Tomtit: Hierarchical Federated Fine-Tuning of Giant Models based on Autonomous Synchronization

Tianyu Qi[1], Yufeng Zhan[1,2], Peng Li[3], and Yuanqing Xia[1]
[1]School of Automation, Beijing Institute of Technology, Beijing, China
[2]Yangtze Delta Region Academy of Beijing Institute of Technology (Jiaxing), Jiaxing, China
[3]School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu, Japan
{qitianyu, yu-feng.zhan, xia_yuanqing}@bit.edu.cn, pengli@u-aizu.ac.jp

IEEE INFOCOM 2024 - IEEE Conference on Computer Communications | 979-8-3503-8350-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/INFOCOM52122.2024.10621369

***Abstract*—With the quick evolution of giant models, the paradigm of pre-training models and then fine-tuning them for downstream tasks has become increasingly popular. The adapter has been recognized as an efficient fine-tuning technique and attracts much research attention. However, adapter-based fine-tuning still faces the challenge of lacking sufficient data. Federated fine-tuning has been recently proposed to fill this gap, but existing solutions suffer from a serious scalability issue, and they are inflexible in handling dynamic edge environments. In this paper, we propose *Tomtit*, a hierarchical federated fine-tuning system that can significantly accelerate fine-tuning and improve the energy efficiency of devices. Via extensive empirical study, we find that model synchronization schemes (i.e., when edge servers and devices should synchronize their models) play a critical role in federated fine-tuning. The core of *Tomtit* is a distributed design that allows each edge and device to have a unique synchronization scheme with respect to their heterogeneity in model structure, data distribution and computing capability. Furthermore, we provide a theoretical guarantee about the convergence of *Tomtit*. Finally, we develop a prototype of *Tomtit* and evaluate it on a testbed. Experimental results show that it can significantly outperform the state-of-the-art.**

I. Introduction

Thanks to the great success of ChatGPT, large-scale language models (LLM) based on transformers have attracted great attention due to their amazing capability in handling various natural language processing tasks [1], [2]. Since the inception of LLM, models in various fields (e.g., computer vision) [3], [4] starts to evolve into giant ones to pursuit emergent abilities.

Since training giant models needs a huge amount of high-quality data and massive computing power, it is impossible for every task to train its giant model from scratch. Thus, a typical usage pattern of giant models is to pre-train models with sufficient generality and then fine-tune these models using data of specific tasks [5]. A popular fine-tuning approach is to add several adapters [6], [7], i.e., small modules consisting of several simple layers, into pre-trained models and train them while freezing other parameters. Since only the parameters of adapters are trainable, fine-tuning could be very efficient. However, fine-tuning for specific tasks is usually short of data. Therefore, several recent works [8], [9] have proposed to apply federated learning (FL) for adapter-based fine-tuning, which has succeeded greatly. Specifically, multiple devices can contribute their data for fine-tuning by training local models, which are periodically synchronized for knowledge sharing.

We desire more devices to join the federated fine-tuning to leverage more data for improving model accuracy. On the other hand, we are facing a critical scalability challenge that the efficiency of federated fine-tuning degrades seriously as more devices join it. Recently, hierarchical federated learning (HFL) [10]–[12] has been proposed to improve scalability by dividing devices into several learning groups led by edge servers, and each group individually conducts model aggregation and update. Then, edge servers periodically send their models to the cloud for global model update.

However, when HFL meets federated fine-tuning, there are several unique challenges unaddressed by existing work. First, although existing HFL considers device heterogeneity in computing capability and network conditions [13], [14], it applies the same synchronization scheme for devices within the same group, i.e., they conduct the same number of local training rounds before group model aggregation. However, devices may insert different numbers of adapters, which need different local training rounds. Such a feature should be considered into the synchronization scheme design of federated fine-tuning. Second, edge environment is dynamic. Devices may change their adapter settings [8], which need to be handled by different synchronization schemes. In addition, the distribution of data held by devices also changes. Existing works cannot well handle such dynamic of federated fine-tuning environment. Finally, since the number of devices could be enormous, making an appropriate synchronization scheme for each device has high complexity. Moreover, the convergence process of federated fine-tuning has no close-form expression, and traditional methods based on optimization can hardly work here.

In this paper, we propose *Tomtit*, a hierarchical federated fine-tuning system that can significantly accelerate federated fine-tuning and improve energy efficiency of devices. *Tomtit*'s superiority stems from a new synchronization scheme that fully exploits device heterogeneity of models, computing capability and network connections. Different from existing works, *Tomtit* allows devices and edge servers to have different adapter aggregation frequencies, which are determined by a highly efficient distributed algorithm based on multi-agent reinforcement learning (MARL). When applying traditional

MARL for *Tomtit*, we find that it suffers from slow convergence because of large state space. We address this weakness by designing a unique global state for the critic network and a special reward function based on convergence characteristics. The main contributions of this paper can be summarized as follows.

- We propose a hierarchical federated fine-tuning system called *Tomtit*, aiming to achieve higher model accuracy and lower energy consumption when fine-tuning giant models among heterogeneous devices.
- We design a distributed algorithm based on MARL to determine the aggregation frequency of edge servers and devices by respecting their heterogeneity in models and computing capability. The convergence is also proved.
- We develop a prototype of *Tomtit* and deploy it on a testbed for performance evaluation. Experimental results show that *Tomtit* can not only enhance model accuracy but also reduce energy consumption by up to 59.6%, compared to the state-of-the-art.

The rest of the paper is organized as follows: Section II shows the background and motivation of this paper. Section III provides detailed design of *Tomtit* and demonstrates the convergence of our proposed aggregation approach. Extensive experiments are conducted in Section IV, followed by a review of related work in Section V. Finally, Section VI concludes this paper.

II. Background and Motivation

In this section, we first briefly review the background of HFL and adapter-based fine-tuning. Then we conduct experimental study to motivate the design of *Tomtit*.

A. Hierarchical Federated Learning

To improve FL's scalability, HFL has been proposed by letting edge servers aggregate models of a subset of devices within its service range, to avoid frequently global model synchronization. Specifically, devices upload their models to edge servers for aggregation, and after a few training rounds, edge servers upload the aggregated models to the cloud.

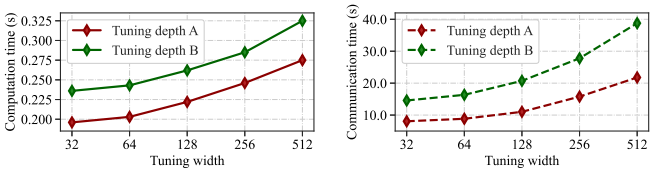
Given N devices, each device j trains its local model w_j using data samples $(x, y) \in \mathcal{D}_j$. Let $f(w_j, x, y)$ denote the loss function, which can be represented as

$$f_j(w_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} f_j(w_j, x, y), \tag{1}$$

where $|\mathcal{D}_j|$ denotes the number of samples. There are M edge servers in the HFL system, and each edge server i connects to a set of N_i devices, i.e., $\sum_{i=1}^{M} N_i = N$.

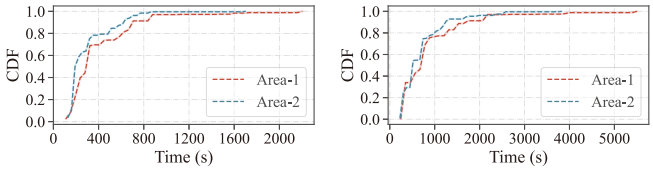
After conducting γ_d rounds of local SGD training, devices upload their models to edge servers for aggregation as

$$w_i^e = \sum_{j=1}^{N_i} \frac{|\mathcal{D}_j|\, w_j}{\sum_{j=1}^{N_i} |\mathcal{D}_j|}, \tag{2}$$



(a) Computation time (b) Communication time

Fig. 1. Overhead of adapter-based fine-tuning



(a) MobilBert (b) ALBert

Fig. 2. Device computing capability statistics under different edges

where w_i^e is the model maintained by edge server i. After γ_e times of aggregations, the edge server sends the model w_i^e to the cloud for further aggregation as

$$w = \sum_{i=1}^{M} \frac{|\mathcal{D}_i|\, w_i^e}{\sum_{i=1}^{M} |\mathcal{D}_i|}. \tag{3}$$

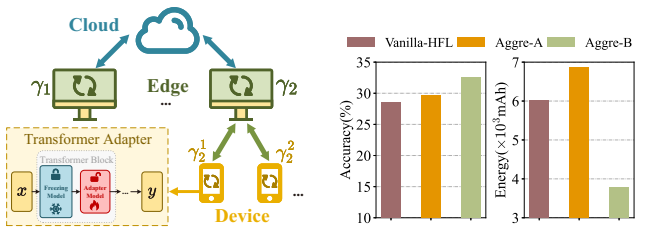
Note that when $\gamma_e = 1$, HFL becomes the traditional FL.

B. Influence of Adapters and Device Heterogeneity

Fine-tuning is a common operation when applying giant models to specific tasks. To reduce fine-tuning cost, adapters have been proposed and widely adopted [6]. An adapter is essentially a module consisting of several simple layers. By inserting some adapters into pre-trained giant models, we can obtain sufficient flexibility to adapt to domain-specific data while maintaining low training cost because only the parameters of adapters are trainable.

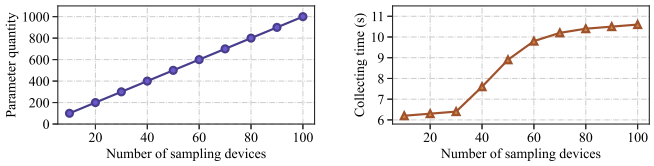
Adapters can be customized by adjusting depth and width. The depth determines how many adapters are inserted into the transformer model, while the width represents the adapter size, i.e., the projection dimension. Federated fine-tuning allows different devices to use different adapter configurations, leading to varying local training time, communication time, as well as convergence speed. This characteristic is verified by experiments using a transformer-based Bert model trained with the T-Rex dataset [15]. As shown in Fig. 1, both computing time and communication time increase as the growth of adapter widths. Meanwhile, increasing adapter depth from A to B leads to longer computing and communication time because more parameters are introduced.

We have also verified the influence of the computing capability of devices by studying real-world data collected from the Measuring Broadband America (MBA) project, which contains information of device location and types [16]. We use AIBenchmark to measure the corresponding devices' computational capability [17]. As shown in Fig. 2, the training time of devices under each area (base station considered as edge) follows a heavy-tailed distribution, indicating heterogeneity among devices at network and computation capability.



(a) Framework of hierarchical federated fine-tuning (b) Accuracy and energy efficienty of different schemes

Fig. 3. Aggregation frequency strategy



(a) State parameter quantity (b) State collecting time

Fig. 4. Overhead of centralized control in large-scale systems

Moreover, the device distribution varies among different base stations, suggesting heterogeneity between edges.

C. Influence of Model Synchronization Frequency

To study the influence of aggregation frequency, we build a testbed consisting of 50 Raspberry Pis as devices and 5 laptops as edges, running a hierarchical federated learning on the CIFAR-100 dataset. Each device is assigned sample data with different labels and trains adapters with different depth and width. We constrain the maximum utilization of the device CPU using stress-ng [18], ranging from 10% to 80%, to simulate hardware heterogeneity.

We design 2 synchronization schemes, namely *Aggre-A* and *Aggre-B*. In *Aggre-A*, synchronization frequencies of devices whose CPU utilization is less than 30% are set to 5, and others are 1. In *Aggre-B*, we conduct a fine-grained control by setting synchronization frequencies proportional to device CPU utilization. We compare these two schemes with *Vanilla-HFL* by running the same number of training rounds. As shown in Fig. 3(b), *Aggre-A* has higher accuracy than *Vanila-HFL* because it can effectively accelerate some stragglers. On the other hand, it incurs high energy consumption because of heavier training workloads. *Aggre-B* can balance training efficiency and energy consumption well, leading to the highest final accuracy and the lowest energy consumption.

The above results motivate us that a large optimization space exists when applying HFL for federated fine-tuning, which can be obtained if we can design a sophisticated synchronization scheme with fine-grained control over individual devices according to their model features and computing capability.

D. Centralized Control Challenges

In recent years, researchers have realized the importance of synchronization schemes in hierarchical federated learning and have proposed various solutions [19]–[22]. However, almost all of the existing works are based on centralized control that needs to collect global system states and run complex algorithms, which can hardly scale as more devices join the HFL.

We conduct an empirical study about the cost of centralized control by using the settings and methods proposed in [19]. As shown in Fig. 4(a), the number of system parameters collected for centralized control linearly increases as the growth of devices. Meanwhile, the time needed for decision making also increases to an unaffordable level, as shown in Fig. 4(b). Other recent works, e.g., [22], have a similar level of overhead and we do not include details due to space limit. Thus, we are motivated to design a distributed solution to address the bottleneck incurred by centralized control.

III. Design of Tomtit

In this section, we first present the framework of *Tomtit*. Then, the design details of MARL are described, followed by a novel design tailored specifically for federated fine-tuning. Finally, we prove its convergence.

A. Framework

Instead of making all synchronization decisions in the cloud, *Tomtit* lets each edge server independently observe federated learning status and adjust its synchronization strategy as well as the ones of devices associated with it. *Tomtit* can adapt itself to the open and dynamic federated fine-tuning environment, where devices can decide their adapter width and depth and they are free to join or leave, by using a distributed synchronization control based on MARL. Specifically, we deploy a control module called an agent in each edge server. As shown in Fig. 5, these agents collect information, including model parameters at corresponding edge servers and devices, training time and their energy consumption, as observations. Then, each agent determines a synchronization frequency as the action for each associated device. The cloud returns rewards to agents. The provided observation, action and reward constitute a trajectory for the current cloud communication round. We set a training time threshold, and upon exceeding this threshold, we terminate the current HFL training and utilize the collected trajectories to update the agents. *Tomtit* aims to achieve the following goals.

High accuracy. We aim to train adapters within a limited time to swiftly adapt to dynamic environments and enhance model accuracy.

Low energy consumption. Fine-tuning giant models is a process consuming time and energy. We expect *Tomtit* to minimize the average energy consumption of devices as much as possible.

Autonomy. We require meticulous design of states and observations to enable each agent to make independent and efficient decisions, which is advantageous for deploying large-scale systems.

In order to achieve the above design goals, we customize existing MARL by redefining states, actions and rewards, which play significant roles in guaranteeing the convergence of MARL and its efficiency.

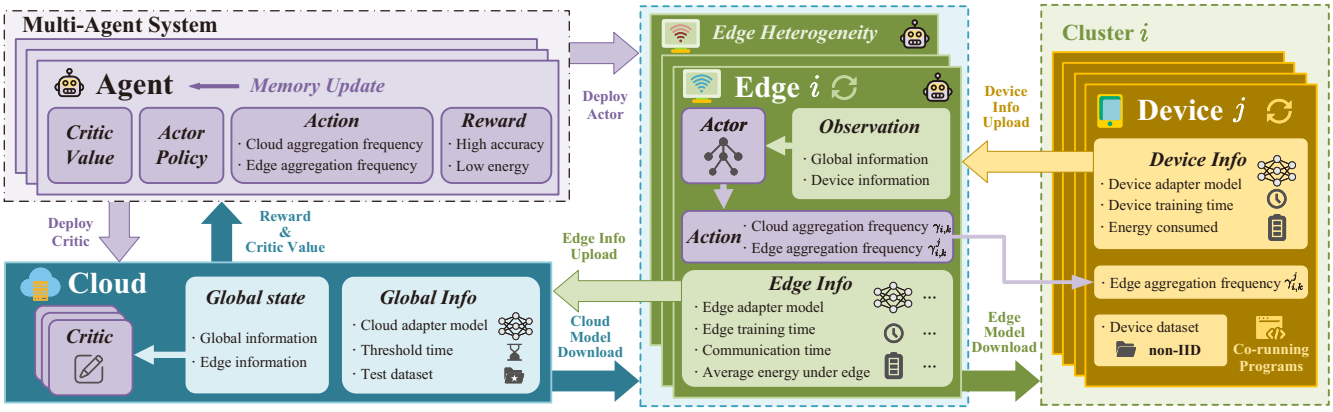


Fig. 5. *Tomtit* Design

1) State: The agents deployed on the edge servers $\mathbb{M} = \{1, 2, \cdots, M\}$ gather observations $\boldsymbol{o}_i(k)$ and the global state $\boldsymbol{s}(k)$ during each round of cloud communication k. The observations are used as inputs to the actor network to generate actions. Our observation design consists of three components. First, the agent i collects the adapter models of the current round from the cloud and the connected devices $\mathbb{N}_i = \{1, 2, \cdots, N_i\}$ as $(w(k), w_1(k), \ldots, w_{N_i}(k))$. We flatten the model parameters and arrange them, and compress the models into a lower-dimensional representation using principle component analysis (PCA). Experimental evidence from [19] has confirmed that model parameters can reflect data distribution. The first part of the observation is

$$\boldsymbol{o}_i^1(k) = \mathrm{PCA}\{[g(w(k))^T \; g(w_1(k))^T \ldots g(w_{N_i}(k))^T]^T\}, \tag{4}$$

where $g(\cdot)$ indicates the operation of flattening the model. Each observation of an agent also includes the training information $\boldsymbol{\varrho}_i^j(k) = [Ts_i^j(k-1) \; E_i^j(k-1)]$ for each device under the edge, where $Ts_i^j(k-1)$ and $E_i^j(k-1)$ represent the training time and energy consumption of device j under edge i in last round. Another component of the observation can be described as

$$\boldsymbol{o}_i^2(k) = [\boldsymbol{\varrho}_i^1(k) \; \boldsymbol{\varrho}_i^2(k) \ldots \boldsymbol{\varrho}_i^{N_i}(k)]^T. \tag{5}$$

The global information of the observation comprises the current remaining time $T^{re}(k)$, and the previous round's test accuracy $A^{test}(k-1)$, which can be written as $\boldsymbol{o}_i^3(k) = [T^{re}(k) \; A^{test}(k-1)]$. And the final observations are concatenated into a two-dimensional matrix as

$$\boldsymbol{o}_i(k) = \mathrm{cat}\{(\boldsymbol{o}_i^1(k), \mathrm{cat}\{(\boldsymbol{o}_i^3(k), \boldsymbol{o}_i^2(k)), \mathrm{dim} = 0\}), \mathrm{dim} = 1\}. \tag{6}$$

The process of observation concatenating is illustrated in detail in Fig. 6. We utilize the actor incorporating CNN to derive the action. The global state $\boldsymbol{s}(k)$ is for value generation to evaluate the current policy. In the following section, we provide a detailed explanation of its innovative construction.

2) Action: In each round of cloud communication, each agent i determines its action $\boldsymbol{a}_i(k) = \{\gamma_{i,k}, \gamma_{i,k}^1, \gamma_{i,k}^2, \ldots, \gamma_{i,k}^{N_i}\}$ based on local observations, where the elements in $\boldsymbol{a}_i(k)$ are all natural numbers. The $\gamma_{i,k}$ represents the cloud aggregation frequency of edge i, and $\gamma_{i,k}^j$ denotes the edge aggregation frequency of device j connected to edge i in cloud communication round k. We design the actor network to output a dimensionality of $2(N_i + 1)$, where every two parameters represent the mean and variance of a Gaussian distribution. After Gaussian sampling, we obtain $\boldsymbol{b}_i$, where the dimension is $(N_i + 1)$, comprising continuous non-zero values. We search for a suitable action $\boldsymbol{a}_i$ within a predefined action space, minimizing the norm distance between $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$. Finally we can the actual action $\boldsymbol{a}_i$, i.e., $\gamma_{i,k}, \gamma_{i,k}^j, j \in \mathbb{N}_i$.

3) Reward: After the k-th cloud communication, the cloud will provide a reward to each agent based on the training outcome. We aim to enhance the predictive performance of the model while reducing energy consumption. We denote the reward given to agent i after the k-th cloud communication as

$$r_i(k) = Q(A^{test}(k)) - Q(A^{test}(k-1)) - \epsilon E_i(k). \tag{7}$$

The first part of the rewards $Q(A^{test}(k)) - Q(A^{test}(k-1))$ represents the improvement in accuracy. $Q(\cdot)$ is a monotonically increasing function, which ensures that as the testing accuracy increases, the reward also increases correspondingly. The second part $E_i(k)$ represents the average energy consumption associated with the group i, where ϵ denotes the balancing weight.

4) Workflow: Algorithm 1 presents the training process of *Tomtit*, which can be summarized in the following steps:

1) Initialize the cloud model $w(0)$. Set threshold time T, remaining time $T^{re}(0)$ and $Agent_i$ for each edge i; (Line 1)
2) Train HFL by the given aggregation frequency in multiple rounds of cloud communication and train PCA modules by the adapter models parameters. Record $T^{use}(1)$, where $T^{use}(k) = \max\{Ts_i(k) + Tc_i(k)\}, i \in \mathbb{M}$ and the T_{init}^{re}; (Line 2-4)
3) For each $Agent_i$, get the $\boldsymbol{s}(k)$ and $\boldsymbol{o}_i(k)$, then choose the action $\boldsymbol{a}_i(k)$. Train HFL by $\{\boldsymbol{a}_i(k)\}$ and up-

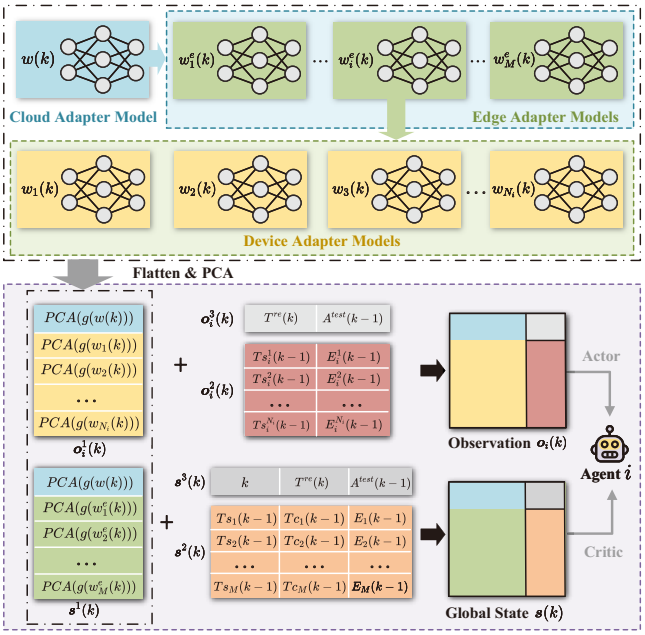


Fig. 6. Composition of the observation and global state

date the remaining time $T^{re}(k)$. We can get $r_i(k)$ and new state $\boldsymbol{s}(k+1)$ and $\boldsymbol{o}_i(k+1)$. Then push $(\boldsymbol{s}(k), \boldsymbol{o}_i(k), \boldsymbol{a}_i(k), r_i(k), \boldsymbol{s}(k+1), \boldsymbol{o}_i(k+1))$ to the agent memory; (Line 6-12)

4) Repeat step 3 until $T^{re}(k) < 0$, then end this episode and reset k and $T^{re}(k)$; (Line 13-16)
5) Update each $Agent_i$ by state-action-reward in the agent's memory pool and then clear the memory. In this paper, we use PPO to update the agent. (Line 18)

Steps 3-5 will be performed Ω times until MARL converges. It is worth noting that the PCA process is only trained once at the beginning and subsequently only performs inference at the edge, significantly reducing computational costs.

B. Enhancement

We devise custom crafts for MARL tailored specifically for HFL in order to enhance the convergence performance during training in our system.

Specific Global State. In a multi-agent system, the state of each agent i comprises a local observation and a global state. The observation is used by the actor network to generate policies, while the global state is often constructed by concatenating the observations and used by the critic network to generate values. However, concatenation in our system leads to an explosion in the dimensionality of the global state and violates privacy. As shown in Fig. 6, we devise a specific construction for the global state inspired by the EP design in SMAC [23]. Like the observation, the global state consists of three components. The first part includes the processed cloud and edge adapter models as

$$\boldsymbol{s}^1(k) = \mathrm{PCA}\{[g(w(k))^T \ g(w_1^e(k))^T \dots \ g(w_M^e(k))^T]^T\}. \tag{8}$$

Algorithm 1 *Tomtit*'s Training Process

1: Initialize round of cloud aggregations $k = 0$, threshold time T, remaining time $T^{re}(0) = T$, global model $w(0)$, $Agent_i$ for edge i, $i \in \mathbb{M}$;
2: Train several cloud aggregation by given aggregation frequencies, get $w(1)$, $w_i^e(1)$, $w_j(1)$, and record $T^{use}(1)$;
3: Train PCA module by $w(1)$,$w_i^e(1)$, and $w_j(1)$;
4: Update $T^{re}_{init} = T^{re}(1) - T^{use}(1)$; k++;
5: **for** 1 **to** Ω **do**
6: Get $\boldsymbol{s}(k)$ and $\boldsymbol{o}_i(k)$ for $Agent_i$, $i \in \mathbb{M}$;
7: **while true do**
8: $\boldsymbol{a}_i(k) = Agent_i.choose\_action(\boldsymbol{o}_i(k)), i \in \mathbb{M}$;
9: Train HFL by $\{\boldsymbol{a}_i(k)\}_{i\in\mathbb{M}}$, record $T^{use}(k)$ and update $T^{re}(k) = T^{re}(k-1) - T^{use}(k)$;
10: $\boldsymbol{s}(k+1), \boldsymbol{o}_i(k+1), r_i(k) = Agent_i.step(), i \in \mathbb{M}$;
11: $\tau_i^k = (\boldsymbol{s}(k), \boldsymbol{o}_i(k), \boldsymbol{a}_i(k), r_i(k), \boldsymbol{s}(k+1), \boldsymbol{o}_i(k+1))$;
12: $Agent_i.push(\tau_i^k), i \in \mathbb{M}$; k++;
13: **if** $T^{re}(k) < 0$ **then**
14: Set $k = 1$, $T^{re}(k) = T^{re}_{init}$;
15: **break**
16: **end if**
17: **end while**
18: $Agent_i.PPO\_update\_\&\_clear(), i \in \mathbb{M}$;
19: **end for**

The second part consists of $\boldsymbol{\varrho}_i(k) = [Ts_i(k-1) \ Tc_i(k-1) \ E_i(k-1)]$, where $Ts_i(k-1)$ and $E_i(k-1)$ represent the maximum training time and average energy consumption of cluster i, respectively, $Tc_i(k-1)$ denotes the communication time of edge i, is

$$\boldsymbol{s}^2(k) = [\boldsymbol{\varrho}_1(k) \ \boldsymbol{\varrho}_2(k) \dots \boldsymbol{\varrho}_M(k)]^T. \tag{9}$$

The third part encompasses a current iteration number k of cloud communication rounds, the remaining time $T^{re}(k)$, and the previous test accuracy $A^{test}(k-1)$, i.e., $\boldsymbol{s}^3(k) = [k \ T^{re}(k) \ A^{test}(k-1)]$. The final global state is

$$\boldsymbol{s}(k) = \mathrm{cat}\{(\boldsymbol{s}^1(k), \mathrm{cat}\{(\boldsymbol{s}^3(k), \boldsymbol{s}^2(k)), \mathrm{dim} = 0\}), \mathrm{dim} = 1\}. \tag{10}$$

Similarly, we put the global state into a CNN structure similar to the actor network to construct the critic network, from which we obtain the value.

Deployment of Disaggregated Agents. Traditional agent integrates actor and critic networks. The global state $\boldsymbol{s}(k)$ encompasses the adapter model information of all edges as the critic's input. However, deploying the critic at the edge and gathering models from other edges would result in significant additional overhead. To address this challenge, as shown in Fig. 5, we deploy the critic in the cloud while the actor is deployed at the edge. During the HFL process in round k, the cloud collects edge models $w_i^e(k), i \in \mathbb{M}$, eliminating the need for additional communication costs in constructing the global state $\boldsymbol{s}(k)$. Additionally, since the critic generates only value outputs, the communication overhead for transmitting them to the edge can be negligible.

Utility Reward Function. Based on the convergence property of the machine learning task, the model's accuracy growth rate slows down during the later stages of training. If $Q(\cdot)$ is simply implemented as a linear increasing function, the agent may not receive substantial rewards in the later stages of training, partly due to the slow improvement in accuracy. It may lead to a sparse reward problem when training MARL. Therefore, we set $Q(u) = \Upsilon^u$. Because when $\ddot{Q} > 0$ and the accuracy increases slowly in the later stages, we can still obtain substantial rewards. Here we set $\Upsilon = 64$. Besides, our reward design takes into account both overall accuracy and local energy consumption on edge, which facilitates collaborative cooperation among multiple agents.

In the following section, we design ablation experiments to validate the superiority of the special designs.

C. Convergence

In this part, we provide the convergence bound for the synchronization scheme determined by *Tomtit*. We first make two assumptions about the loss function as follows:

Assumption 1: The loss function is L-smooth and the Lipschitz constant $L > 0$, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$.

Assumption 2: The estimated stochastic gradient is unbiased for devices, i.e., $\mathbb{E}\left[\left\|\tilde{\nabla} f_j(w_j) - \nabla f(w)\right\|^2 \mid w\right] \leq \sigma^2$.

Theorem 1: After subjecting our refined frequency variation method to a round of cloud communication, the convergence bound is as follows

$$\begin{aligned}&\mathbb{E}[f\left(w(k+1)\right)] - \mathbb{E}[f\left(w(k)\right)] \\ \leq &\frac{L^2\eta^3}{4}\widetilde{\gamma}_1\widetilde{\gamma}_2\left((\widetilde{\gamma}_1 - 1) + \frac{M}{N}\widetilde{\gamma}_1\left(\widetilde{\gamma}_2 - 1\right)\right)\sigma^2 \\ &+ \frac{L\eta^2}{2}\frac{1}{N}\widetilde{\gamma}_1\widetilde{\gamma}_2\sigma^2 - \frac{\eta}{2}\widetilde{\gamma}_1\widetilde{\gamma}_2\mathbb{E}\left\|\nabla f\left(w(k)\right)\right\|^2,\end{aligned} \tag{11}$$

where $\widetilde{\gamma_1}$ and $\widetilde{\gamma_2}$ are respectively the maximum values of the edge aggregation frequency and the cloud aggregation frequency in round k.

Proof: For the SGD optimization algorithm, the model parameters evolve as follows

$$w(t) = w(t-1) - \eta\tilde{\nabla} f\left(w(t-1)\right). \tag{12}$$

After each round of cloud aggregation, we can get the model updates by combining the aggregation formula as

$$\begin{aligned}&w(k+1) = w(k) - \\ &\eta\sum_{i\in\mathbb{M}}\frac{N_i}{N}\frac{1}{N_i}\sum_{\alpha=0}^{\gamma_{i,k}-1}\sum_{j\in\mathbb{N}_i}\sum_{\beta=0}^{\gamma_{i,k}^j-1}\tilde{\nabla} f_j\left(w_j\left(k,\alpha,\beta\right)\right).\end{aligned} \tag{13}$$

where $w_j\left(k,\alpha,\beta\right)$ represents the model of the j-th device during the k-th cloud communication, α rounds of edge aggregation, and β rounds of local training.

Based on the assumptions, we can deduce the relationship between the models $w(k)$ and $w(k+1)$ as follows

$$\begin{aligned}\mathbb{E}[f\left(w(k+1)\right)] - \mathbb{E}[f\left(w(k)\right)] \leq &\frac{L}{2}\mathbb{E}\left\|w(k+1) - w(k)\right\|^2 \\ &+ \mathbb{E}\left\langle\nabla f\left(w(k)\right), w(k+1) - w(k)\right\rangle.\end{aligned} \tag{14}$$

After taking the expectation, the second part of (14) can be written as

$$\begin{aligned}&\mathbb{E}\left\langle\nabla f\left(w(k)\right), w(k+1) - w(k)\right\rangle = -\eta\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\frac{1}{N}\cdot \\ &\sum_{t_2=0}^{\gamma_{i,k}-1}\sum_{t_1=0}^{\gamma_{i,k}^j-1}\mathbb{E}\left\langle\nabla f\left(w(k)\right), \nabla f\left(w_j(k,t_2,t_1)\right)\right\rangle.\end{aligned} \tag{15}$$

And the expectation is defined as

$$\begin{aligned}&-\mathbb{E}\left\langle\nabla f\left(w(k)\right), \nabla f\left(w_j(k,t_2,t_1)\right)\right\rangle \\ =&-\frac{1}{2}\mathbb{E}\left\|\nabla f\left(w(k)\right)\right\|^2 - \frac{1}{2}\mathbb{E}\left\|\nabla f\left(w_j(k,t_2,t_1)\right)\right\|^2 \\ &+\frac{1}{2}\mathbb{E}\left\|\nabla f\left(w(k)\right) - \nabla f\left(w_j(k,t_2,t_1)\right)\right\|^2\end{aligned} \tag{16}$$

Along with $2\langle\boldsymbol{a},\boldsymbol{b}\rangle = \|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 - \|\boldsymbol{a}-\boldsymbol{b}\|^2$ and $\mathbb{E}\|x\|^2 = \|\mathbb{E}x\|^2 + \mathrm{Var}(x)^2$, we can get the bound of the second part of (14) as

$$\begin{aligned}&\mathbb{E}\left\langle\nabla f\left(w(k)\right), w(k+1) - w(k)\right\rangle \\ \leq &-\frac{\eta}{2}\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\frac{1}{N}\sum_{t_2=0}^{\gamma_{i,k}-1}\sum_{t_1=0}^{\gamma_{i,k}^j-1}\mathbb{E}\left\|\nabla f\left(w_j(k,t_2,t_1)\right)\right\|^2 \\ &+\frac{L^2\eta^3}{2}\frac{1}{N}\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\left(\frac{\gamma_{i,k}^j(\gamma_{i,k}^j-1)}{2} + \frac{(\gamma_{i,k}-1)\gamma_{i,k}}{2}\widetilde{\gamma_1}^2\right)\cdot \\ &\sum_{\alpha=0}^{\gamma_{i,k}-1}\sum_{\beta=0}^{\gamma_{i,k}^j-1}\mathbb{E}\left\|\nabla f\left(w_j(k,\alpha,\beta)\right)\right\|^2 - \frac{\eta}{2}\widetilde{\boldsymbol{\gamma_2}}\widetilde{\boldsymbol{\gamma_1}}\mathbb{E}\left\|\nabla f\left(w(k)\right)\right\|^2 \\ &+\frac{L^2\eta^3}{2}\frac{\widetilde{\boldsymbol{\gamma_2}}\widetilde{\boldsymbol{\gamma_1}}}{2}\left((\widetilde{\boldsymbol{\gamma_1}} - 1) + \frac{M}{N}\widetilde{\boldsymbol{\gamma_1}}\left(\widetilde{\boldsymbol{\gamma_2}} - 1\right)\right)\sigma^2\end{aligned} \tag{17}$$

The bound of the first part in (14) is similar to the second part, we can get

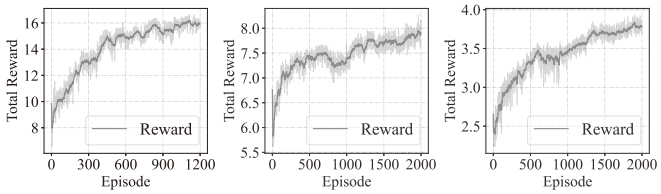
$$\begin{aligned}&\mathbb{E}\left\|w(k+1) - w(k)\right\|^2 \leq \eta^2\frac{1}{N^2}\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\sum_{t_2=0}^{\gamma_{i,k}-1}\widetilde{\boldsymbol{\gamma_1}}\sigma^2 + \\ &\eta^2\frac{1}{N}\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\gamma_{i,k}\gamma_{i,k}^j\sum_{t_2=0}^{\gamma_{i,k}-1}\sum_{t_1=0}^{\gamma_{i,k}^j-1}\mathbb{E}\left\|\nabla f\left(w_j(k,t_2,t_1)\right)\right\|^2\end{aligned} \tag{18}$$

By combining (14), (17) and (18), we can obtain the convergence bound as

$$\begin{aligned}&\mathbb{E}[f\left(w(k+1)\right)] - \mathbb{E}[f\left(w(k)\right)] \leq -\frac{\eta}{2}\widetilde{\boldsymbol{\gamma_1}}\widetilde{\boldsymbol{\gamma_2}}\mathbb{E}\left\|\nabla f\left(w(k)\right)\right\|^2 \\ &-\frac{\eta}{2}\frac{1}{N}\sum_{i\in\mathbb{M}}\sum_{j\in\mathbb{N}_i}\sum_{t_2=0}^{\gamma_{i,k}-1}\sum_{t_1=0}^{\gamma_{i,k}^j-1}P\cdot\mathbb{E}\left\|\nabla f\left(w_j(k,t_2,t_1)\right)\right\|^2 + \\ &\frac{L^2\eta^3}{4}\widetilde{\gamma}_1\widetilde{\gamma}_2\left((\widetilde{\gamma}_1 - 1) + \frac{M}{N}\widetilde{\gamma}_1\left(\widetilde{\gamma}_2 - 1\right)\right)\sigma^2 + \frac{L\eta^2}{2}\frac{1}{N}\widetilde{\gamma}_1\widetilde{\gamma}_2\sigma^2,\end{aligned} \tag{19}$$

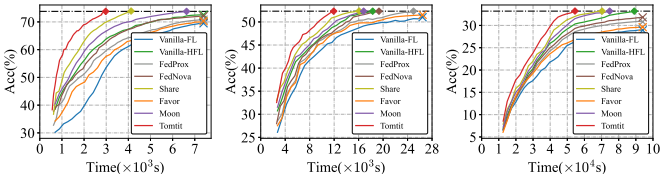
where

$$\begin{aligned}P =&1 - \frac{L^2\eta^2\gamma_{i,k}^j(\gamma_{i,k}^j - 1)}{2} \\ &- \frac{L^2\eta^2\widetilde{\boldsymbol{\gamma_2}}^2\gamma_{i,k}^j(\gamma_{i,k} - 1)}{2} - L\eta\gamma_{i,k}\gamma_{i,k}^j.\end{aligned} \tag{20}$$



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 7. The reward of training the MARL agent



(a) MNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 8. Accuracy *v.s.* time of different FL methods

L represents a small value, and Theorem 1 can be proven from (19) when $P \geq 0$.

IV. Evaluation

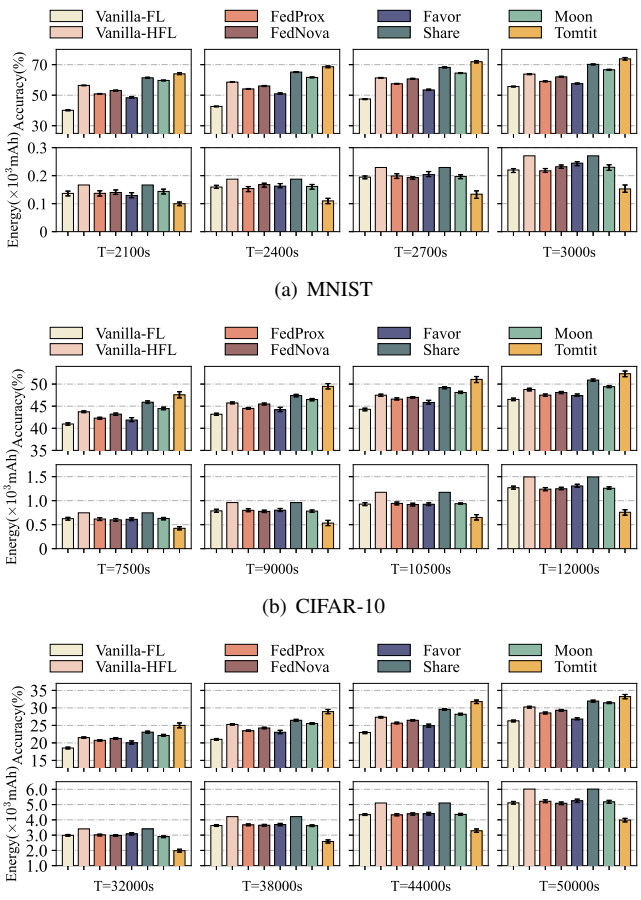
In this section, we evaluate *Tomtit* on a testbed using real data and conduct various experiments to compare it with state-of-the-arts, highlighting the superiority of *Tomtit*.

A. Settings

Testbed. We construct an HFL testbed using Raspberry Pis, laptops, and Alibaba Cloud. The edge servers and devices are configured in different scales: the numbers of edge servers are 5 (default) or 10, and devices are 20, 50 (default), or 100, to demonstrate various sizes of HFL systems. To address system heterogeneity, we utilize *stress-ng* [18] to configure the CPU usage of the Raspberry Pis across 5 different categories, ranging from 10% to 80%. As for edge communication heterogeneity, we sample bandwidth from MobiPerf and apply the obtained results to the edge [24].

Experimental Setup. We use the commonly visual model which is pre-trained. Subsequently, we follow the method in [8] to add adapters into the vision model and perform fine-tuning on three datasets: MNIST, CIFAR-10, and CIFAR-100. We have two settings of non-IID data. One is the label non-IID, where each device trains data with specific labels, and the other is based on the Dirichlet distribution [25]. For each agent, we design the actor and critic networks with 2 convolutional layers and 3 fully connected layers to handle observations and global states.

Baselines. We compare *Tomtit* with *Vanilla-FL*, *Vanilla-HFL*, as well as several recent FL solutions: *FedProx* [26], *FedNova* [27], *Share* [14], and *Moon* [28]. *FedProx* addresses the non-IID problem by modifying the loss function. *FedNova* takes into account the scenario of system heterogeneity. *Favor* presents an RL algorithm for optimizing FL. *Share* comprehensively considers the heterogeneity of data and communication based on the HFL framework. *Moon* is an FL algorithm incorporating contrastive learning.



(a) MNIST

(b) CIFAR-10

(c) CIFAR-100

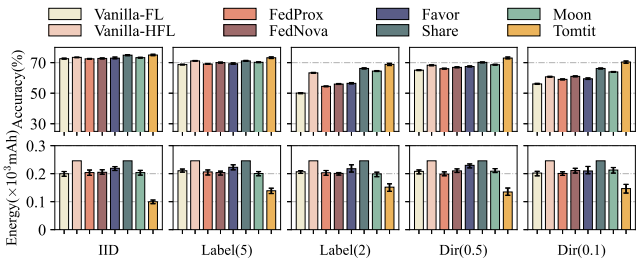
Fig. 9. Accuracy and energy consumption under different training time

Hyperparameters. For baselines, we refer to their original paper's configuration. The hyperparameter settings for *Tomtit* are as follows: The SGD batch size is 32, and the learning rate is 0.0002 for MNIST, 0.0035 for CIFAR-10 and 0.003 for CIFAR-100. For the clip function, we set ε = 0.2. We set reward weight $\epsilon = 0.04$ for MNIST, $\epsilon = 0.001$ for CIFAR-10 and $\epsilon = 0.0002$ for CIFAR-100.

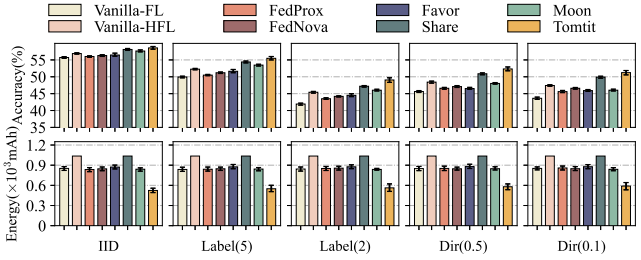
B. Training Performance

DRL Training. We conduct HFL training on three datasets for 3000s, 12000s and 55000s, respectively. Fig. 7 depicts the training rewards of an agent, illustrating the gradual convergence of the agent's reward. The accuracy of *Tomtit* on MNIST, CIFAR-10 and CIFAR-100 can converge to 73.8%, 52.9%, and 33.2%, respectively. The time-to-accuracy curves are shown in Fig. 8, where we can observe that other systems need significantly longer time to achieve the similar accuracy of *Tomtit*. We can save 59.6%, 56.8%, and 48.3% training time in the three datasets, respectively.

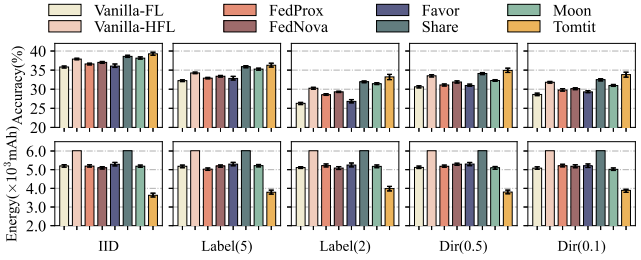
Different Threshold time. We conduct experiments to study the influence of training time on accuracy and average



(a) MNIST



(b) CIFAR-10



(c) CIFAR-100

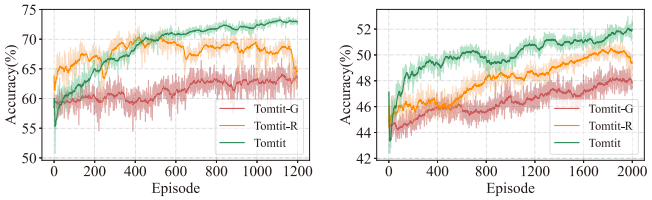
Fig. 10. Accuracy and energy consumption under different non-IID levels

energy consumption. As shown in Fig. 9, *Tomtit* exhibits remarkable performance across different threshold time. The accuracy of *Tomtit* outperforms other algorithms by an average increase of 19.7%, 9.5% and 13.3% in the three datasets, while the average energy consumption is reduced by 43.6%, 42.5% and 33.7%.

Different non-IID levels. We consider 5 kinds of data distributions in our study: IID, random allocation of 2 or 5 labels per device (Label(2)&(5)), and Dirichlet distribution with two distinct sets of hyperparameters (Dir(0.5)&(0.1)). As shown in Fig. 10, *Tomtit* demonstrates favorable results across various non-IID levels, especially with significant improvements under strong non-IID conditions. Specifically, *Tomtit* shows an average improvement of 5.9%, 8.4% and 8.7% in accuracy and reduction of 40.5%, 39.2% and 30.7% in energy consumption for MNIST, CIFAR-10 and CIFAR-100, respectively.

C. Adaptability

Scalability. We evaluate the performance under various scale configurations, as presented in Table I. The threshold time T of datasets is set to 3000s, 12000s, and 55000s, respectively. As Table I indicates, *Tomtit* achieves the lowest



(a) Accuracy of training MNIST (b) Accuracy of training CIFAR-10

Fig. 11. Training the DRL agent

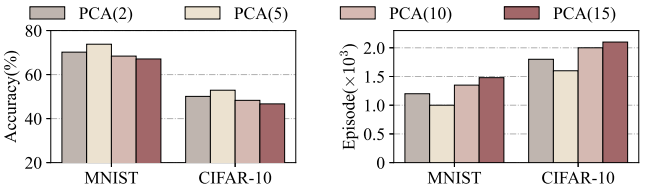


Fig. 12. Impact of different principal component

energy consumption across all cases. However, the action dimension per agent is slightly higher than other systems when the numbers of edges and devices are 5 and 100. In most of cases, *Tomtit* can achieve the highest accuracy. And the improvement becomes more pronounced as the ratio between edges and devices increases. When we have 5 edges and 100 devices, *Tomtit*'s accuracy is lower than *Share*, but the gap is only 0.6%.

Dynamic System Environment. In practice, CPU utilization during training may dynamically change. In addition, the non-IID distribution of data held by devices may also change, such as in the case of cameras collecting real-time information in a factory. We then study the influence of such dynamic to demonstrate the robustness of *Tomtit*. First, we train MARL with default settings, achieving accuracies of 73.82%, 52.93% and 33.23% on the MNIST, CIFAR-10 and CIFAR-100 datasets, respectively. After that, we randomly select 10%, 30%, and 50% of the devices and perform random exchanges of their CPU usage, and data to simulate environmental dynamic. Experimental results are summarized in Table II. We can observe that dynamic changes have minimal impact on *Tomtit*'s performance. Compared to other baselines, *Tomtit* still achieves remarkable results, which demonstrate excellent performance in dynamic environments.

D. Ablation experiment

The Impact of Agent Design. We conduct ablation experiments to demonstrate the effectiveness of the enhancement design of MARL. We adopt the strategy of directly concatenating observations as the global state for training, and refer to this approach as *Tomtit-G*. We also analyze the effectiveness of the reward. We replace $Q(u) = \Upsilon^u$ with $Q(u) = u$ and call it as *Tomtit-R*. As shown in Fig 11, thanks to the sophisticated design of state and reward, *Tomtit* can achieve the highest accuracy with excellent convergence stability.

State Dimension. In the design of MARL, the dimensionality of the state changes due to different hyperparameter settings of PCA. Fig. 12 shows the final convergence accuracy

TABLE I
PERFORMANCE AT DIFFERENT SCALES

Scale [Edges, Devices]	Data	*Vanilla-FL* Acc (%)	*Vanilla-FL* Energy (mAh)	*Vanilla-HFL* Acc (%)	*Vanilla-HFL* Energy (mAh)	*FedProx* Acc (%)	*FedProx* Energy (mAh)	*FedNova* Acc (%)	*FedNova* Energy (mAh)	*Favor* Acc (%)	*Favor* Energy (mAh)	*Share* Acc (%)	*Share* Energy (mAh)	*Moon* Acc (%)	*Moon* Energy (mAh)	*Tomtit* Acc (%)	*Tomtit* Energy (mAh)
[5, 20]	MNIST	51.8	218.3	58.9	268.7	55.3	225.3	57.4	226.8	53.1	209.5	62.2	268.7	60.9	228.1	**64.5**	**148.3**
	CIFAR-10	43.3	1258.8	46.0	1453.2	45.7	1240.3	45.8	1251.6	44.9	1289.2	47.8	1453.2	46.1	1247.4	**48.4**	**781.5**
	CIFAR-100	24.5	5008.3	28.4	6093.6	27.5	5109.7	27.9	5153.6	25.1	5220.7	29.2	6093.6	29.0	5111.6	**31.5**	**3844.5**
[5, 50]	MNIST	55.7	220.3	63.8	271.0	59.6	217.5	62.1	232.1	57.6	244.7	70.2	271.0	66.7	229.6	**73.8**	**152.7**
	CIFAR-10	46.5	1270.4	48.8	1494.8	47.5	1240.0	48.1	1249.3	47.4	1306.8	50.9	1494.8	49.4	1258.8	**52.9**	**758.6**
	CIFAR-100	26.3	5112.6	30.3	6014.8	28.6	5217.5	29.3	5087.4	26.8	5248.8	32.0	6014.8	31.4	5176.8	**33.2**	**3985.6**
[5, 100]	MNIST	48.1	231.8	54.2	270.8	52.1	218.9	52.9	206.9	50.8	228.7	56.3	270.8	55.7	222.5	**56.3**	**179.5**
	CIFAR-10	40.2	1249.2	44.1	1437.6	42.5	1286.3	43.4	1256.7	42.4	1251.2	**45.8**	1437.6	43.5	1248.5	45.2	**795.4**
	CIFAR-100	22.1	5023.1	27.0	5983.7	25.2	5154.2	26.1	5169.3	22.9	5201.5	28.1	5983.7	27.8	5281.4	**30.2**	**4021.0**
[10, 100]	MNIST	48.1	231.8	54.9	270.8	52.1	218.9	52.9	206.9	50.8	228.7	57.0	270.8	55.7	222.5	**58.6**	**161.2**
	CIFAR-10	40.2	1249.2	44.3	1437.6	42.5	1286.3	43.4	1256.7	42.4	1251.2	45.6	1437.6	43.5	1248.5	**46.9**	**781.8**
	CIFAR-100	22.1	5023.1	27.6	5983.7	25.2	5154.2	26.1	5169.3	22.9	5201.5	28.5	5983.7	27.8	5281.4	**30.6**	**3912.7**

TABLE II
ROBUSTNESS IN DYNAMIC ENVIRONMENTS

		10%	30%	50%
CPU	MNIST	73.71%(-0.21%)	73.45%(-0.37%)	73.12%(-0.60%)
	CIFAR-10	52.64%(-0.29%)	52.35%(-0.58%)	52.11%(-0.82%)
	CIFAR-100	32.92%(-0.31%)	32.76%(-0.47%)	32.44%(-0.79%)
Data	MNIST	73.15%(-0.67%)	72.91%(-0.91%)	72.50%(-1.21%)
	CIFAR-10	52.27%(-0.66%)	51.68%(-1.25%)	51.49%(-1.44%)
	CIFAR-100	32.85%(-0.38%)	32.11%(-1.12%)	31.70%(-1.53%)
CPU+Data	MNIST	72.54%(-1.28%)	72.05%(-1.77%)	71.86%(-1.94%)
	CIFAR-10	52.03%(-0.90%)	51.30%(-1.63%)	50.96%(-1.97%)
	CIFAR-100	32.49%(-0.74%)	31.88%(-1.35%)	31.24%(-1.99%)

and the required training episodes for MARL convergence under different PCA dimensionality reduction scenarios. We can observe that excessively large and small dimensions of the state result in suboptimal performance. Selecting appropriate PCA parameters is crucial.

V. RELATED WORKS

Federeted Learning for Giant Models. With the current advancements in LLMs, numerous FL efforts have been directed toward addressing issues concerning giant models. Lin *et al.* [29] introduced FedNLP, which currently serves as a benchmark for handling NLP data in FL. Based on this, Cai *et al.* [8] proposed FedAdapter, which incorporates adapter fine-tuning within the realm of FL. Guo *et al.* [30] introduced PromptFL, which leverages prompt learning to significantly reduce communication overhead. Furthermore, Sun *et al.* [9] presented FedPEFT, a method that requires only localized adjustments and aggregates a small fraction of model weights on a global scale.

Deep Reinforcement Learning in FL. Many works employ intelligent algorithms such as reinforcement learning to address the dynamic and heterogeneous effects in FL. Kim *et al.* [20] introduced AutoFL, which dynamically allocates CPU voltage and frequency to devices. Deng *et al.* [21] presented the Auction, incorporating transformer models into reinforcement learning to enhance system scalability. Zhang *et al.* [31] proposed FedMarl based on multi-agent reinforcement learning, utilizing VDN for client selection. Yu *et al.* [32] proposed the SPATL, which employs reinforcement learning to select the primary model parameters for aggregation in each communication round. Nguyen *et al.* [33] introduced a novel non-IID type and employed a DRL-based algorithm to deal with it.

Representive work in HFL. Many efforts are made to optimize the HFL architecture. Wang *et al.* [10] proposed RFL-HA that combines asynchronous and synchronous approaches. Mhaisen *et al.* [13] optimized the combination problem between devices and edges to minimize the class distribution distance. Li *et al.* [12] introduced FedGS for the IIoT environment, employing the gradient-based binary permutation algorithm (GBP-CS) for subset selection. Cui *et al.* [34] proposed a heterogeneity-aware heuristic user selection strategy to select devices and determine operating frequency.

VI. CONCLUSION

In this paper, we have introduced *Tomtit*, a hierarchical federated fine-tuning system that can significantly accelerate the learning process and improve the energy efficiency. To address scalability issues, we have introduced the HFL framework. Additionally, we have proposed a novel synchronization scheme, which controls the edge and device aggregation frequency, to tackle the heterogeneity problem caused by adapters and environments. We have designed MARL to make decisions on the synchronization scheme and provided corresponding convergence bounds. Furthermore, we have presented unique designs for the agent in our system. Finally, we have conducted experiments on the fine-tuning tasks, comparing our approach to the state-of-the-arts. The results have demonstrated that *Tomtit* performs remarkably well across multiple datasets and exhibits significant improvements in dynamic and heterogeneous conditions.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China No. 62102022, Japan Society for the Promotion of Science (JSPS) KAKENHI No. 21H03424, and Japan Science and Technology Agency (JST) PRESTO No. 23828673. Yufeng Zhan and Peng Li are the corresponding authors.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," in *Proc. of NeurIPS*, pp. 1877–1901, 2020.

[3] L. Yuan, D. Chen, Y.-L. Chen, N. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, *et al.*, "Florence: A new foundation model for computer vision," *arXiv preprint arXiv:2111.11432*, 2021.

[4] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. of ICML*, pp. 8748–8763, 2021.

[5] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. of ACL*, pp. 328–339, 2018.

[6] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *Proc. of ICML*, pp. 2790–2799, 2019.

[7] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, G. Cao, D. Jiang, M. Zhou, *et al.*, "K-adapter: Infusing knowledge into pre-trained models with adapters," *arXiv preprint arXiv:2002.01808*, 2020.

[8] D. Cai, Y. Wu, S. Wang, F. X. Lin, and M. Xu, "Fedadapter: Efficient federated learning for modern nlp," in *Proc. of ACM MobiCom*, pp. 1–14, 2023.

[9] G. Sun, M. Mendieta, T. Yang, and C. Chen, "Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning," *arXiv preprint arXiv:2210.01708*, 2022.

[10] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. of IEEE INFOCOM*, pp. 1–10, 2021.

[11] Y. Cui, K. Cao, J. Zhou, and T. Wei, "Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 5, pp. 1518–1531, 2022.

[12] Z. Li, Y. He, H. Yu, J. Kang, X. Li, Z. Xu, and D. Niyato, "Data heterogeneity-robust federated learning via group client selection in industrial iot," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17844–17857, 2022.

[13] N. Mhaisen, A. A. Abdellatif, A. Mohamed, A. Erbad, and M. Guizani, "Optimal user-edge assignment in hierarchical federated learning based on statistical properties and network topology constraints," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 55–66, 2021.

[14] Y. Deng, F. Lyu, J. Ren, Y. Zhang, Y. Zhou, Y. Zhang, and Y. Yang, "Share: Shaping data distribution at edge for communication-efficient hierarchical federated learning," in *Proc. of IEEE ICDCS*, pp. 24–34, 2021.

[15] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, "T-rex: A large scale alignment of natural language with knowledge base triples," in *Proc. of LREC*, 2018.

[16] F. C. Commission, "Measure broadband america." https://www.fcc.gov/general/measuring-broadband-america 2021.

[17] A. Benchmark, "All about deep learning on smartphones." https://ai-benchmark.com/ranking_deeplearning_detailed.html 2021.

[18] K. C. I, "Stress-ng." http://kernel.ubuntu.com/git/king/stressng.git/(visitedon28/03/2018) 2017.

[19] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *Proc. of IEEE INFOCOM*, pp. 1698–1707, 2020.

[20] Y. G. Kim and C.-J. Wu, "Autofl: Enabling heterogeneity-aware energy efficient federated learning," in *Proc. of IEEE/ACM MICRO*, pp. 183–198, 2021.

[21] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "Auction: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, 2021.

[22] J. Zhang, S. Guo, Z. Qu, D. Zeng, Y. Zhan, Q. Liu, and R. Akerkar, "Adaptive federated learning on non-iid data with resource constraint," *IEEE Transactions on Computers*, vol. 71, no. 7, pp. 1655–1667, 2021.

[23] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. of AAAI*, 2018.

[24] M-Lab, "Mobiperf data set." https://www.measurementlab.net/tests/mobiperf/.

[25] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[26] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. of MLSys*, pp. 429–450, 2020.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTATS*, pp. 1273–1282, 2017.

[28] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. of IEEE/CVF CVPR*, pp. 10713–10722, 2021.

[29] B. Y. Lin, C. He, Z. Zeng, H. Wang, Y. Huang, C. Dupuy, R. Gupta, M. Soltanolkotabi, X. Ren, and S. Avestimehr, "Fednlp: Benchmarking federated learning methods for natural language processing tasks," *arXiv preprint arXiv:2104.08815*, 2021.

[30] T. Guo, S. Guo, J. Wang, and W. Xu, "Promptfl: Let federated participants cooperatively learn prompts instead of models–federated learning in age of foundation model," *arXiv preprint arXiv:2208.11625*, 2022.

[31] S. Q. Zhang, J. Lin, and Q. Zhang, "A multi-agent reinforcement learning approach for efficient client selection in federated learning," in *Proc. of AAAI*, pp. 9091–9099, 2022.

[32] S. Yu, P. Nguyen, W. Abebe, W. Qian, A. Anwar, and A. Jannesari, "Spatl: salient parameter aggregation and transfer learning for heterogeneous federated learning," in *Proc. of IEEE SC*, pp. 495–508, 2022.

[33] N. H. Nguyen, P. L. Nguyen, T. D. Nguyen, T. T. Nguyen, D. L. Nguyen, T. H. Nguyen, H. H. Pham, and T. N. Truong, "Feddrl: Deep reinforcement learning-based adaptive aggregation for non-iid data in federated learning," in *Proc. of ACM ICPP*, pp. 1–11, 2022.

[34] Y. Cui, K. Cao, J. Zhou, and T. Wei, "Optimizing training efficiency and cost of hierarchical federated learning in heterogeneous mobile-edge cloud computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.