TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 20/20 pp911–926
DOI: 10.26599/TST.2023.9010066
Volume 29, Number 3, June 2024

Federated Meta Reinforcement Learning for Personalized Tasks

Wentao Liu, Xiaolong Xu*, Jintao Wu, and Jielin Jiang

Abstract: As an emerging privacy-preservation machine learning framework, Federated Learning (FL) facilitates different clients to train a shared model collaboratively through exchanging and aggregating model parameters while raw data are kept local and private. When this learning framework is applied to Deep Reinforcement Learning (DRL), the resultant Federated Reinforcement Learning (FRL) can circumvent the heavy data sampling required in conventional DRL and benefit from diversified training data, besides privacy preservation offered by FL. Existing FRL implementations presuppose that clients have compatible tasks which a single global model can cover. In practice, however, clients usually have incompatible (different but still similar) personalized tasks, which we called task shift. It may severely hinder the implementation of FRL for practical applications. In this paper, we propose a Federated Meta Reinforcement Learning (FMRL) framework by integrating Model-Agnostic Meta-Learning (MAML) and FRL. Specifically, we innovatively utilize Proximal Policy Optimization (PPO) to fulfil multi-step local training with a single round of sampling. Moreover, considering the sensitivity of learning rate selection in FRL, we reconstruct the aggregation optimizer with the Federated version of Adam (Fed-Adam) on the server side. The experiments demonstrate that, in different environments, FMRL outperforms other FL methods with high training efficiency brought by Fed-Adam.

Key words: federated learning; reinforcement learning; meta-learning; personalization

1 Introduction

In recent years, applications based on machine learning have achieved great success in various areas, which is inseparable from the sustentation of abundant data. To deal with the sheer volume of data, sufficient computing power is essential for model training and is usually realized by the distributed system[1–5]. Meanwhile, the development of end devices (smartphones, laptops, Internet of Things (IoT) devices, etc.) further expands the scope of distributed model training. For the public, large-scale data collection is a double-edged sword. While providing more accurate services, data also portray users' behavior habits, leaking personal information, etc. The data privacy laws such as General Data Protection Regulation (GDPR)[6] also reflect governments' concerns about the risk of data breaches. In this context, data privacy for distributed learning is becoming the spotlight. To tackle the privacy problem, Federated Learning (FL)[7] is proposed as a new paradigm that let different devices collaboratively train a shared global model without exchanging any raw data with each other. The most commonly used method is FedAvg[7] which averages the collected models to obtain the new global model. As a universal framework, FL can be adapted to most machine learning methods, including supervised learning and reinforcement learning, that is, Federated

• Wentao Liu is with School of Computer Science, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: liuwentao728@gmail.com.
• Xiaolong Xu, Jintao Wu, and Jielin Jiang are with School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: xlxu@nuist.edu.cn; wjt@nuist.edu.cn; jiangjielin2008@nuist.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-04-03; revised: 2023-06-26; accepted: 2023-06-27

Reinforcement Learning (FRL)[8].

Due to the feature extraction ability on different levels, especially in high-dimensional space[9, 10], Deep Reinforcement Learning (DRL) becomes increasingly popular and shows its power in various domains that involve decision-making issues such as network control[11], robotics[12], and autonomous driving[13]. As the price for good performance, DRL requires a long time and colossal amounts of records for policy training. The poor training and sampling efficiency hinders the implementation of DRL in some real-world scenarios, for example, in healthcare, where data trajectories cannot be generated synthetically or exchanged due to privacy preservation[14, 15]. For these privacy-sensitive scenarios, FRL provides a credible solution that collaborates with multiple data sources to improve the training efficiency of DRL policy while avoiding the exposure of data trajectories. Due to the distributed nature of FRL, the participating clients are able to construct a high-performance global policy in their local environments without heavy sampling and training overhead.

However, given the various requirements in reality, clients involved in FRL usually have similar but different personalized tasks, which we call task shift. For example, for resource management in edge computing[16–18], depending on the location of the edge server, service hotspots conform to different distributions, which directly influences the computing offloading or service caching decision of the policy. The existing works of FRL[19–21] aim to obtain a shared global model as the policy for decision making, and the differences in tasks are explicitly presented in some observation dimensions. There is no doubt when the differences are implicit such as service hotspots, it is hard for a single policy to cover all these hidden distributions. For those tasks with implicit differences, the same policy may polarize on different tasks. The phenomenon is rooted in the Non-Identically and Independently Distributed (Non-IID) setting of FL[22], which is one of the main research branches in FL[23, 24]. And it also derives a concept, Personalized Federated Learning (PFL)[25, 26], which calls on each client to develop their personalized local models. At present, most of the works on PFL focused on supervised learning, but there is little on reinforcement learning. Different from supervised learning, the Non-IID problem in reinforcement learning is mainly presented on the discrepancy of tasks instead of the distribution of data inputs or labels, which limits the direct immigration of those PFL works in supervised learning to FRL.

In this paper, we propose a Federated Meta Reinforcement Learning method, named FMRL, to explore the possibility of PFL in reinforcement learning. The method builds on the Model-Agnostic Meta-Learning (MAML)[27] and we reformulate the FRL problem to fit our goal which is to find an easy-trained global model for different personalized tasks. It can quickly align with the task of the current client through a few interactions with the environment. At each step of the local training, the client needs to sample data with the current policy, which brings a large sampling overhead for the multi-step local training. For sampling and training efficiency, we utilize importance sampling and update clipping in Proximal Policy Optimization (PPO)[28] to achieve multi-step local training with a single round of sampling. Considering the characteristics of FRL (sensitive local learning rate and high variance of the model aggregation), we adopt the Federated version of Adam[29, 30] (Fed-Adam) as the aggregation optimizer to adaptively adjust the update rate of the global model, to accelerate the training process. Meanwhile, we extract the general form of the aggregation optimizer and compare its difference in FedAvg, FMRL, and other PFL methods.

Our contributions:

- We expand PFL to the field of reinforcement learning and reformulate the problem with MAML to fit the personalized task problem.
- Due to the requirements of multi-batch sampling in MAML, we integrate importance sampling to greatly reduce the sampling rounds of MAML. In addition, update clipping is adopted to improve the training efficiency of multi-step local training on the client side.
- To accelerate the convergence of the global model, we replace the conventional aggregation optimizer in FedAvg with Fed-Adam. It will adjust the global update rate according to the uploaded client model, which is more suitable for FRL scenarios. The related experiments show that Fed-Adam greatly improves training efficiency.
- We empirically evaluate the performance of FMRL in various environments in which each client has different personalized tasks. The experiments attest that FMRL achieves higher rewards than other methods with the local model after 1, 2, and 3 inner adaptions.

Besides, the adaptive experiment shows that FMRL also performs well on those new clients who have not participated in federated learning before.

The paper is organized as follows. Section 2 summarizes current related works on FRL and PFL. Section 3 describes the details of our model. Section 4 presents the experiment results on the performance and training efficiency. Section 5 concludes our work and proposes future work. The acronyms and main notations in this paper are listed in Tables A1 and A2 in Appendix A.

2 Related Work

2.1 Federated reinforcement learning

In Ref. [8], the concept of FRL is first defined and the corresponding application scenarios are discussed. It constructs the private Q-network for each client and uploads the clients' Q-values with Gaussian noise for the inference of the global policy. The approach enables those clients who lack complete data trajectories to generate their own strategies still. Liu et al.[19] proposed a lifelong FRL architecture for navigation in cloud robotic systems. The robots learn their policies in local environments and the server fuses them to the shared policies on the cloud for downloading and training by new robots. Liang et al.[20] studied federated transfer reinforcement learning in autonomous driving. It transfers the knowledge from the simulation environments to the real world, allowing clients in different kinds of environments to participate in the federation process. Nadiger et al.[21] employed FRL to realize fast personalization of non-player characters in games. Similar clients are classified into the same group and share data to train the global policy collaboratively. Compared to the others, this work considers the personalization of FRL and utilized multiple global models for different user groups.

The aforementioned works except Ref. [21] consider the difference in client tasks and expect the global policy to perform well in a new environment. But they assume tasks are compatible with each other, which means the global policy can accumulate knowledge continuously and handle all tasks simultaneously. For incompatible tasks, approaching one task must lead to a deviation from another for a single policy. The conventional methods of FRL which try to find the best global policy would fail in this scene. And for Ref. [21], it avoids the above-mentioned problems by constructing corresponding policies for different clusters, which have a similar idea to a part of PFL methods[31–33]. However, the idea is based on clustering which needs to collect users' information and this step may bring privacy concerns.

2.2 Personalized federated learning

Multiple approaches have been proposed for PFL and the comprehensive survey for PFL can be found in Ref. [34]. Overall, there are several popular directions for the current works: mixtures of the global and local models[35–37], learning between relevant clients (clustering)[31–33], model regularization[38–40], and meta-learning[41–43]. Hanzely and Richtárik[35] proposed L2GD algorithm to realize the mixture optimization of the global and local models. Mansour et al.[32] presented three PFL methods: user clustering, data interpolation, and model interpolation. Thereinto, the user clustering method supposes the potential relationship of clients and trains the global model per client group. To render the clients for pursuing their personalized model while not deviating far from the global model, Dinh et al.[38] proposed pFedMe algorithm which attaches a new regularization with Moreau envelopes, controlling the personalization degree. Fallah et al.[42] introduced MAML into the FL process and proposed the personalized version of FedAvg, Per-FedAvg, which reformulates the FL goal to obtain a high-adaptive global model. Due to the high cost of computing Hessian matrix in MAML, several works investigated the approximate estimates[27, 44].

Considering the characteristics of reinforcement learning, it is hard to migrate existing PFL works directly. The main problem is that the collected data trajectories depend on the current policy, rather than the fixed prior distribution in supervised learning. Also, the privacy of users' tasks is another point that needs to consider. It means the PFL methods based on clustering may be more difficult to be adopted in FRL.

3 Federated Meta Reinforcement Learning with MAML

In the general FL scenario, numerous clients participate the federation process under the orchestration of a central server for the common goal:

$$\min_{\theta\in\mathbb{R}^d} f(\theta) := \frac{1}{|C|}\sum_{i\in C} f_i(\theta) \tag{1}$$

where θ represents the global model parameters and C is the client set. The process aims to find an optimal model θ^* to minimize the objective function $f(\theta)$.

Compared with the above objective, MAML pays more attention to the optimizability of the model, which is defined as the model performance after one-step gradient descent. The final goal of MAML is to find a set of parameters that can quickly converge to an approximate optimal solution. Switch to the FL scene with MAML, we need to get an appropriate global model, the clients can train their personalized models after a few rounds of local training based on this global model. The goal can be reformulated as

$$\min_{\theta\in\mathbb{R}^d} F(\theta) := \frac{1}{|C|}\sum_{i\in C} F_i(\theta) \tag{2}$$

where $F_i(\theta)$ is denoted as

$$F_i(\theta) := f_i(\theta - \alpha\nabla f_i(\theta)) \tag{3}$$

Thereinto, α is the learning rate of inner adaption and $F_i(\theta)$ presents the objective function of client i with model θ after one-step inner adaption. In this section, we first compare the specific form of $f_i(\theta)$ in supervised learning and reinforcement learning. Then, we formulate the personalization problem in FRL. In the end, we propose the FMRL algorithm to solve the problem raised in Eq. (2) and discuss the implementation of different federated optimizers.

3.1 Problem formulation

Under the setting of supervised learning, $f_i(\theta)$ represents the expected loss on the dataset of client i which is defined as

$$f_i(\theta) := E_{(x,y)\sim p_i}[l(\theta; x, y)] \tag{4}$$

where p_i is the data distribution of client i and $l_i(\theta; x, y)$ is the loss function of model θ. In Non-IID scenarios, the data distribution for each client is usually different and fixed.

For reinforcement leaning, it can be modelled as a Markov Decision Process (MDP)[45]. The MDP of client i is defined as $\mathcal{M}_i := (\mathcal{S}_i, \mathcal{A}_i, P_i, \rho_i, r_i)$. $\mathcal{S}_i$ represents the state space and $\mathcal{A}_i$ represents the action space. $\rho_i(s)$ is the probability of initial state $s \in \mathcal{S}_i$, and $P_i(s'|s,a)$ indicates the transfer probability from state s to s' when the agent executes action $a \in \mathcal{A}_i$. The reward of action a under state s is defined as $r_i(s,a)$. The interaction of the agent with the environment can be described as the trajectory $\tau := (s_0, a_0, s_1, a_1, ..., s_H, a_H)$ where H is the time horizon, we suppose H is fixed. The agent follows the policy $\pi_i(\cdot|\cdot;\theta)$ to interact with the environment, and $\pi_i(a|s;\theta)$ is the probability of taking action a under state s. The sampling probability of τ depends on the policy $\pi_i(\theta)$[§] and is given by

$$q_i(\tau;\theta) := \rho_i(s_0)\prod_{i=0}^{H}\pi_i(a_h|s_h;\theta)\prod_{h=0}^{H-1}P_i(s_{h+1}|s_h,a_h) \tag{5}$$

The discounted total reward of a single trajectory τ is defined as

$$R_i(\tau) := \sum_{h=0}^{H}\gamma^h r_i(s_h,a_h) \tag{6}$$

where γ is the discount factor. At this point, the specific form of $f_i(\theta)$ in reinforcement learning is given as

$$f_i(\theta) := -\underbrace{E_{\tau\sim q_i(\cdot|\theta)}[R_i(\tau)]}_{J_i(\theta)} \tag{7}$$

Note that the problem defined in Eq. (1) is a minimization issue but the goal in reinforcement learning is to maximize the reward. To align with Eq. (1), $f_i(\theta)$ is defined as the opposite of $J_i(\theta)$. Comparing Eqs. (4) and (7), the significant difference in the two settings derives from the data distributions p_i and $q_i(\cdot|\theta)$. The former is a fixed probability, while the latter is affected by model θ. It requires clients to sample trajectories before each local training, which brings a large sampling overhead and restricts the rounds of multi-step training. To improve the training efficiency, we introduce PPO and institute $J_i(\theta)$ in Eq. (7) with

$$J_i^{\hat\theta}(\theta) := E_{(s_h,a_h)\sim\pi_i(\hat\theta)}[\min(u_i^{\hat\theta}(s_h,a_h;\ \theta), u_i^{\text{clip},\hat\theta}(s_h,a_h;\ \theta))] \tag{8}$$

where $u_i^{\hat\theta}(s_h,a_h;\theta)$ is defined as

$$u_i^{\hat\theta}(s_h,a_h;\theta) = \frac{\pi_i(a_h|s_h;\theta)}{\pi_i(a_h|s_h;\hat\theta)}A_i(s_h,a_h;\hat\theta) \tag{9}$$

and $u_i^{\text{clip},\hat\theta}(s_h,a_h;\ \theta)$ is defined as

$$u_i^{\text{clip},\hat\theta}(s_h,a_h;\ \theta) = \text{clip}\left(\frac{\pi_i(a_h|s_h;\ \theta)}{\pi_i(a_h|s_h;\ \hat\theta)}, 1-\epsilon, 1+\epsilon\right)\cdot A_i(s_h,a_h;\ \hat\theta) \tag{10}$$

In Eq. (8), the basic unit of data is no longer a whole trajectory, but a tuple of the state and the action (s_h,a_h) which sampled by $\pi_i(\hat\theta)$. $\pi_i(\hat\theta)$ is the old policy which sampled data and $\pi_i(\theta)$ is the current policy to be optimized. For the first condition of choosing

[§]To simplify the notation, we suppress policy $\pi_i(\cdot|\cdot;\theta)$ as $\pi_i(\theta)$. Meanwhile, we call $\pi_i(\theta)$ policy and call θ model to differentiate.

$u_i^{\hat{\theta}}(s_h,a_h;\theta)$ defined in Eq. (9), importance sampling[46] enables current policy $\pi_i(\theta)$ to be trained on the history data sampled by $\pi_i(\hat{\theta})$. The importance weight of tuple (s_h,a_h) is calculated by the rate $\pi_i(a_h|s_h;\theta)/\pi_i(a_h|s_h;\hat{\theta})$. Different with $R_i(\tau)$, the advantage function $A_i(s_h,a_h;\hat{\theta})$ estimates the advantage of action a_h under state s_h. Here we use the Generalized Advantage Estimator (GAE) and for more details see Ref. [47]. Then, the second condition of choosing $u_i^{\text{clip},\hat{\theta}}(s_h,a_h;\,\theta)$ defined in Eq. (10) is just adding the clipping operation and $\epsilon\in(0,1)$ is the parameter to control the clipping range. The operation avoids the extreme importance weight when $\pi_i(\theta)$ and $\pi_i(\hat{\theta})$ are quite different. Intuitively, taking the minimum between $u_i^{\hat{\theta}}(s_h,a_h;\theta)$ and $u_i^{\text{clip},\hat{\theta}}(s_h,a_h;\,\theta)$ prevents the new policy from deviating the old one too much, in order to stabilize the training process on $J_i^{\hat{\theta}}(\theta)$.

Based on Eq. (8), the gradient of $J_i^{\hat{\theta}}(\theta)$ is given by

$$\nabla J_i^{\hat{\theta}}(\theta)=E_{(s_h,a_h)\sim\pi_i(\hat{\theta})}[g_i^{\hat{\theta}}(s_h,a_h;\theta)] \tag{11}$$

where $g_i^{\hat{\theta}}(s_h,a_h;\theta)$ is defined as

$$g_i^{\hat{\theta}}(s_h,a_h;\,\theta):=\begin{cases}\dfrac{\nabla\pi_i(a_h|s_h;\,\theta)}{\pi_i(a_h|s_h;\,\hat{\theta})}A_i(s_h,a_h;\,\hat{\theta}), & u_i^{\hat{\theta}}\leqslant u_i^{\text{clip};\hat{\theta}};\\ \text{Zero}, & u_i^{\hat{\theta}}>u_i^{\text{clip},\hat{\theta}}\end{cases} \tag{12}$$

In the above formula, the variable parts of $u_i^{\hat{\theta}}(s_h,a_h;\theta)$ and $u_i^{\text{clip},\hat{\theta}}(s_h,a_h;\,\theta)$ are omitted to simplify the notation. When $u_i^{\hat{\theta}}\leqslant u_i^{\text{clip},\hat{\theta}}$, Formula (12) calculates the actual gradient of $u_i^{\hat{\theta}}(s_h,a_h;\theta)$. Otherwise, it equals zero because θ is excluded from $u_i^{\text{clip},\hat{\theta}}(s_h,a_h;\,\theta)$, in this case, which equals $(1-\epsilon)A_i(s_h,a_h;\hat{\theta})$ or $(1+\epsilon)A_i(s_h,a_h;\hat{\theta})$. Similar with $\nabla J_i^{\hat{\theta}}(\theta)$, the Hessian $\nabla^2 J_i^{\hat{\theta}}(\theta)$ is given by

$$\nabla^2 J_i^{\hat{\theta}}(\theta)=E_{(s_h,a_h)\sim\pi_i(\hat{\theta})}[v_i^{\hat{\theta}}(s_h,a_h;\theta)] \tag{13}$$

where $v_i^{\hat{\theta}}(s_h,a_h;\theta)$ is defined as

$$v_i^{\hat{\theta}}(s_h,a_h;\,\theta):=\begin{cases}\dfrac{\nabla^2\pi_i(a_h|s_h;\,\theta)}{\pi_i(a_h|s_h;\,\hat{\theta})}A_i(s_h,a_h;\,\hat{\theta}), & u_i^{\hat{\theta}}\leqslant u_i^{\text{clip},\hat{\theta}};\\ \text{Zero}, & u_i^{\hat{\theta}}>u_i^{\text{clip},\hat{\theta}}\end{cases} \tag{14}$$

Take the new definition of $J_i^{\hat{\theta}}(\theta)$ into Eqs. (7) and (3), $F_i(\theta)$ can be written as

$$F_i(\theta):=-J_i^{\Psi_i(\theta)}(\Psi_i(\theta)) \tag{15}$$

where $\Psi_i(\theta)$ represents the model parameters after one-step inner adaption which is defined as

$$\Psi_i(\theta):=\theta+\alpha\nabla J_i^{\theta}(\theta) \tag{16}$$

where α is the learning rate of inner adaption and $\nabla J_i^{\theta}(\theta)$ is the gradient of policy $\pi_i(\theta)$. Both $J_i^{\Psi_i(\theta)}(\Psi_i(\theta))$ and $J_i^{\theta}(\theta)$ are calculated on the data sampled by the current policy $\Psi_i(\theta)$ and θ, which means the importance weight $\pi_i(a_h|s_h;\theta)/\pi_i(a_h|s_h;\hat{\theta})=1$. When it comes to multi-step local training, the weight will no longer be constant. We will discuss this situation in Section 3.2.

To minimize the local objective function $F_i(\theta)$, the gradient is necessary to perform gradient descent. Based on Eq. (15), $\nabla F_i(\theta)$ is given by

$$\nabla F_i(\theta)=-(I+\alpha\nabla^2 J_i^{\theta}(\theta))\nabla J_i^{\Psi_i(\theta)}(\Psi_i(\theta)) \tag{17}$$

where the gradient $\nabla J_i^{\Psi_i(\theta)}(\Psi_i(\theta))$ are calculated by Eq. (11) and the Hessian $\nabla^2 J_i^{\theta}(\theta)$ is calculated by Eq. (13).

3.2 Algorithm

Considering the computation of actual $\nabla J_i^{\hat{\theta}}(\theta)$ is intractable, we sample a batch of trajectories $\mathcal{D}_i^{\hat{\theta}}=\{\tau_1,\tau_2,\ldots,\tau_n\}$ with policy $\hat{\theta}$ to obtain the estimate of gradient

$$\tilde{\nabla} J_i^{\hat{\theta}}(\theta,\mathcal{D}_i^{\hat{\theta}}):=\frac{1}{|\mathcal{D}_i^{\hat{\theta}}|H}\sum_{\tau\in\mathcal{D}_i^{\hat{\theta}}}\sum_{h=0}^{H} g_i^{\hat{\theta}}(s_h^{\tau},a_h^{\tau};\theta) \tag{18}$$

where the tuple (s_h^{τ},a_h^{τ}) represents the h-th step of agent in trajectory τ and H is the fixed time horizon. The estimate of Hessian $J_i^{\hat{\theta}}(\theta)$ can be similarly calculated on $\mathcal{D}_i^{\hat{\theta}}$ as

$$\tilde{\nabla}^2 J_i^{\hat{\theta}}(\theta,\mathcal{D}_i^{\hat{\theta}}):=\frac{1}{|\mathcal{D}_i^{\hat{\theta}}|H}\sum_{\tau\in\mathcal{D}_i^{\hat{\theta}}}\sum_{h=0}^{H} v_i^{\hat{\theta}}(s_h^{\tau},a_h^{\tau};\theta) \tag{19}$$

For the conventional FL process (e.g., FedAvg), at the k-th aggregation round, the server will distribute the global policy θ^k to all clients. Then, before uploading models for the aggregation, these clients usually perform multi-step training locally to reduce communication overhead. In this situation, we define $\theta_i^{k,t}(t\geqslant 0)$ as the model of client i after t local steps. So the stochastic gradient of $F_i(\theta_i^{k,t})$ is calculated as

$$\begin{aligned}\tilde{\nabla} F_i(\theta_i^{k,t}):=&-(I+\alpha\tilde{\nabla}^2 J_i^{\theta^k}(\theta_i^{k,t},\mathcal{D}_i^{\theta^k}))\cdot\\ &\tilde{\nabla} J_i^{\Psi_i(\theta^k)}(\theta_i^{k,t}+\alpha\tilde{\nabla} J_i^{\theta^k}(\theta_i^{k,t},\mathcal{D}_i^{\theta^k}),\mathcal{D}_i^{\Psi_i(\theta^k)})\end{aligned} \tag{20}$$

Note that no matter how many steps client i performs, there are always only two sampled batches: $\mathcal{D}_i^{\theta^k}$ and $\mathcal{D}_i^{\Psi_i(\theta^k)}$. The former is sampled by the global policy θ^k

and the latter is sampled by the policy after one-step inner adaption. When t equals zero, $\theta_i^{k,t}=\theta^k$. It corresponds to the situation where the client loads the global policy θ^k and has not performed any local steps. When $t\geqslant 1$, the policy to be optimized and the policy sampled data are no longer the same, which means the importance weight $\pi_i(a_h|s_h;\theta)/\pi_i(a_h|s_h;\hat{\theta})\neq 1$. According to the stochastic gradient in Eq. (20), the local model is updated as

$$\theta_i^{k,t+1}=\theta_i^{k,t}-\beta\tilde{\nabla}F_i(\theta_i^{k,t}) \tag{21}$$

where β is the learning rate of the local model.

The whole process of FMRL is described in Algorithm 1. At the k-th aggregation round, the server selects a subset of clients in a certain proportion $\lambda\in(0,1)$ (Line 2). The subset is defined as $C^k\subset C$ and $|C^k|=\lambda|C|$. Then, the server distributes the global model θ^k to the clients in C^k (Line 3). For client i, it firstly load the global model $\theta_i^{k,0}=\theta^k$ and samples the batch $\mathcal{D}_i^{\theta^k}$ and $\mathcal{D}_i^{\Psi_i(\theta^k)}$ (Lines 5–8). After T steps of local update (Lines 9–12), the client i uploads the model change given by

$$\Delta\theta_i^k:=\theta_i^{k,T}-\theta^k \tag{22}$$

to the server (Lines 13 and 14). In the aggregation phase (Lines 16–18), we utilize the Fed-Adam[29, 30] as the aggregation optimizer. Specifically, the server treats the mean changes of clients $\sum_{i\in C^k}\Delta\theta_i^k$ as the pseudo-gradient of θ^k. Then optimize θ^k with Adam where β_1 and β_2 are the decay parameters. We define $\Delta\theta^{-1}=0$ and $z^{-1}=0$. The details of the aggregator optimizer are discussed in Section 3.3.

Algorithm 1 Federated meta reinforcement learning

```
Require: Initial global policy parameters $\theta^0$;
1: for aggregation round $k=0$ to $K-1$ do
2:     Select a subset of clients $C^k$ at random;
3:     Distribute the global model $\theta^k$ to all clients in $C^k$;
4:     for each client $i\in C^k$ do
5:         Load the global policy $\theta_i^{k,0}=\theta^k$;
6:         Sample the batch of trajectories $\mathcal{D}_i^{\theta^k}$;
7:         Perform the inner adaption to compute $\Psi_i(\theta^k)$;
8:         Sample the batch of trajectories $\mathcal{D}_i^{\Psi_i(\theta^k)}$;
9:         for local step $t=0$ to $T-1$ do
10:            Compute the stochastic gradient $\nabla F_i(\theta_i^{k,t})$ based on Eq. (20);
11:            Update the model $\theta_i^{k,t}$ based on Eq. (21)
12:        end for
13:        Compute the model change $\Delta\theta_i^k:=\theta_i^{k,T}-\theta^k$;
14:        Upload $\Delta\theta_i^k$ to the server;
15:    end for
16:    $\Delta\theta^k=\beta_1\Delta\theta^{k-1}+\beta_1(\frac{1}{|C^k|}\sum_{i\in C^k}\Delta\theta_i^k)$;
17:    $z^k=\beta_2 z^{k-1}+(1-\beta_2)(\Delta\theta^k)^2$;
18:    Update the global model $\theta^{k+1}=\theta^k+\eta\frac{\Delta\theta^k}{\sqrt{z^k}+\kappa}$;
19: end for
20: return $\theta^K$;
```

3.3 Aggregation optimizer

In this section, we give the general form of the aggregation optimizer and compare the difference in aggregation optimizers between FedAvg, FMRL, and other PFL methods.

In FedAvg, the uploaded models $\{\theta_i^k\},\forall i\in C^k$ are simply averaged to obtain the new global model θ^{k+1}. The aggregation is given by

$$\theta^{k+1}=\frac{1}{|C^k|}\sum_{i\in C^k}\theta_i^{k,T} \tag{23}$$

where $\theta_i^{k,T}$ represents the local model after T local steps of training in client i. To apply different aggregation optimizers on the server, we reformulate the updated local model $\theta_i^{k,T}=\theta^k+\Delta\theta_i^k$. So, the aggregation can be re-written as

$$\theta^{k+1}=\theta^k+\frac{1}{|C^k|}\sum_{i\in C^k}\Delta\theta_i^k \tag{24}$$

The above aggregation form is common in the PFL methods which maintain the single global model[36, 42].

To accelerate the model convergence, some PFL methods[38] used another aggregation form given by

$$\theta^{k+1}=(1-\eta)\theta^k+\frac{\eta}{|C^k|}\sum_{i\in C^k}\theta_i^{k,T} \tag{25}$$

which can be re-written as

$$\theta^{k+1}=\theta^k+\frac{\eta}{|C^k|}\sum_{i\in C^k}\Delta\theta_i^k \tag{26}$$

Obviously, Eq. (24) is included in Eq. (26) when learning rate $\eta=1$ and η can be larger to accelerate the aggregation. The above form can be seen as the federated version of the Stochastic Gradient Descent (Fed-SGD) optimizer.

For FMRL, Fed-Adam is utilized as the aggregate optimizer which is given by

$$\theta^{k+1}=\theta^{k}+\eta\frac{\Delta\theta^{k}}{\sqrt{z^{k}}+\kappa} \tag{27}$$

where the first moment $\Delta\theta^k$ is calculated as

$$\Delta\theta^{k}=\beta_1\Delta\theta^{k-1}+(1-\beta_1)\left(\frac{1}{|C^k|}\sum_{i\in C^k}\Delta\theta_i^k\right) \tag{28}$$

and the second raw moment z^k is calculated as

$$z^{k}=\beta_2 z^{k-1}+(1-\beta_2)(\Delta\theta^k)^2 \tag{29}$$

In the above formulas, β_1 and β_2 are the exponential decay rates of the moving averages $\Delta\theta^k$ and z^k, respectively. η is the learning rate of the aggregation optimizer and κ is a small value that controls the degree of adaptability.

In most circumstances, Fed-SGD is an ideal aggregation optimizer when $\eta=1$ (i.e., FedAvg). However, for reinforcement learning, the model training is sensitive to the value of the learning rate, and finding an appropriate learning rate is laborious. The utilization of Fed-SGD with fixed η may cause the global model to converge slowly. The recent reinforcement learning methods usually use different optimizers (momentum, Adam, etc.) for better performance[28]. However, applying Adam on the client side directly has two problems. (1) Due to different personalized tasks, the update directions and degrees of different local models may diverge a lot, which means the convergence of the global model may not be guaranteed. In contrast, the convergence of Fed-Adam on the server side is theoretically guaranteed[30]. (2) Adam needs to record additional optimizer parameters (first moment $\Delta\theta^k$ and second raw moment z^k), which brings the extra communication overhead for the aggregation of these optimizer parameters. Therefore, we use Fed-Adam on the service side instead of using Adam on the client side.

4 Numerical Experiment

In this section, we validate the effectiveness of FMRL in the personalized setting with extensive experiments. Firstly, we evaluate the overall performance of FMRL under different environments with personalized tasks. Then, we analyze the effect of the involving hyperparameters in FMRL. Afterwards, we compare the efficiency improvement brought by different federated optimizers: Fed-Adam and Fed-SGD. Finally, we explore the adaptability of FMRL for clients with new tasks.

4.1 Experimental setup

We consider a locomotion problem (including four environments) with the MuJoCo simulator[48]. Details of the four environments are as follows:

- **Half-cheetah random direction**: The simulated planar cheetah needs to run faster in a target direction $d\in\{1,-1\}$ (forward or backward). The reward is the velocity in the target direction minus the control cost.
- **Half-cheetah random velocity**: The same cheetah needs to run at the target velocity $v\in[0,2]$ which is sampled in the uniform distribution. The reward is the negative of the absolute difference between its current and the target velocity minus the control cost.
- **Ant random direction**: The simulated 3D quadruped (the “ant”) needs to run faster in a target direction $d\in\{1,-1\}$ (forward or backward). The reward is the velocity in the target direction. The reward is similar to half-cheetah random direction.
- **Ant random velocity**: The same ant needs to run at the target velocity $v\in[0,3]$ which is sampled in the uniform distribution. The reward is similar to half-cheetah random velocity.

The parameter settings of the above four environments and the subsequent settings of reinforcement learning refer to Ref. [27]. These environments can be divided into two categories: direction environments and velocity environments, corresponding to discrete personalized tasks with limited types but large variations and continuous personalized tasks with infinite types but minor variations.

We set $H=200$ for two half-cheetah environments and $H=100$ for ant environments. In each sampling, we sample $|\mathcal{D}_i^{\theta^k}|=|\mathcal{D}_i^{\Psi_i(\theta^k)}|=20$ episodes for both inner adaption and local training. Besides, we set $\alpha=0.1$ as the learning rate of the inner adaption and $\beta=0.001$ as the learning rate of the local training. In particular, to avoid the high computation overhead of Hessian $\nabla J_i^{\hat{\theta}}(\theta)$, we use the first-order approximation of $\nabla F_i(\theta)$ which is similar to the First-Order MAML (FO-MAML) proposed in Ref. [27]. For the setting of FL, we set $|C|=200$ clients with different personalized tasks and $K=500$ aggregation rounds. In each round, $\lambda|C|$ clients are selected with $\lambda=0.2$ to run $T=5$ steps of local training. The optimizer on the client side is

SGD with $\beta = 0.001$ and the aggregation optimizer is Fed-Adam with $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\kappa = 1 \times 10^{-8}$.

4.2 Overall performance

To evaluate the empirical performance of FMRL, We consider:

- **FedAvg**[7], the standard method in the federated setting. It averages the local models of the clients during the model aggregation and distributes the aggregated model to all clients for next-step training.
- **pFedMe**[38], one of the PFL methods based on the model regularization. It attaches a new regularization with Moreau envelopes, controlling the degree to which the local model deviates from the global model during the training.
- **FedRep**[37], one of the PFL methods based on the mixtures of the global and local models. It keeps high-level model layers local and only shares the low-dimensional representation layers of model aggregation.
- **Ditto**[40], one of the PFL methods based on model regularization, which trains global and local models separately. The global model is trained just like FedAvg and its parameters instruct the training of the local model.

The episode rewards over 5 random seeds along with 95% confidence intervals are reported in Table 1. For FedRep and Ditto, since each client has its own local model, the first line of each environment (global/personalized model) in Table 1 represents the local model performance after 0 training step. Meanwhile, all the above methods are equipped with Fed-Adam instead of Fed-SGD for a fair comparison.

In the performance comparison, we set $\eta = 0.0005$ for half-cheetah random direction and $\eta = 0.001$ for others. In particular, to accelerate the training of Ditto which needs to train the local model independently, we set the local learning rate $\alpha = 0.01$ for it and $\alpha = 0.001$ for the others. The remaining settings are the same as those in Section 4.1. Since the objective of FMRL is to obtain the personalized model after single or multiple steps of local training, the performance of its global model is lower than other methods in three environments. In all four environments, the local model of FMRL after 1, 2, and 3 steps outperforms the others on the gained rewards. For the global model, due to the existence of personalized tasks, five methods perform similarly except Ditto which has the personalized model. It is worth mentioning that FMRL achieves great rewards in direction environments: half-cheetah random direction and ant random direction. In contrast, there is almost no improvement from the global model to the optimized local model for FedAvg and pFedMe, which indicates they are limited when the personalized tasks of clients are quite different or even the opposite. In ant random direction, pFedMe even underperforms FedAvg. We believe that the additional model

Table 1 Mean episode rewards of the global model and local models after 1, 2, and 3 steps of inner adaption on all clients.

Environment	Model	Mean episode reward				
		FMRL	pFedMe	FedAvg	FedRep	Ditto
Half-cheetah random direction	Global/personalized model	−14.59 ± 5.44	−6.33 ± 5.06	−2.49 ± 1.43	5.53 ± 7.76	**161.13 ± 4.14**
	Local model after 1 step	**457.02 ± 30.82**	−5.07 ± 2.88	0.10 ± 1.33	24.06 ± 25.88	163.94 ± 4.02
	Local model after 2 steps	**480.96 ± 23.73**	−2.15 ± 1.04	2.76 ± 1.98	38.60 ± 33.21	166.72 ± 3.82
	Local model after 3 steps	**484.26 ± 29.51**	−0.85 ± 2.63	5.68 ± 3.58	51.67 ± 34.17	169.52 ± 3.90
Half-cheetah random velocity	Global/personalized model	**−112.86 ± 4.87**	−113.47 ± 2.40	−113.46 ± 1.84	−117.62 ± 5.49	−195.29 ± 24.19
	Local model after 1 step	**−94.35 ± 10.37**	−107.12 ± 3.59	−109.73 ± 2.21	−112.74 ± 5.56	−195.26 ± 24.22
	Local model after 2 steps	**−82.43 ± 13.35**	−101.10 ± 4.74	−106.15 ± 2.43	−107.96 ± 6.16	−195.14 ± 24.33
	Local model after 3 steps	**−74.53 ± 14.81**	−95.91 ± 5.67	−102.59 ± 2.37	−103.76 ± 6.29	−195.05 ± 24.22
Ant random direction	Global/personalized model	2.08 ± 2.53	4.84 ± 0.19	4.65 ± 0.21	5.53 ± 0.36	**43.42 ± 0.60**
	Local model after 1 step	**123.17 ± 9.65**	5.30 ± 0.30	5.05 ± 0.25	7.54 ± 0.88	43.93 ± 0.89
	Local model after 2 steps	**127.50 ± 10.22**	5.64 ± 0.41	5.26 ± 0.44	8.41 ± 0.69	44.43 ± 0.56
	Local model after 3 steps	**126.82 ± 11.32**	6.32 ± 0.78	5.57 ± 0.82	8.91 ± 0.55	44.79 ± 0.66
Ant random velocity	Global/personalized model	19.48 ± 3.51	**20.56 ± 2.28**	19.42 ± 2.08	12.34 ± 3.97	−28.86 ± 16.62
	Local model after 1 step	**44.30 ± 5.42**	24.64 ± 3.48	23.91 ± 2.88	14.51 ± 3.80	−28.72 ± 16.48
	Local model after 2 steps	**44.85 ± 5.56**	27.20 ± 4.11	27.14 ± 4.56	16.63 ± 3.96	−28.30 ± 16.57
	Local model after 3 steps	**45.18 ± 5.50**	28.59 ± 5.50	29.63 ± 5.72	18.29 ± 3.65	−28.08 ± 16.29

regularization could fail and hurt the model performance instead when the personalized tasks diverge a lot. And for FedRep and Ditto, their personalized models reach a higher performance than FedAvg and pFedMe, which implies that the models indeed capture the personalized tasks of clients. However, the improvement brought by the subsequent optimization of these models is far less than FMRL. In velocity environments (i.e., half-cheetah random velocity and ant random velocity), with the increase of local optimization steps, the gap between FMRL and other methods is gradually widening. It shows that FMRL learns the generalized policy and finds more favorable parameters for further fine-tuning simultaneously.

The detailed training curves are depicted in Fig. 1 where the upper row represents the performance of the global model and the lower one represents the performance of the local model after 1 step. Except for FedRep, for those methods which fully or partially share the model, the global model has a large oscillation on the gained rewards during the training, especially in half-cheetah random direction. One of the reasons is that the task distributions of the selected clients are different at each round, which makes the global model may not match the current personalized tasks, performing unsteadily. Besides, the tendentiousness of the model is affected by the degree of task deviations, reflected in the wider error bands of direction environments than velocity environments. For the local model after 1 step inner adaption, matched to Table 1, FMRL far surpasses other methods in direction environments. The rewards of the local models of FedAvg and pFedMe hover around the *x*-

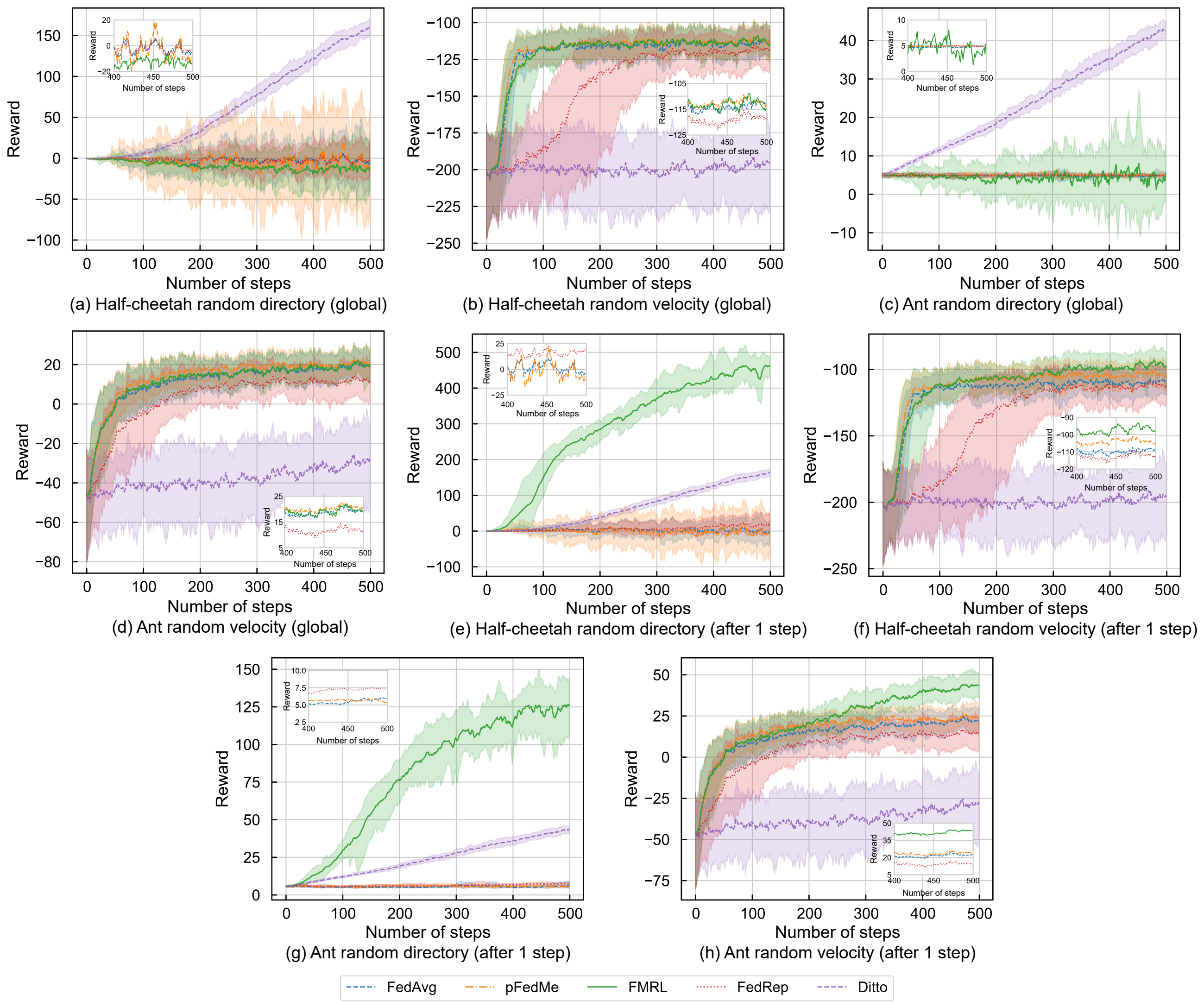


Fig. 1 Performance comparison of FedAvg, pFedMe, and FMRL on the randomly selected clients.

axis throughout the training, similar to the performance of the global model. And FedRep and Ditto gain varying improvements after 1 step optimization. In velocity environments, FMRL obtains a relatively higher reward than other methods. It seems that FedAvg, pFedMe, and FedRep grasp the average goal for continuous personalized tasks. However, the model only embeds knowledge of robot controlling (generalized policy) but no oriented task representation (personalized policy). Specially, Ditto has barely improved after 500 epochs of training. Considering FedRep, which is also underperforming, it seems that the personalization impedes the convergence of the model and its degree is directly proportional to the degree of personalization. Due to the random selection in the federated learning, the sampling and training rounds of each client cannot support it to train their personalized model independently, resulting in slow convergence or even no convergence of the local model.

4.3 Effect of hyperparameters

In this section, we investigate the effect of some hyperparameters (learning rates η, β, and client selection rate λ) to the performance of FMRL.

4.3.1 Learning rate

Figures 2 and 3 illustrate the effect of aggregation learning rate η and local learning rate β to the local model with one inner adaption, respectively.

In Fig. 2, β is fixed to 0.001 to observe the impact of η changing. According to Fig. 2, similar with Adam, the default value $\eta = 0.001$ reaches higher rewards except in half-cheetah random direction. In fact, the performance of $\eta = 0.001$ in two direction environments are both unstable. It is beneficial to choose a smaller aggregation learning rate for discrete personalized tasks and the value can be appropriately increased for the continuous personalized tasks in the early stages to accelerate the FL process.

Under the appropriate aggregation learning rate, the effect of local learning rate β is relatively small, which is verified in Fig. 3. We select η that performs the best in Fig. 2 (i.e., $\eta = 0.0003$ for the first and $\eta = 0.001$ for the other environments). Setting local learning rate $\beta = 0.001$ is still a good choice in most cases. Note that the changes of β bring a large difference in ant random directory, which is caused by the large aggregation learning η. This is consistent with the above conclusion that smaller η is more suitable for discrete personalized tasks.

In general, η and β should be adjusted in inverse proportion to reach the balance of the convergence speed and the training stability.

4.3.2 Client selection rate

In FL, the presence of stragglers is non-negligible

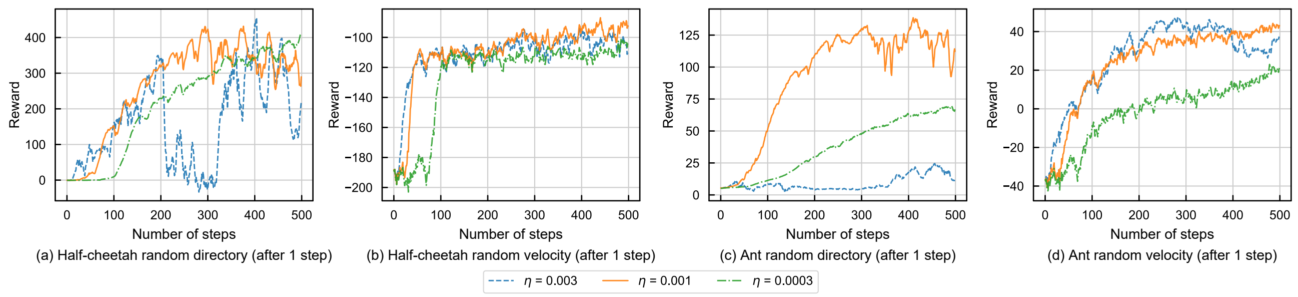


Fig. 2 Effect of aggregation learning rates on the local model.

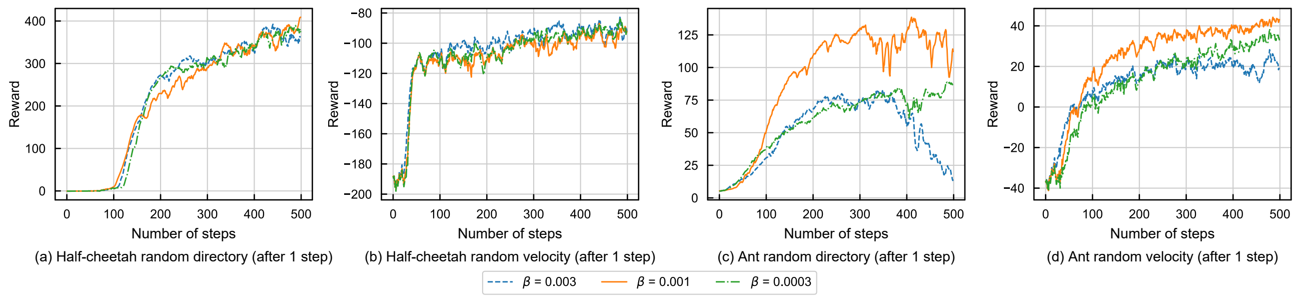


Fig. 3 Effect of local learning rates on the local model.

which causes the number of clients actually participating in the aggregation to be lower than expected. On this level, the degree of the effect of client selection rate λ indirectly reflects the method robustness in FL. Figure 4 shows the rewards of the clients selected for aggregation each round (upper) and the rewards of all clients each 100 rounds (lower) with different values of λ. Changes in λ have different effects on direction and velocity environments. Compared to velocity environments, the low selection rate ($\lambda = 0.1$) is more destructive to the convergence of the model in direction environments. It reduces the final rewards and makes the training process unstable. Since the subset of clients for the aggregation at each round is randomly selected, the task distribution of the subset may be greatly different from the overall task distribution, which is reflected in high variances, especially when the selection rate of clients λ is small. For the velocity environments where personalized tasks are continuous, the differences in task distribution are alleviated to a certain extent. Besides, although the final reward boost is less, a high selection rate ($\lambda = 0.3$) accelerates the aggregation in more complicated environments (ant random direction and ant random velocity).

4.4 Effect of aggregation optimizers

To evaluate the improvement of training efficiency brought by Fed-Adam, we compare Fed-Adam with Fed-SGD under different η and β in Fig. 5. When $\eta = 1$, the Fed-SGD behaves the same with FedAvg. Here we use the single combination of the learning rate for Fed-Adam. When the local learning rate is small ($\beta = 0.001$), the rewards of Fed-SGD almost do not improve. Even the aggregation learning rate η is increased, there are few changes in the rewards. Under the same β, the rewards of Fed-Adam grow rapidly in the early stages of training. To highlight the influence of η, we study the performance of Fed-SGD with a larger local learning rate $\beta = 0.01$ as well. When $\eta = 5$, the rewards of Fed-SGD presents a large improvement during the training. However, as the training progresses, the rewards become unstable and back down to the low in half-cheetah random direction and ant random direction. Though the large β accelerates the federation process, it also causes huge deviations between different local models. In this case, the aggregated global model may deviate from the optimization goal of all clients, and the deviation is amplified by the large aggregation learning rate η. Compared with the continuous tasks, the discrete tasks with high variations are more affected by the model deviation, which explains the poorer performance of Fed-SGD in direction environments.

In summary, the utilization of Fed-Adam greatly improves the efficiency and performance of FMRL compared to Fed-SGD. Meanwhile, the above analysis shows that it is difficult to find an appropriate combination of the local and aggregation learning rates for Fed-SGD. Compared with Fed-SGD, Fed-Adam does not require much effort to adjust the learning rates.

4.5 Scalability

In addition to the clients involved, we also study the performance of FMRL on the clients that never

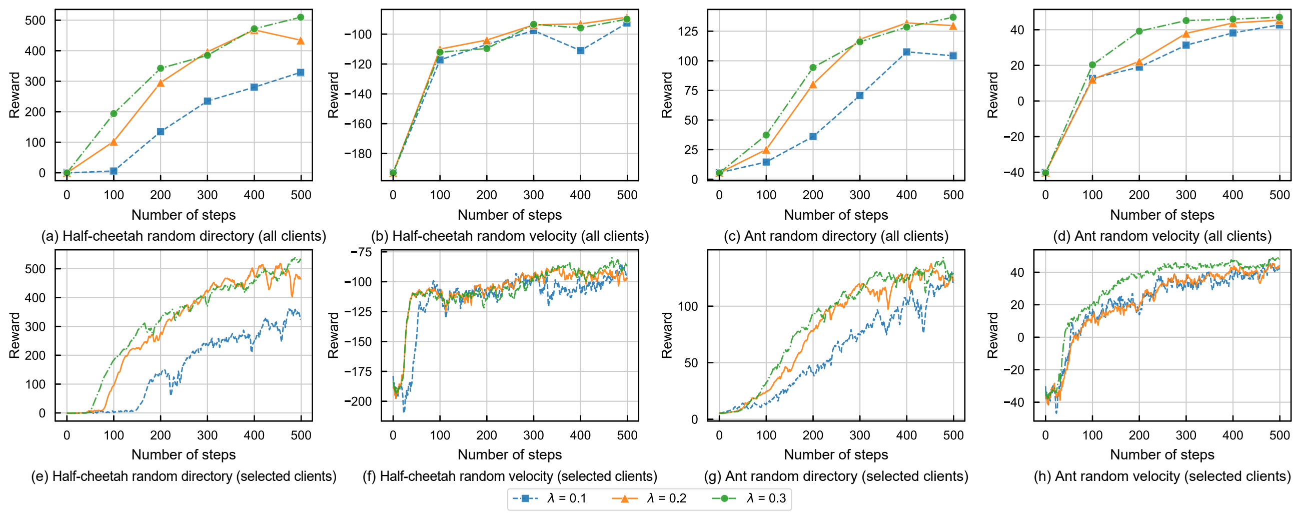


Fig. 4 Effect of client selection rates on the local model of selected and all clients.

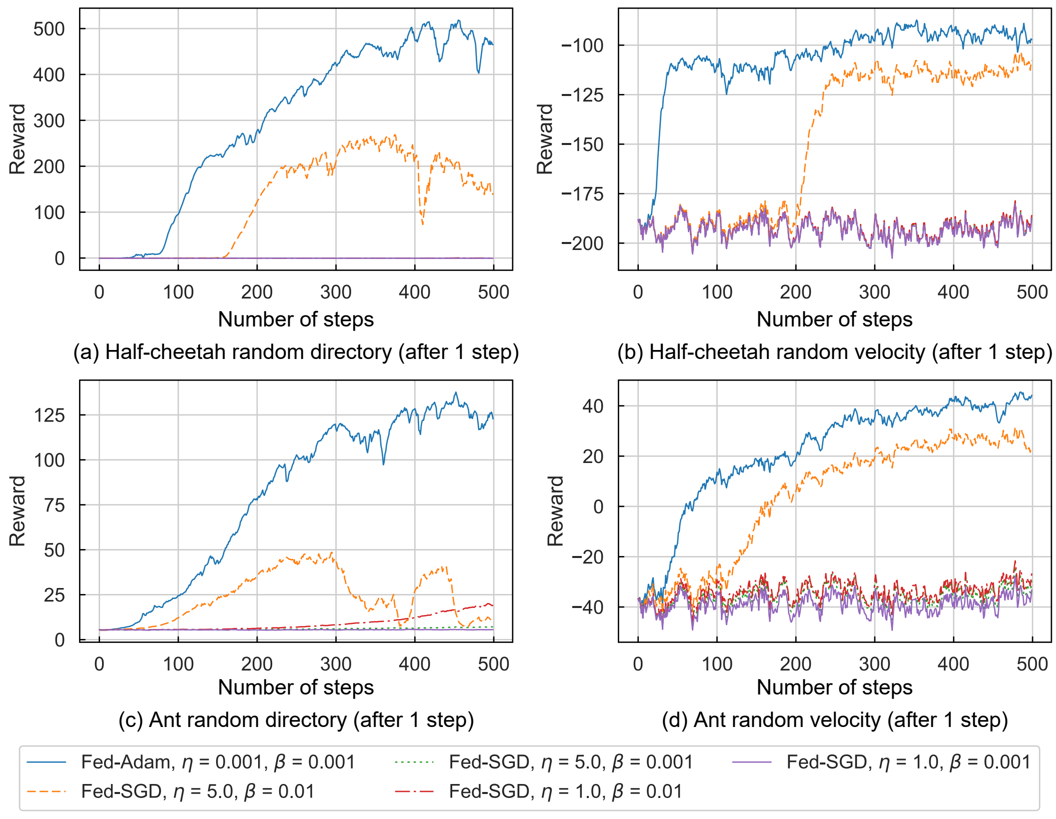


Fig. 5 Effect of aggregation optimizers with different learning rates.

participate in the federation process before. The result in Fig. 6 shows that FMRL can adapt to new personalized tasks without additional rounds of federation training. Thereinto, we set $\alpha = 0.1$ in the first step and $\alpha = 0.05$ in the subsequent steps. In direction environments, FMRL also performs well on new clients. And other methods have little improvement in the first five steps. Compared to discrete personalized tasks, the continuous tasks in velocity environments bring more challenges to the model training. Even so, the reward of FMRL still increases faster and is more stable than other methods. In fact, the methods based on personalized models such as FedRep and Ditto show little scalability on new clients due to the overly specialized personalized models, especially in complex environments. Beside, pFedMe performs even worse than FedAvg in ant random velocity. To obtain the personalized model, pFedMe executes additional local updates with customized regularization, which makes the global model only optimized for the participating clients, impairing its scalability on new clients.

5 Conclusion and Future Work

In this paper, we propose a method of federated meta reinforcement learning for personalized tasks, named FMRL, to solve the task shift problem in FRL. Considering the conflict between the communication overhead of aggregations and the sampling overhead of multi-step local training, FMRL leverages PPO to achieve the reuse of historical data for gradient and Hessian computation in MAML. Meanwhile, for better convergence speed and performance, we replace the Fed-SGD in FedAvg with Fed-Adam as the new aggregation optimizer. The results of numerical experiments corroborate that our approach outperforms other methods in different environments, including discrete and continuous personalized tasks, and can easily adapt to new clients. Future work is to analyze the effectiveness and efficiency of FMRL when various privacy mechanisms are applied.

Appendix

Acronym and Notation

For convenience, we list all acronyms and main notations in Tables A1 and A2. All notations with subscript i are related to client i, which will not be emphasized in Table A2 any more.

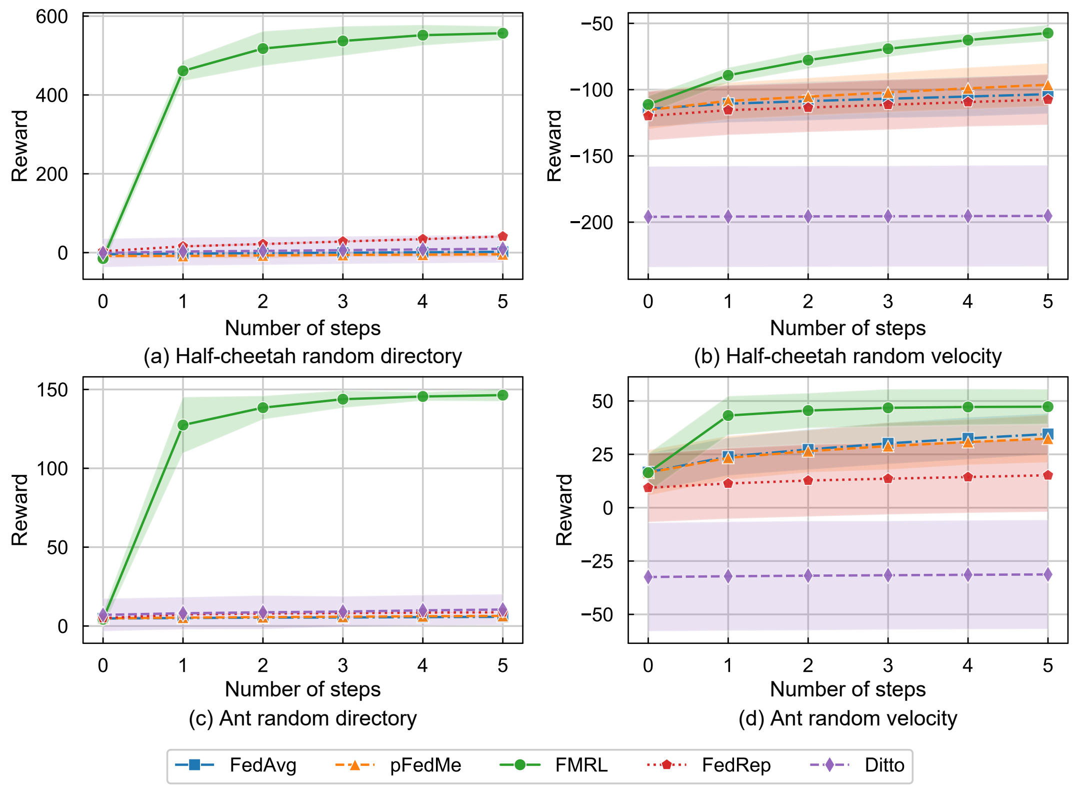


Fig. 6 Performance of the global model after inner adaptions on the new clients.

Table A1 Acronyms used in the paper.

Acronym	Full name
FL	Federated learning
FRL	Federated reinforcement learning
DRL	Deep reinforcement learning
Non-IID	Non-identically and independently distributed
PFL	Personalized federated learning
MAML	Model-agnostic meta-learning
PPO	Proximal policy optimization
Fed-Adam	Federated version of Adam
Fed-SGD	Federated version of stochastic gradient descent
MDP	Markov decision process

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2020YFB1707601), the Major Research Plan of the National Natural Science Foundation of China (No. 92267104), the Natural Science Foundation of Jiangsu Province of China (No. BK20211284), and the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps (No. 2020DB005).

References

[1] T. Ben-Nun and T. Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, *ACM Comput. Surv.*, vol. 52, no. 4, p. 65, 2019.

[2] M. Langer, Z. He, W. Rahayu, and Y. Xue, Distributed training of deep learning models: A taxonomic perspective, *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 12, pp. 2802–2818, 2020.

[3] L. Gu, M. Cui, L. Xu, and X. Xu, Collaborative offloading method for digital twin empowered cloud edge computing on Internet of vehicles, *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 433–451, 2023.

[4] X. Zhou, W. Liang, K. I. K. Wang, and L. T. Yang, Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations, *IEEE Trans. Comput. Soc. Syst.*, vol. 8, no. 1, pp. 171–178, 2021.

[5] Q. He, Z. Dong, F. Chen, S. Deng, W. Liang, and Y. Yang, Pyramid: Enabling hierarchical neural networks with edge computing, in *Proc. ACM Web Conf. 2022*, Virtual Event, Lyon, France, 2022, pp. 1860–1870.

[6] P. Voigt and A. V. D. Bussche, *The EU General Data Protection Regulation* (*GDPR*): *A Practical Guide*. Cham, Switzerland: Springer, 2017.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, Communication-efficient learning of deep networks from decentralized data, arXiv preprint arXiv: 1602.05629, 2016.

[8] H. H. Zhuo, W. Feng, Y. Lin, Q. Xu, and Q. Yang, Federated deep reinforcement learning, arXiv preprint

Table A2 Main notations used in the paper.

Notation	Description
τ	Trajectory consisting of the couples of states and actions
H	Time horizon of trajectory τ
γ	Discount factor for the calculation of trajectory reward
$q_i(\tau;\theta)$	Probability of trajectory τ with policy $\pi_i(\theta)$
$\pi_i(\theta)$	Policy with model θ
$\rho_i(s)$	Probability of initial state s
$\pi_i(a_h\|s_h;\theta)$	Probability of taking action a_h under state s_h with policy $\pi_i(\theta)$
$r_i(s_h,a_h)$	Single step reward of taking action a_h under state s_h
$P_i(s_{h+1}\|s_h,a_h)$	Transfer probability from state s_h to s_{h+1} when action a is taken
$q_i(\tau;\theta)$	Sampling probability of trajectory τ with policy $\pi_i(\theta)$
$R_i(\tau)$	Discounted total reward of trajectory τ
$J_i(\theta)$	Expected trajectory reward pf policy $\pi_i(\theta)$ with sampling probability $q_i(\cdot;\theta)$
$J_i^{\hat{\theta}}(\theta)$	Expected trajectory reward of policy $\pi_i(\theta)$ with PPO which utilizes the data sampled by policy $\pi_i(\hat{\theta})$
$A_i(s_h,a_h;\hat{\theta})$	Advantage function which estimates the advantage of action a_h under state s_h
$u_i^{\hat{\theta}}(s_h,a_h;\theta)$	Weighted advantage function whose weight is the rate of $\pi_i(\theta)$ and $\pi_i(\hat{\theta})$
$u_i^{\text{clip},\hat{\theta}}(s_h,a_h;\theta)$	Clipped $u_i^{\hat{\theta}}(s_h,a_h;\theta)$ with the clipping rage $(1-\epsilon,1+\epsilon)$
$\nabla J_i^{\hat{\theta}}(\theta)$	Gradient of $J_i^{\hat{\theta}}(\theta)$
$\nabla^2 J_i^{\hat{\theta}}(\theta)$	Hessian of $J_i^{\hat{\theta}}(\theta)$
$F_i(\theta)$	Objective function
α	Learning rate of inner adaption
$\Psi_i(\theta)$	Model parameter θ after one step inner adaption
$\tilde{\nabla} J_i^{\hat{\theta}}(\theta,\mathcal{D}_i^{\hat{\theta}})$	Estimate of gradient with the batch of trajectories $\mathcal{D}_i^{\hat{\theta}}$
$\tilde{\nabla}^2 J_i^{\hat{\theta}}(\theta,\mathcal{D}_i^{\hat{\theta}})$	Estimate of Hessian with the batch of trajectories $\mathcal{D}_i^{\hat{\theta}}$
$\tilde{\nabla} F_i(\theta_i^{k,t})$	Stochastic gradient of $F_i(\theta_i^{k,t})$
T	Steps of local training at a single aggregation round
θ^k	Global model θ at the k-th aggregation round
$\theta_i^{k,t}$	Local model θ_i after t step local training at the k-th aggregation round
$\Delta\theta^k$	Change of the global model at the k-th aggregation round
$\Delta\theta_i^k$	Change of the local model at the k-th aggregation round
β	Learning rate of local training
λ	Selection proportion of the clients for aggregation
C	Total client set
C^k	Selected client set at the k-th aggregation round
η	Learning rate of the aggregation optimizer
z^k	Raw moment similar to $\Delta\theta^k$
β_1	Exponential decay rates of the moving averages $\Delta\theta^k$
β_2	Exponential decay rates of the moving averages z^k
κ	Small value that controls the degree of adaptability

arXiv: 1901.08277, 2019.

[9] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, An introduction to deep reinforcement learning, *Found. Trends® Mach. Learn.*, vol. 11, nos. 3&4, pp. 219–354, 2018.

[10] X. Zhou, W. Liang, K. Yan, W. Li, K. I. K. Wang, J. Ma, and Q. Jin, Edge-enabled two-stage scheduling based on deep reinforcement learning for Internet of everything, *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3295–3304, 2022.

[11] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. C. Liang, and D. I. Kim, Applications of deep

reinforcement learning in communications and networking: A survey, *IEEE Commun. Surv. Tutor.*, vol. 21, no. 4, pp. 3133–3174, 2019.

[12] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al., QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation, arXiv preprint arXiv:1806.10293v3, 2018.

[13] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, Deep reinforcement learning for autonomous driving: A survey, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022.

[14] F. X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low, Fault-tolerant federated reinforcement learning with theoretical guarantee, arXiv preprint arXiv: 2110.14074, 2021.

[15] S. Liu, K. C. See, K. Y. Ngiam, L. A. Celi, X. Sun, and M. Feng, Reinforcement learning for clinical decision support in critical care: Comprehensive review, *J. Med. Internet Res.*, vol. 22, no. 7, p. e18477, 2020.

[16] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network, *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238–2251, 2021.

[17] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, Online collaborative data caching in edge computing, *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 281–294, 2021.

[18] L. Yuan, Q. He, F. Chen, J. Zhang, L. Qi, X. Xu, Y. Xiang, and Y. Yang, CSEdge: Enabling collaborative edge storage for multi-access edge computing based on blockchain, *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1873–1887, 2022.

[19] B. Liu, L. Wang, and M. Liu, Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems, *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4555–4562, 2019.

[20] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, Federated transfer reinforcement learning for autonomous driving, in *Federated and Transfer Learning*, R. Razavi-Far, B. Wang, M. E. Taylor, and Q. Yang, eds. Cham, Switzerland: Springer, 2023, pp. 357–371.

[21] C. Nadiger, A. Kumar, and S. Abdelhak, Federated reinforcement learning for fast personalization, in *Proc. 2019 IEEE Second Int. Conf. Artificial Intelligence and Knowledge Engineering* (*AIKE*), Sardinia, Italy, 2019, pp. 123–127.

[22] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, *Found. Trends® Mach. Learn.*, vol. 14, nos. 1&2, pp. 1–210, 2021.

[23] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, Federated optimization in heterogeneous networks, arXiv preprint arXiv: 1812.06127, 2018.

[24] Q. Li, Y. Diao, Q. Chen, and B. He, Federated learning on non-IID data silos: An experimental study, in *Proc. 2022 IEEE 38th Int. Conf. Data Engineering* (*ICDE*), Kuala Lumpur, Malaysia, 2022, pp. 965–978.

[25] V. Smith, C. K. Chiang, M. Sanjabi, and A. Talwalkar, Federated multi-task learning, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 4427–4437.

[26] J. Mills, J. Hu, and G. Min, Multi-task federated learning for personalised deep neural networks in edge computing, *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 630–641, 2022.

[27] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *Proc. 34th Int. Conf. Machine Learning - Volume 70*, Sydney, Australia, 2017, pp. 1126–1135.

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

[29] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014.

[30] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, Adaptive federated optimization, arXiv preprint arXiv: 2003.00295, 2020.

[31] C. Y. Chen, J. Ni, S. Lu, X. Cui, P. Y. Chen, X. Sun, N. Wang, S. Venkataramani, V. Srinivasan, W. Zhang, et al., ScaleCom: Scalable sparsified gradient compression for communication-efficient distributed training, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 13551–13563.

[32] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, Three approaches for personalization with applications to federated learning, arXiv preprint arXiv: 2002.10619, 2020.

[33] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, Personalized federated learning with first order model optimization, arXiv preprint arXiv: 2012.08565, 2020.

[34] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, Towards personalized federated learning, *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2022.3160699.

[35] F. Hanzely and P. Richtárik, Federated learning of a mixture of global and local models, arXiv preprint arXiv: 2002.05516, 2020.

[36] Y. Deng, M. M. Kamani, and M. Mahdavi, Adaptive personalized federated learning, arXiv preprint arXiv: 2003.13461, 2020.

[37] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, Exploiting shared representations for personalized federated learning, arXiv preprint arXiv: 2102.07078, 2021.

[38] C. T. Dinh, N. H. Tran, and T. D. Nguyen, Personalized federated learning with Moreau envelopes, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 21394–21405.

[39] Y. T. Huang, L. Y. Chu, Z. R. Zhou, L. J. Wang, J. C. Liu, J. Pei, and Y. Zhang, Personalized cross-silo federated learning on non-IID data, *Proc. AAAI Conf. Artif. Intell.*,

vol. 35, no. 9, pp. 7865–7873, 2021.

[40] T. Li, S. Hu, A. Beirami, and V. Smith, Ditto: Fair and robust federated learning through personalization, arXiv preprint arXiv: 2012.04221, 2020.

[41] M. Khodak, M. F. Balcan, and A. Talwalkar, Adaptive gradient-based meta-learning methods, arXiv preprint arXiv: 1906.02717, 2019.

[42] A. Fallah, A. Mokhtari, and A. Ozdaglar, Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 3557–3568.

[43] D. A. E. Acar, Y. Zhao, R. Zhu, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, Debiasing model updates for improving personalized federated training, presented at 38th Int. Conf. Machine Learning, Virtual Event, 2021.

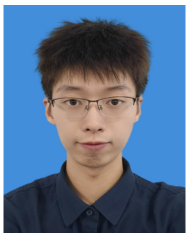
[44] A. Fallah, A. Mokhtari, and A. Ozdaglar, On the convergence theory of gradient-based model-agnostic meta-learning algorithms, arXiv preprint arXiv: 1908.10400, 2019.

[45] R. S. Sutton and A. G. Barto, *Reinforcement Learning*: *An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[46] S. T. Tokdar and R. E. Kass, Importance sampling: A review, *Wires Comput. Stat.*, vol. 2, no. 1, pp. 54–60, 2010.

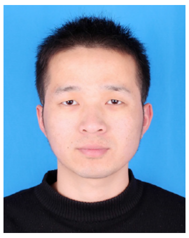
[47] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, High-dimensional continuous control using generalized advantage estimation, arXiv preprint arXiv: 1506.02438, 2015.

[48] E. Todorov, T. Erez, and Y. Tassa, MuJoCo: A physics engine for model-based control, in *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, 2012, pp. 5026–5033.



Wentao Liu is currently pursuing the MEng degree in computer science and technology at School of Computer Science, Nanjing University of Information Science and Technology, China. His research interests include federated learning and edge computing.

Jintao Wu received the PhD degree in computer science and technology from Anhui University, China, in 2020. He is currently a lecturer at School of Software, Nanjing University of Information Science and Technology, China. His research interests include service computing, edge computing, and computer vision.



Jielin Jiang received the PhD degree in pattern recognition and intelligence system from Nanjing University of Science and Technology, Nanjing, China in 2015. From February 2013 to August 2013, he was an exchange student with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, China. From July 2014 to July 2017, he was a research associate with Institute of Textiles and Clothing, Hong Kong Polytechnic University, Hong Kong, China. Now, he is an associate professor at School of Software, Nanjing University of Information Science and Technology, China. His current research interests include anomaly detection, image denoising, and edge computing.

Xiaolong Xu received the PhD degree in computer science and technology from Nanjing University, Nanjing, China, in 2016. He was a research scholar with Michigan State University, East Lansing, MI, USA, from 2017 to 2018. He is currently a full professor at School of Software, Nanjing University of Information Science and Technology, Nanjing, China. He has authored or co-authored more than 100 peer-review articles in international journals and conferences, including *IEEE TITS*, *IEEE TII*, *ACM TOIT*, *ACM TOMM*, *IEEE IOT*, IEEE TCC, *IEEE TBD*, *IEEE TCSS*, *IEEE TETCI*, IEEE ICWS, ICSOC, etc. Among them, 10 papers are selected as the ESI highly cited papers and 8 of them are selected as the Essential Science Indicators (ESI) hot paper. His research interests include edge computing, the Internet of Things (IoT), cloud computing, and big data. He is a fellow of European Alliance for Innovation (EAI). He was the recipient of the Best Paper Award from the CBD 2016, IEEE CPSCom 2020, and SPDE 2020, the distinguish paper award from EAI Cloudcomp 2019, the best student paper award from EAI Cloudcomp 2019, and the Best Session Paper Award from IEEE DSAA 2020.