 \\
&\leq \mathbb{E}\left\|\bar{w}_t - w^*\right\|^2 + \mathbb{E}\left\|\sum_{i=1}^{T} \alpha_i g^{(i)}\right\|^2 \\
&\leq \underbrace{\mathbb{E}\left\|\bar{w}_t - w^*\right\|^2}_{\mathcal{B}} + \alpha^2 T \mathbb{E}\left\|\nabla\mathcal{L}(\bar{w}_t; \mathcal{D}_t)\right\|^2
\end{aligned} \tag{12}$$

Remark. The inequality above introduces $\mathcal{B}$ as the bound of FedAvg, the details can be found in the lemma 1 of [24]. The second term in the inequality is influenced by the distillation training steps T and the gradient of $\bar{w}_t$. This suggests that the upper bound of our algorithm would be affected by the shape of the distillation loss function and the distribution of the fictitious data. Ideally, as the value function of the agent progresses, the norm of the gradient in the second term could gradually decrease. This implies that with sufficient optimization to approach the optimum w^* on the server side, our method with server-side distillation can converge and be bounded similarly to FedAvg.

5 EXPERIMENTS

In this section, we first introduce the heterogeneous environments used in our federated reinforcement learning experiments and then compare the FRL methods that incorporate different FL methods to overcome the heterogeneity under the `FedTD3` framework [1]. Finally, we analyze the results and discuss the advantages and disadvantages of the current methods.

5.1 Experimental Setup

5.1.1 Dataset and Model Structure. In our experiments, we build a four-layer MLP with 256 hidden neurons and ReLU activation to represent the value network and the policy network respectively. During the training, the Adam optimizer is applied. The smoothing factor τ involved in the target network update is set at 0.01, noise clipping threshold is 0.5. Four environments developed by Open-AI are used in the experiments. The first two environments are the classic control task *CartPole* and *Pendulum* and the other two are *BipedalWalkerHardcore* and *LunarLander* [6]. In all experiments, we explore $K = 5$ agents for all algorithms. To fairly compare each method, the hyper-parameters e.g. the capacity of the replay buffer, and the distillation learning rate are shown in Table 2. The standard deviation of the noisy action keeps the same in each task. For the parameter

[1]The source codes are available at https://github.com/tmlr-group/SCCD

β of the regular term and penalty α in the client-side distillation loss function, we set them to be 0.01 and 0.1 in each task. The learning rate of the critic and actor is set to be 2e-3 for each task.

5.1.2 Baselines. We compare SCCD with the following baselines, where each method is conducted based on the `FedTD3` framework:

- FedAvg: Naively aggregate the local actor-critic models into a global model. The averaging weights are calculated based on the local training batch size.
- FedProx: Adding a l_2 norm to the local training objective.
- MOON: Leveraging the concept of contrastive learning [9] in federated learning and propose a new loss function that considers the global and local model parameters as a pair of samples. This loss function measures the contrast between the global and local models:

$$l_{\text{con}} = -\log \frac{\exp(z^\top z_{\text{glob}}/\tau)}{\exp(z^\top z_{\text{glob}})/\tau) + \exp(z^\top z_{\text{prev}}/\tau)},$$

  where z is the representation before the output layer of the network, z_{glob} is the representation of the aggregated global model, z_{prev} is the last round global model.
- Scaffold: Construct the variance reduction to correct the client shift in the local updates.
- FedDF: Ensemble the local networks as the teacher networks and leverage the synthesis data to conduct the model distillation.
- MTFRL: Similar to FedAvg, but there are two parameters to control the model averaging which make it more smooth and more stable.
- FRD: The local experiences are clustered and sent to the server to construct global proxy states, The agents receive the global proxy states and perform policy distillation.

5.1.3 Environment Heterogeneity. In *CartPole*, the pendulum starts upright and the goal is to prevent it from falling over. The agent gets a -1 score punishment once the pole cannot remain vertical and the angle of the pole is more than 15 degrees. We set the length of each episode as 300, so the maximum score would be no more than 300. In *Pendulum*, the pendulum starts in a random position and the goal is to swing it up so it could stay upright. The episode length is set to be 200, and the closer the score approaches 0 the better. To generate the heterogeneous environment, we introduce Gaussian noise to the physical coefficient of the environment, namely, the mass of the cart in *CartPole* and the length of the pole in *Pendulum* etc. For the *BipedalWalkerHardcore* environment, we aim to study how the data imbalance issue in the FRL setting affects the training.

In this environment, the goal of the robot is to go as far as possible. The reward is given for moving forward and totaling 300+ points up to the far end. The robot gets punished when it falls to the ground. The state consists of the hull angle speed, the angular velocity, the horizontal speed, the vertical speed, the position of joints, the joints' angular speed, and a boolean indicator of whether the legs contact with the ground and the measurements of 10 lidar range finder. There are three kinds of obstacles in this environment, normally in the original environment setting, the occurrence of each kind of obstacle is the same probability. We skew the probability so that the data collected by each agent would be imbalanced in the local replay buffers. In Lunar Lander, the agent perceives the coordinate of the landing pad and its own velocity, and the goal is to land on the pad safely. The initial random force perturbation applies to the rocket and the smoothness of the land is set to be different. Figure 4 shows some exemplary local environments.

5.1.4 Metrics. In our experiments, we evaluate all algorithms from the perspective of the overall performance and stability in each local environment. For the classical control environment, we set the different seeds to generate different environmental parameters and report the average reward of different seeds. For *BipedalWalkerHardcore* and *Lunar Lander*, the local environment parameters

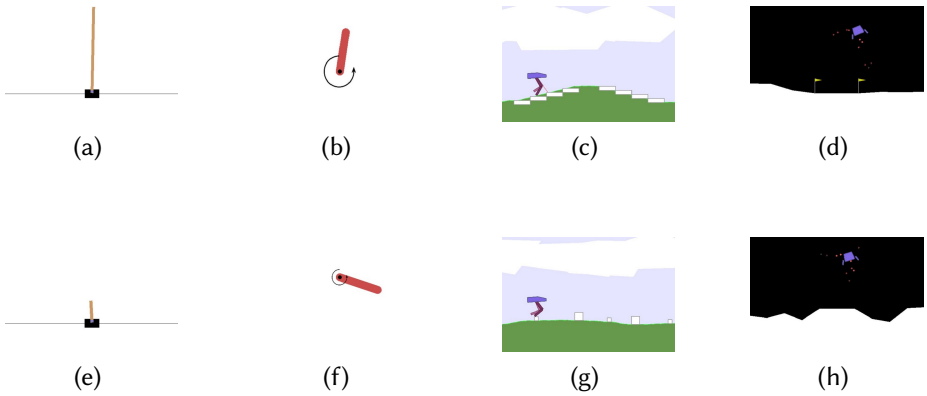


Fig. 4. (a)&(e): Cartpole; (b)&(f): Pendulum; (c)&(g) BipedalWalkerHardcore; (d)&(h): Lunar Lander. For CartPole and Pendulum, the length of the pole and other physical parameters *e.g.* gravity and mass are different. For BipedalWalkerHardcore, the agent interacts with the environment with three kinds of obstacles, and the appearance probability is variant across the local environments. For Lunar Lander, the regularity of the surface of the moon and the height of the landing pad in different environments vary.

Table 2. The setting of the key hyper-parameters of SCCD. Server epochs and client epochs represent the number of distillation epochs on the server and client side.

	distill lr	**server epochs**	**client epochs**	**replay buffer size**
CartPole	1 e-02	10	40	1 e+04
Pendulum	1 e-02	20	20	1 e+04
Walker	1 e-02	20	40	1 e+06
Lunar	1 e-02	40	10	3.2 e+04

are fixed, because empirically the variance of the final result can be very large even if we set it to be fixed. We run multiple times for each method and report their final average reward.

5.2 Evaluation with Different Heterogeneity

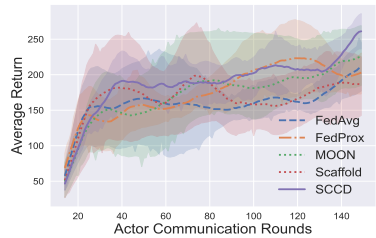
We evaluate the aforementioned baseline methods and SCCD under different environment heterogeneity. For the classic control tasks, the standard deviation σ of the Gaussian noise added to the environment parameters would be varied, the bigger the σ the more heterogeneous the environments. For the *BipedalWalkerHardcore* environment, in each local environment, the probability of each obstacle is generated by the Dirichlet distribution. However, it's not like supervised federated learning, as the probability is generated within each environment so that the probability of each obstacle would sum to one. Therefore, the heterogeneity of the environments is evaluated by the Jensen-Shannon divergence [11] of each environmental probability.

$$D_{\mathrm{JS}}(p||q) = \frac{1}{2}D_{\mathrm{KL}}\left(p||\frac{p+q}{2}\right) + \frac{1}{2}D_{\mathrm{KL}}\left(q||\frac{p+q}{2}\right).$$

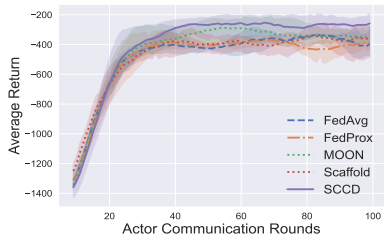
In the above equation, p and q refer to probabilistic vectors of each environment, and we sum up the paired Jensen-Shannon divergence as the overall heterogeneity of the Bipedal-Walker-Hardcore problem. Similarly in the classic control problem, the higher value of D_{JS} means that the distributions of each environment are more dispersed. The heterogeneity is defined as three levels:

Table 3. Average reward of different federated learning methods in four environments with heterogeneity $h_1: (\sigma = 0.05, D_{\text{JS}} = 0)$, $h_2: (\sigma = 1, D_{\text{JS}} = 2.43)$, $h_3: (\sigma = 2, D_{\text{JS}} = 4.7)$.

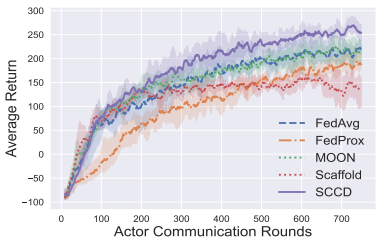
Datasets		FedAvg	MOON	FedProx	Scaffold	FedDF	MTFRL	FRD	SCCD (ours)
CartPole	h_1	289.15±21.69	295.13±9.73	294.58±10.75	248.85±54.97	135.08±120.19	234.68±43.7	240.13±77.57	**294.77**±11.9
	h_2	220.94±46.46	216.57±36.56	243.30±60.57	182.53±53.94	12.58±6.03	196.87±34.19	215.18±67.22	**254.31**±32.09
	h_3	233.16±46.67	207.88±43.51	193.35±37.72	231.4±44.17	24.42±13.68	218.19±41.75	115.92±38.56	**252.39**±10.74
Pendulum	h_1	-148.54±5.64	-207.35±110.28	-146.65±4.13	-157.61±5.69	-149.90±5.11	-150.08±0.96	-154.74±21.29	**-147.22**±4.35
	h_2	-364.30±90.24	-364.87±49.97	-301.66±97.45	-388.76±41.29	-511.45±161.91	-380.78±135.32	-528.88±68.93	**-221.94**±76.98
	h_3	-537.14±121.62	-522.14±183.05	-518.56±218.37	-577.81±339.88	-533.29±169.94	-537.01±134.93	-730.01±260.31	**-342.93**±114.49
BipedalWalker	h_1	263.91±7.34	247.71±15.90	186.24±20.97	232.24±11.44	-10.76±7.19	-72.85±15.08	229.08±25.09	**266.44**±13.46
	h_2	247.45±3.12	233.14±266.66	206.13±31.6	181.58±45.7	-57.62±19.13	-70.86±10.06	223.92±10.06	**261.90**±11.39
	h_3	232.31±23.32	227.35±18.91	196.17±14.84	130.82±24.71	-34.81±48.50	-86.02±3.23	213.33±12.4	**271.33**±14.62
Lunar Lander	h_1	212.91±16.35	227.19±3.29	**227.79**±6.72	218.96±16.87	115.23±144.27	131.01±14.19	125.45±51.49	221.72±18.13
	h_2	196.47±17.18	205.62±17.04	214.18±16.16	152.87±12.63	196.46±13.51	124.81±11.43	140.98±47.16	**220.02**±0.84
	h_3	178.59±27.50	202.69±16.30	218.02±2.49	173.83±26.22	184.05±17.31	115.97±6.84	163.21±61.09	**219.18**±13.97



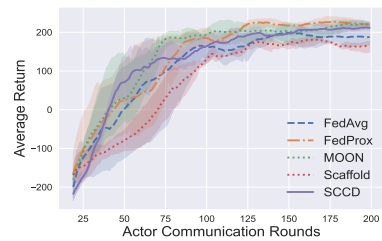
(a) CartPole



(b) Pendulumn



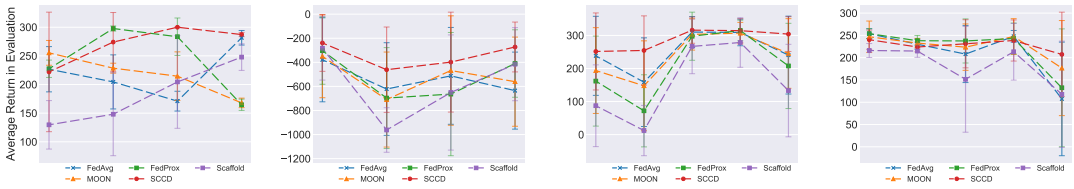
(c) BipedalWalker



(d) Lunar Lander

Fig. 5. The validation curves during training were chosen from different heterogeneity levels. Each point value is the mean over four learning curves and then smoothed by a sliding window with a fixed window width. The shadow is the standard deviation of each trial on a particular point.

$h_1: (\sigma = 0.05, D_{\text{JS}} = 0)$, $h_2: (\sigma = 1, D_{\text{JS}} = 2.43)$, $h_3: (\sigma = 2, D_{\text{JS}} = 4.7)$. For the classical control tasks, we run 4 times with different random seeds, each time the environment parameter would be different. In the testing period, each agent would be tested in the local environment for 100 episodes, and the average reward over all the local environments is reported as the final performance.



(a) CartPole (b) Pendulumn (c) BipedalWalker (d) Lunar Lander

Fig. 6. The error bar of each dataset in different local environmental parameters. For the classical control datasets *CartPole* and *Pendulumn*, the x-axis represents 4 experiments with a different random seed. For *BipedalWalkerHardcore* and *Lunar Lander*, the x-axis represents 5 local environments. We choose the best performance of each method among different trials.

Table 3 shows the overall performance of the baselines and SCCD. By comparing FedAvg, FedProx, and MOON, we can see that both MOON and FedProx cannot always improve the performance in the former three environments and sometimes even get a lower reward than FedAvg, which indicates that by adding a regular term to constraint the model is not always suitable in the FRL setting. However, SCCD utilizes the distillation technique to overcome the heterogeneity issue and gets the best result in a highly heterogeneous environment. We can also observe that generally, the average reward of each method decreases with the increasing heterogeneity. SCCD could perform well in these four heterogeneous environments and it is more robust to high-level heterogeneity than the other methods. We plot training curves of each baseline method and SCCD in a specific heterogeneity level in Figure 5. The x-axis represents the actor communication times (1 round as a unit in *CartPole* and *Pendulum*, 3 rounds as a unit in *BipedalWalkerHardcore* and *Lunar Lander*). Compared to the other methods, most of the time SCCD converges faster than the other methods in the early training stage, which means the server-side distillation can effectively enhance the evolution of the critic model by distilling knowledge from local models. Besides, the smaller standard deviation suggests the potential stabilization ability of the client-side distillation.

5.3 Generalization and Personalization

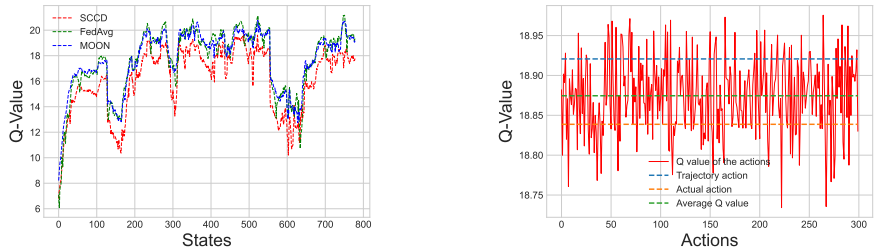
In this section, we analyze the generalization performance across the local environments of each method. For *CartPole* and *Pendulum*, we draw the error bar of the average test score with different random seeds. For the *BipedalWalkerHardcore* and *Lunar Lander*, we choose one of the best test results of each method and plot the agent's performance in five local environments. Based on the results, From Figure 6(a)&(b), we can observe that SCCD is more robust to different environment parameters (the dashed line of SCCD is straighter than the others). The changing random seed does not bring a huge influence on SCCD compared to the other methods.

Figure 6(c)&(d) reflects the performance of the agent in the local environments. In Figure 6(c), the *BipedalWalkerHardcore* local environment 2 (the second bar) only appears the stump, which is a much more difficult environment than the other local ones. We can observe that the performance of SCCD can still exceed the others, which indicates its better generalization and personalization ability.

To analyze the reliability of each method, we roll out a trajectory in *BipedalWalkerHardcore* (about 776 state-action pairs within one episode) from a pre-trained policy π' and plot the Q-value of three methods in Figure 7(a). We can observe that the baseline methods tend to have a higher Q-value which means the actor may select overestimated actions that could lead to poor performance.

Table 4. T-test for the difference between the reward of origin policy and updated policy. "Original" represents the mean reward of the original policy, the same for "Updated". The improvement of the updated policy may indicate that the Q network learned by the agents is trustworthy and reliable. If the statistic is positive, the lower the P-value represents the higher confidence for the improvement of the policy.

Methods	Original	Updated	Difference	Statistic	P-val
FedAvg	261.77	259.92	-1.85	-0.16	0.87
Moon	266.49	275.70	+9.2	0.82	0.41
FedProx	206.91	192.97	-13.93	-0.91	0.36
MTFRL	-57.95	-58.48	-0.53	-0.19	0.84
FRD	256.39	244.97	-11.42	-0.84	0.39
SCCD	269.11	281.34	+12.22	0.99	0.32



(a) (b)

Fig. 7. (a) The Q-value of FedAvg, MOON, and SCCD. We can see that FedAvg and MOON would have an overestimation issue. (b) We sample one state from the replay buffer and concatenate it with 300 continuous actions sampled from the action space as the input, and plot the output Q value of SCCD.

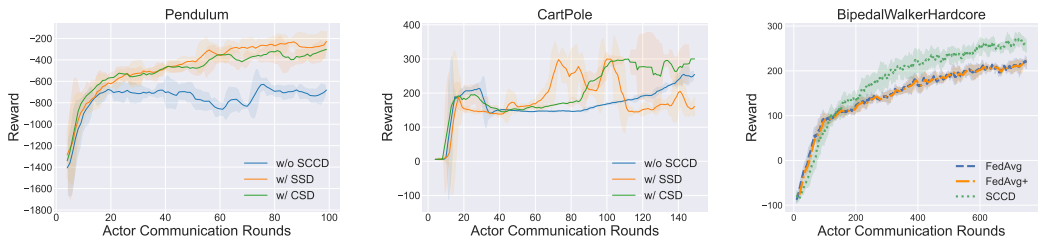
In Figure 7(b), we sample one state from the replay buffer and plot the Q value of the actions sample from the action space. The red line shows the Q value of the random actions and the blue line is the Q value of the action chosen by the agent during training. While the orange line is the value of the action chosen by the agent during testing. From the result we can see that the agent would not always choose the action with a high value given a state in the testing period, this phenomenon appears in all methods. With this, we want to further explore whether there's still room to improve the actor by performing one policy optimization step according to the value network learned by the agents. Specifically, we use the well-trained agent to play one episode game and collect the trajectory data, later on, the agent's actor network would be updated based on these station-action pairs by the policy gradient, which means we enforce the agent to act with a higher Q value. Theoretically, the agent could enhance its performance as long as the value function is accurate enough. We run 100 trials of one-step policy updates of each method. Table 4 shows the results of the single sample T-test for the difference between the reward of the original policy and the updated policy. We can see the mean reward of the original policy of SCCD and MOON can still be enhanced, especially SCCD has a large improvement space, while the others decline to different degrees.

5.4 Ablation Study

In this section, we target to answer the following questions:

Table 5. The effect of extra local training. FedAvg+ means more local training iterations than FedAvg.

Dataset	Method	trial 1	trial 2	trial 3	trial 4	overall
CartPole	FedAvg+	174.26	243	294.33	238.46	237.51±49.19
	FedAvg	226.67	204.55	170.96	281.58	220.94±46.46
	SCCD	209.62	253.2367	281.8733	272.52	**254.31 ± 32.09**
Pendulum	FedAvg+	-384.59	-747.28	-625.35	-670.51	-606.933±184.56
	FedAvg	-379	-621.85	-512.63	-635.08	-537.14±121.63
	SCCD	-239.23	-461.38	-398.42	-272.72	**−342.938 ± 114.50**
BipedalWalker	FedAvg+	229.61	198.14	218.14	247.56	225.50±20.72
	FedAvg	240.75	204.90	251.29	205.08	225.51±24.07
	SCCD	260.96	264.94	288.04	272.27	**271.55 ± 11.95**



(a) (b) (c)

Fig. 8. (a)&(b) The average score of one of the local environments during the training period in Pendulum and CartPole. (c) The learning curve of FedAvg, FedAvg+ and SCCD in BipedalWalker.

- Can local agents benefit from server-side distillation?
- Is the superior performance of SCCD caused by the extra local training?

For question 1, we can compare the results of SCCD and FRD. SCCD involves both client-side distillation (CSD) and server-side distillation (SSD), while FRD only involves CSD. As shown in Table 3, FRD tends to perform well when the local environments are complementary, such as in the case of *BipedalWalker*. However, without a fused Q network on the server side, the aggregated actor in FRD may not be able to effectively handle diverse local environment dynamics in other datasets. This limitation of FRD is one reason why SCCD is a more robust choice in certain situations.

Furthermore, we decompose SCCD and compare the performance of its two components individually with a baseline method that does not involve either CSD or SSD (FedAvg). Figure 8 shows the learning curve in Pendulum Figure 8(b) and CartPole Figure 8(c), where the blue line denotes the training curve without SCCD *i.e.* the FedAvg. We can see that with the assistance of the SSD, the training could converge faster. From Figure 8(c), the learning curve of the SSD could achieve the highest score, but it also becomes very unstable and has dropped down lately. The negative transfer probably causes this during the late stage. The final score it gets is even worse than the FedAvg. In comparison, with CSD, the agent performance could remain at a relatively high score, which means the CSD could stabilize the training.

Question 2 can be verified by increasing the local training steps in FedAvg. Each round the client receives the global predictor from the server and then we can perform an extra TD update as the same as SCCD. Table 5 shows the testing results in 4 random seeds with the same heterogeneity. In

CartPole the performance of FedAvg is increased modestly but still can not exceed SCCD. However, in Pendulum, the extra training would even spoil the correctness of the critic model and lead to worse performance. Figure 8(d) shows the learning curve of FedAvg+, extra training does not change the convergence rate and improve FedAvg. Therefore, the benefit is not due to the extra training steps, but because CSD is more effective than the naive model aggregation.

6 CONCULSION AND FUTURE WORK

In this paper, we have proposed a federated reinforcement learning framework `FedTD3`, which is designed on top of the state-of-the-art RL algorithm TD3 to federally train a robust intelligent agent. On the basis of `FedTD3`, we have presented a server-client side model update procedure termed SCCD to overcome the non-stationary heterogeneity issue that occurred in federated reinforcement learning. By conducting a range of experiments, the results have shown that the proposed method has superior performance, in terms of generalization ability and communication efficiency, in comparison with the other federated learning counterparts.

In the future, we will further explore the generalization of SCCD by integrating it into other off-policy RL frameworks like SAC. On the other hand, it is crucial to explore a more accurate distribution generation method for the local representations. Additionally, It's worth noting that fairness is also an important issue in FRL. While our proposed SCCD approach in FRL does not directly tackle fairness, it does offer a framework for integrating fairness considerations into the development of distributed reinforcement learning systems. For example, the SCCD approach can be extended to incorporate fairness constraints or objectives during the training process, such as ensuring that the learned policies do not discriminate against certain groups of agents. Another point is while designing the local reward function, make sure the distribution of rewards is equitable across all agents. We encourage future research in FRL to consider fairness as an important aspect of the design and evaluation of distributed RL systems.

7 ACKNOWLEDGEMENTS

WMM and BH were supported by NSFC Young Scientists Fund No. 62006202, Guangdong Basic and Applied Basic Research Foundation No. 2022A1515011652, RGC Early Career Scheme No. 22200720, CAAI-Huawei MindSpore Open Fund, and HKBU CSD Departmental Incentive Grant. YMC was supported in part by the NSFC/Research Grants Council (RGC) Joint Research Scheme under Grant: N_HKBU214/21, in part by the General Research Fund of RGC under Grants: 12202622 and 12201321, and in part by Hong Kong Baptist University (HKBU) under Grant: RC-FNRA-IG/18-19/SCI/03. JCY and YZ were supported by the National Key R&D Program of China (No. 2022ZD0160702, No. 2022ZD0160703), STCSM (No. 22511106101, No. 22511105700, No. 21DZ1100100), 111 plan (No. BP0719010). CG was supported by NSF of China (No: 61973162), NSF of Jiangsu Province (Nos: BZ2021013, BK20220080), and the Fundamental Research Funds for the Central Universities (Nos: 30920032202, 30921013114).

REFERENCES

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2017. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641* (2017).

[2] Aqeel Anwar and Arijit Raychowdhury. 2021. Multi-task federated reinforcement learning with adversaries. *arXiv preprint arXiv:2103.06473* (2021).

[3] Richard Bellman. 1954. The theory of dynamic programming. *Bull. Amer. Math. Soc.* 60, 6 (1954), 503–515.

[4] Anand Bodas, Bhargav Upadhyay, Chetan Nadiger, and Sherine Abdelhak. 2018. Reinforcement learning for game personalization on edge devices. *2018 International Conference on Information and Computer Technologies, ICICT 2018* (2018), 119–122. https://doi.org/10.1109/INFOCT.2018.8356853

[5] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017* 54 (2017). arXiv:1602.05629

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

[7] Han Cha, Jihong Park, Hyesung Kim, Mehdi Bennis, and Seong Lyun Kim. 2020. Proxy Experience Replay: Federated Distillation for Distributed Reinforcement Learning. *IEEE Intelligent Systems* 35, 4 (2020), 94–101. https://doi.org/10.1109/MIS.2020.2994942 arXiv:2005.06105

[8] Baiming Chen, Zuxin Liu, Jiacheng Zhu, Mengdi Xu, Wenhao Ding, Liang Li, and Ding Zhao. 2021. Context-aware safe reinforcement learning for non-stationary environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10689–10695.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[10] Xiaofeng Fan, Yining Ma, Zhongxiang Dai, Wei Jing, Cheston Tan, and Bryan Kian Hsiang Low. 2021. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems* 34 (2021).

[11] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* IEEE, 31.

[12] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.

[13] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning, ICML 2018* 4 (2018), 2587–2601. arXiv:1802.09477

[14] Ruchi Gupta and Tanweer Alam. 2022. Survey on federated-learning approaches in distributed environment. *Wireless Personal Communications* 125, 2 (2022), 1631–1652.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[16] Hao Jin, Yang Peng, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2022. Federated reinforcement learning with environment heterogeneity. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 18–37.

[17] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*. PMLR, 5132–5143.

[18] Young Geun Kim and Carole-Jean Wu. 2021. Autofl: Enabling heterogeneity-aware energy efficient federated learning. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 183–198.

[19] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[20] Kwei-Herng Lai, Daochen Zha, Yuening Li, and Xia Hu. 2020. Dual policy distillation. *arXiv preprint arXiv:2006.04061* (2020).

[21] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[22] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10713–10722.

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated Optimization in Heterogeneous Networks. (2018). arXiv:1812.06127 http://arxiv.org/abs/1812.06127

[24] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).

[25] Xinle Liang, Yang Liu, Tianjian Chen, Ming Liu, and Qiang Yang. 2019. Federated transfer reinforcement learning for autonomous driving. *arXiv preprint arXiv:1910.06001* (2019).

[26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[27] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 2351–2363.

[28] Boyi Liu, Lujia Wang, and Ming Liu. 2019. Lifelong Federated Reinforcement Learning: A Learning Architecture for Navigation in Cloud Robotic Systems. *IEEE International Conference on Intelligent Robots and Systems* 4, 4 (2019), 1688–1695. https://doi.org/10.1109/IROS40897.2019.8967908

[29] Fan-Ming Luo, Shengyi Jiang, Yang Yu, Zongzhang Zhang, and Yi-Feng Zhang. 2022. Adapt to Environment Sudden Changes by Learning a Context Sensitive Policy. (2022).

[30] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. 2021. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems* 34 (2021).
[31] Aritra Mitra, Rayana Jaafar, George J. Pappas, and Hamed Hassani. 2021. Linear Convergence in Federated Learning: Tackling Client Heterogeneity and Sparse Gradients. NeurIPS (2021). arXiv:2102.07053 http://arxiv.org/abs/2102.07053
[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[33] Chetan Nadiger, Anil Kumar, and Sherine Abdelhak. 2019. Federated reinforcement learning for fast personalization. *Proceedings - IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019* (2019), 123–127. https://doi.org/10.1109/AIKE.2019.00031
[34] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *34th International Conference on Machine Learning, ICML 2017* 6, July (2017), 4108–4122. arXiv:1703.06182
[35] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.
[36] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. 2021. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887* (2021).
[37] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295* (2015).
[38] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.
[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[40] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.
[41] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).
[42] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.
[43] Jin Wang, Jia Hu, Jed Mills, and Geyong Min. 2021. Federated Ensemble Model-based Reinforcement Learning. (2021), 1–14. arXiv:2109.05549 http://arxiv.org/abs/2109.05549
[44] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481* (2020).
[45] Jiangchao Yao, Feng Wang, Xichen Ding, Shaohu Chen, Bo Han, Jingren Zhou, and Hongxia Yang. 2022. Device-Cloud Collaborative Recommendation via Meta Controller. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4353–4362.
[46] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. 2021. Device-Cloud Collaborative Learning for Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3865–3874.
[47] Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, et al. 2022. Edge-cloud polarization and collaboration: A comprehensive survey for ai. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[48] Linlin You, Mazen Danaf, Fang Zhao, Jinping Guan, Carlos Lima Azevedo, Bilge Atasoy, and Moshe Ben-Akiva. 2023. A Federated Platform Enabling a Systematic Collaboration Among Devices, Data and Functions for Smart Mobility. *IEEE Transactions on Intelligent Transportation Systems* 24, 4 (2023), 4060–4074.
[49] Hankz Hankui Zhuo, Wenfeng Feng, Yufeng Lin, Qian Xu, and Qiang Yang. 2019. Federated Deep Reinforcement Learning. (2019). arXiv:1901.08277 http://arxiv.org/abs/1901.08277

Received 1 January 2023; revised 26 March 2023; accepted 8 June 2023