Enhancing Federated Learning with Server-Side Unlabeled Data by Adaptive Client and Data Selection

Yang Xu, *Member, IEEE,* Lun Wang, Hongli Xu, *Member, IEEE,* Jianchun Liu, Zhiyuan Wang, Liusheng Huang, *Member, IEEE*

Abstract—Federated learning (FL) has been widely applied to collaboratively train deep learning (DL) models on massive end devices (*i.e.*, clients). Due to the limited storage capacity and high labeling cost, the data on each client may be insufficient for model training. Conversely, in cloud datacenters, there exist large-scale unlabeled data, which are easy to collect from public access (*e.g.*, social media). Herein, we propose the *Ada-FedSemi* system, which leverages both on-device labeled data and in-cloud unlabeled data to boost the performance of DL models. In each round, local models are aggregated to produce pseudo-labels for the unlabeled data, which are utilized to enhance the global model. Considering that the number of participating clients and the quality of pseudo-labels will have a significant impact on the training performance, we introduce a multi-armed bandit (MAB) based online algorithm to adaptively determine the participating fraction and confidence threshold. Besides, to alleviate the impact of stragglers, we assign local models of different depths for heterogeneous clients. Extensive experiments on benchmark models and datasets show that given the same resource budget, the model trained by Ada-FedSemi achieves 3%∼14.8% higher test accuracy than that of the baseline methods. When achieving the same test accuracy, Ada-FedSemi saves up to 48% training cost, compared with the baselines. Under the scenario with heterogeneous clients, the proposed HeteroAda-FedSemi can further speed up the training process by 1.3×∼1.5×.

Index Terms—*Edge Computing, Federated Learning, Semi-supervised Learning, Pseudo-labeling.*

1 Introduction

With the considerable development of deep learning (DL) in recent years, more and more AI applications are penetrating our daily life, such as smart transportation [1], virtual reality [2] and intelligent assistants [3]. In order to utilize data generated at the network edge without possible leakage of personal privacy, federated learning (FL) [4] is proposed to collaboratively train DL models on massive end devices with the aid of the parameter server (PS). In FL, end devices (*i.e.*, clients) keep their data locally during training and only upload local models to the PS periodically for global aggregation. Then, the PS broadcasts the global model back to clients, and the interaction procedure will last until the model converges.

The most existing works of FL concentrate on training efficiency and assume sufficient labeled data on clients. However, due to the high labeling cost or lack of expert knowledge for annotation, the scale of labeled data on each client may be small [5]. Since the strong performance of DL models is largely attributed to the availability of abundant data (especially the labeled data), the insufficient labeled data on clients may result in model [6]. On the contrary, there are various other data sources (*e.g.*, social networks [7]) continuously generating unlabeled data [8], which are collected and stored in cloud datacenters. For example, the large-scale WebVision Database [9] consists of 2.4 million web images crawled from the Internet, and the images may lack ground-truth labels [10].

To fully utilize both labeled and unlabeled data in FL, a new technology of federated semi-supervised learning (FSSL) has been proposed [11]. Long *et al.* [12] and Jeong *et al.* [11] assume the labeled and unlabeled data are already on clients, while some works [13], [14] distribute unlabeled data from the cloud to clients and then implement FSSL using the mixed data on clients. However, limited by the storage capacity of clients, the data size on clients is much smaller than that in cloud, which restricts the performance boost of DL models. Besides, delivering additional data from the cloud to clients will incur a large amount of communication cost and also increase computation overhead for the resource-constrained clients. Instead, in works [15], [16], the PS first collects local models from all clients as teacher models, and then produces pseudo-labels for the in-cloud unlabeled data in terms of the teachers' predictions. Subsequently, the pseudo-labeled data are exploited to improve the trained model. However, since they adopt all pseudo-labeled data in the training without considering the quality of pseudo-labels, there may be many incorrect labels, which will lead to noisy training [17]. Moreover, given the massive quantity of the clients, collecting local models from all the clients will result in extremely high communication cost,

- *A preliminary version of this paper titled "Enhancing Federated Learning with In-Cloud Unlabeled Data" was accepted by IEEE ICDE 2022 DOI: 10.1109/ICDE53745.2022.00015.*
- *Y. Xu, L. Wang, H. Xu, J. Liu, Z. Wang and L. Huang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China, 230027, and also with Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu, China, 215123. E-mails: xuyangcs@ustc.edu.cn; wanglun0@mail.ustc.edu.cn; xuhongli@ustc.edu.cn; jcliu17@ustc.edu.cn; cswangzy@mail.ustc.edu.cn; lshuang@ustc.edu.cn.*
- *H. Xu is the corresponding author.*

which is infeasible for FL systems.

In this paper, we consider a practical FL scenario where there are limited labeled data on clients and large-scale unlabeled data in cloud, and the clients are equipped with limited computation, communication and storage resource. We fuse FL and semi-supervised learning, and leverage both the on-device labeled data and in-cloud unlabeled data to better boost the performance of DL models, even when the labeled data are not independent and identically distributed (non-IID) across clients. During FL training, the quality of pseudo-labels (depicted as *confidence*) is low at the early stage and increases gradually as the training progresses [17]. If a large number of low-confidence pseudo-labels are adopted, the training performance will be significantly degraded. Besides, the number of participating clients (referred as *participating fraction*) has an impact on training cost, including time and communication. With the increasing participating fraction, the communication cost will increase and the time will be shorten when achieving the same accuracy [18]. Note that different FL tasks usually have various preferences towards communication cost and time cost, and their preferences may change over time. Therefore, the approaches [18], [19], [20] with fixed strategies can barely achieve satisfactory training performance.

To improve training efficiency as well as model accuracy, we propose an adaptive FSSL system termed *Ada-FedSemi*, which employs a multi-armed bandit (MAB) based online learning algorithm to adaptively determine the participating fraction (*i.e.*, P) and confidence threshold (*i.e.*, C). Furthermore, considering the heterogeneity among edge devices, we propose a simple yet effective approach to assign local models of different depths for the participating clients. In general, high-performance clients train models with more layers while the clients with low capability only train the first few layers, which alleviates the impact of stragglers. To update the global model, the PS aggregates these heterogeneous local models in a layer-wise manner. In this way, training cost in FL is further reduced without sacrificing model accuracy. The main contributions of this paper are summarized as follows:

- Considering the constrained computation and communication resource of clients at the network edge, we propose to exploit limited on-device labeled data and large-scale in-cloud unlabeled data to boost the training performance of FL in a semi-supervised way.
- To adapt to different cost preferences of FL tasks, we present a multi-armed bandit based online algorithm to adaptively determine the participation fraction of clients and the confidence threshold of pseudo-labels to improve training efficiency and model accuracy.
- To alleviate the impact of heterogeneous clients on FL training, we propose to train local models of distinct depths in accordance with clients' capabilities. Through layer-wise aggregation, the global model is recovered and updated.
- We implement an FL hardware prototype system and conduct extensive experiments on benchmark models and datasets. The experimental results demonstrate that (i) given the same resource budget, Ada-FedSemi can improve test accuracy by 3%$\sim$14.8%; (ii) when achieving the same test accuracy, Ada-FedSemi saves up to 48% training cost, compared with the baseline methods; (iii) under the scenario with heterogeneous clients, the proposed method further speeds up the model training by 1.3$\times\sim$1.5$\times$.

The rest of the paper is organized as follows. Section 2 introduces the adaptive FSSL system and formulates the optimization problem. Section 3 describes the MAB based online algorithm that adaptively determines the participation fraction of clients and pseudo-labeling confidence threshold. In Section 4, we propose an extended method to further deal with client heterogeneity. The experimental evaluation is presented in Section 5. Section 6 reviews some related works and Section 7 gives the conclusions.

2 System Description and Problem Definition

In this section, we first introduce the FSSL system and the main training procedure of Ada-FedSemi. Then, we conduct several experiments to show the motivation of utilization of unlabeled data and further present the impact of participating fraction as well as confidence threshold. Finally, we formally describe the problem to be solved in this paper.

2.1 System Description for FSSL

An FSSL system usually includes a parameter server (PS) and a set of M distributed clients (*e.g.*, IoT devices and edge nodes) $\mathcal{V} = \{v_1, v_2, \ldots, v_M\}$, which collaboratively train DL models over the networks. Each client $v_m \in \mathcal{V}$ trains a local model on its own private dataset $\mathcal{D}_m$ with N_m labeled data, and only needs to synchronize model parameters with the PS rather than sharing the original data.

Let $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$ denote the whole training dataset, where $\mathcal{D}_L = \mathcal{D}_{1,L} \cup \mathcal{D}_{2,L} \cup \cdots \cup \mathcal{D}_{M,L}$ is the labeled dataset distributed across clients and $\mathcal{D}_U$ is the unlabeled dataset collected in the cloud. For the sake of description, we assume that there is no intersection between local datasets. Thus, there are $N_L = \sum_{m=1}^{M} N_m$ data samples in $\mathcal{D}_L = \{(x_i, y_i)\}_{i=1}^{N_L}$, where x_i is the features of the i-th data sample and $y_i = [y_{i,1}, \ldots, y_{i,Q}] \in \{0,1\}^Q$ is a one-hot label, and Q is the total number of classes. $y_{i,q} = 1, q \in [1, Q]$ means that the data sample x_i belongs to class q. For the unlabeled dataset, there are N_U data samples in $\mathcal{D}_U = \{x_j\}_{j=1}^{N_U}$, which lack the ground-truth labels. Let $F(w, x, y)$ denote the loss function over the data (x, y), and w is the model parameter. When considering FL on labeled local data, for each client v_m, its local loss function is defined as:

$$f_m(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_m} F(w, x, y). \tag{1}$$

In order to utilize unlabeled data during the model training, pseudo-labeling is a general and efficient method [17], [21], in which there are two alternating steps, including training and labeling. In the training step, models are trained on both labeled and pseudo-labeled data, which is similar to traditional supervised learning but has different loss functions. In the labeling step, a trained model, also called teacher model, is used to produce predictions for unlabeled data. For a certain data sample x_j, the prediction

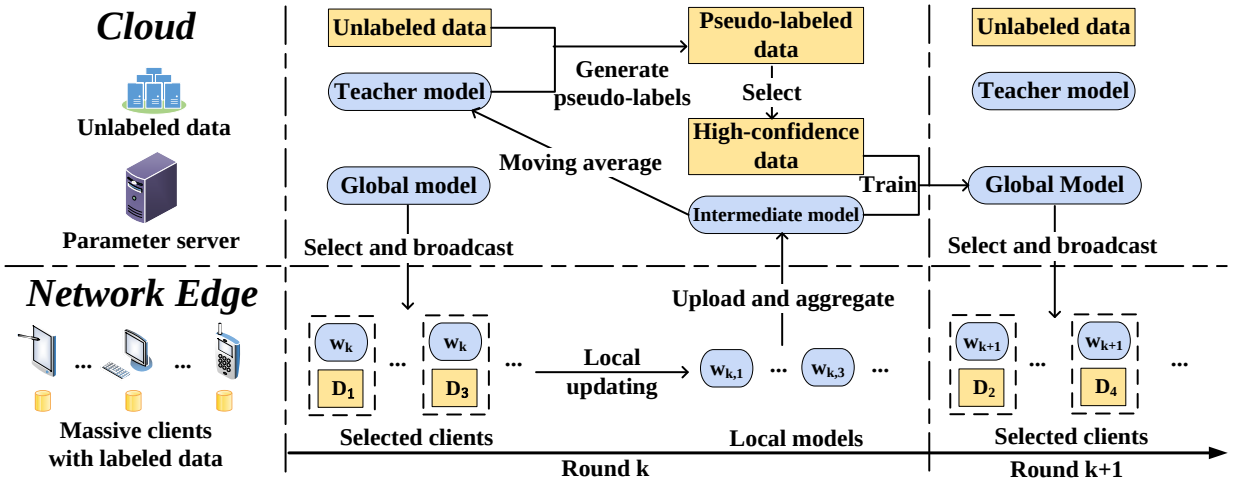


Fig. 1: System workflow of Ada-FedSemi.

by the teacher model is $p_j = [p_{j,1}, \ldots, p_{j,Q}] \in [0,1]^Q$ and $\sum_{q=1}^{Q} p_{j,q} = 1$. The pseudo-label $\hat{y}_j$ of x_j is defined as:

$$\hat{y}_j = \arg\max_q p_{j,q}. \tag{2}$$

When the model is trained on unlabeled data, the pseudo-labels are treated as their training targets. Therefore, the FSSL system aims to obtain the optimal global model w^* by minimizing the loss function on both labeled and unlabeled datasets:

$$w^* := \arg\min_w \mathbb{E}_{(x,y)\sim\mathcal{D}_L} F(w,x,y) + \mathbb{E}_{(x,\hat{y})\sim\mathcal{D}_U} F(w,x,\hat{y}). \tag{3}$$

2.2 System Workflow of Ada-FedSemi

In Ada-FedSemi, the model will be trained on both on-device labeled data and in-cloud unlabeled data iteratively as shown in Fig. 1. The process involves many training rounds, and at each round k, it mainly consists of the following four steps:

(1) **Model Broadcast and Local Updating**. Let P_k denote the participating fraction and $\mathcal{V}_k$ denote the set of selected clients at round k. In this step, the PS selects a fraction of clients, and broadcasts the global model w_k to them. Most researches [4], [22] select specific clients based on a predefined P_k. In general, larger fractions lead to faster convergence but result in more traffic consumption, and vice versa [18]. Thus, considering the properties of FL tasks and the limited resource in the distributed data system, our algorithm concentrates on determining P_k. With a determined P_k, we randomly select clients as in most prior works [4], [18], [22], while advanced client selection strategies [23], [24] can also be applied to further improve the performance.

For each selected client v_m, it first initializes $w_{k,m}(0) = w_k$ and then performs local updating on its local dataset by stochastic gradient descent (SGD) [25]:

$$w_{k,m}(\tau'+1) = w_{k,m}(\tau') - \eta_k \nabla F_m(w_{k,m}(\tau')), \tag{4}$$

where η_k is the learning rate, $\nabla F_m(w_{k,m}(\tau'))$ is the stochastic gradient, $\tau' \in [0,\tau)$ and τ is the number of local updating. Finally, client v_m gets its updated model $w_{k,m}$.

(2) **Model Uploading and Global Aggregation**. After finishing local updating, clients in $\mathcal{V}_k$ upload their local models to the PS, and the PS aggregates these models based on the number of samples in their local datasets as follows:

$$w_{k+\frac{1}{2}} = \frac{\sum_{v_m\in\mathcal{V}_k} N_m w_{k,m}}{\sum_{v_m\in\mathcal{V}_k} N_m}, \tag{5}$$

where $w_{k+\frac{1}{2}}$ is referred as the intermediate model. Then, the teacher model is updated by the intermediate model, which will be elaborated in Section 3.

(3) **Pseudo-labels Generation and Selection**. At the PS, in terms of the teacher model, we make predictions for the unlabeled data and then generate pseudo-labels. Since $\hat{y}_j$ may not be the ground-truth label, we need to estimate the confidence of the pseudo-labels, which indicates how likely a pseudo-label is true. Specifically, we regard the probability of label $\hat{y}_j$ in the prediction as its confidence [26]:

$$a_j = \max_q p_{j,q}. \tag{6}$$

Generally, high-confidence pseudo-labels are more likely to be the ground-truth labels and vise versa.

To mitigate data noise introduced by pseudo-labeling, at round k, we only train the model on the high-confidence pseudo-labeled data samples, whose confidence a_j is over a threshold C_k [27].

(4) **Semi-supervised Model Training**. The intermediate model derived from the aggregation of local models may suffer from poor generalization, since the local models can easily overfit to the insufficient on-device labeled data [6]. Thus, we expect to improve the model's generalization ability by learning additional knowledge from the massive in-cloud unlabeled data. Specifically, the intermediate model in step (2) is further trained on pseudo-labeled data at the PS in a semi-supervised way:

$$w_{k+1} = w_{k+\frac{1}{2}} - \eta_k \nabla F_s(w_{k+\frac{1}{2}}), \tag{7}$$

where $F_s(w)$ is the loss function on $\mathcal{D}_{k,U} = \{(x_j, \hat{y}_j) | x_j \in \mathcal{D}_U \text{and } a_j > C_k\}$. $\mathcal{D}_{k,U}$ is the high-confidence pseudo-labeled dataset at the server. As a result, the updated global model w_{k+1} trained on both labeled and unlabeled data is obtained. These four steps are executed repeatedly until the model converges.

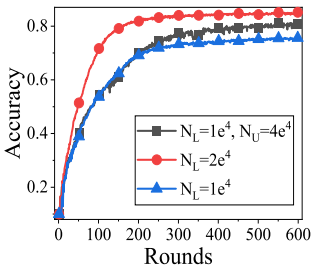


Fig. 2: Test Accuracy *vs.* Rounds.

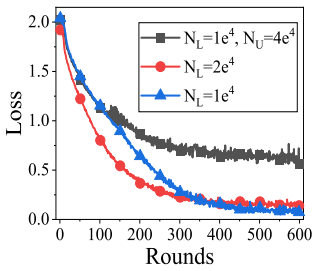


Fig. 3: Training Loss *vs.* Rounds.

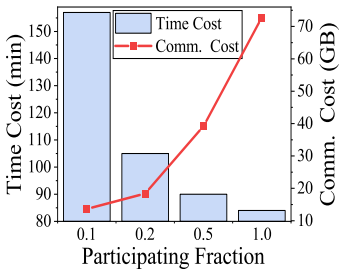


Fig. 4: Time and comm. cost to achieve 60% accuracy.

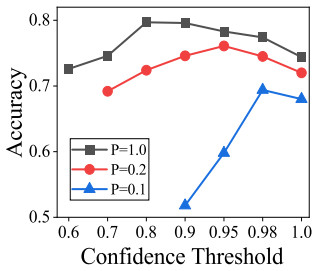


Fig. 5: Accuracy with different P and C.

2.3 Motivation for the Design of Ada-FedSemi

In this section, we conduct several sets of experiments to present the motivation of federated model training with the mixture of labeled and unlabeled data (Figs. 2-3), and analyze the impacts of participating fraction (*i.e.*, P) and confidence threshold (*i.e.*, C) (Figs. 4-5) on training performance. For the sake of simplicity, P and C without subscript are used to indicate that their values keep fixed during the training. We train VGG9 models for 600 rounds on the dataset CIFAR10 with 20 clients, and the details of experiments are introduced in Section 5.

We begin with experiments on labeled datasets with different scale and an unlabeled dataset. The first experiment involves 10,000 labeled data samples and 40,000 unlabeled data samples, while the second and the third experiments separately involve 20,000 and 10,000 labeled data samples without unlabeled data. The confidence threshold for pseudo-labeling is set as 0.8. In Fig. 2, with the increasing number of labeled data from 10,000 to 20,000, the accuracy of the trained model is improved significantly. Besides, we also find that utilizing additional unlabeled data can also achieve higher test accuracy, compared with training only on labeled data. This motivates us to exploit massive in-cloud unlabeled data when the scale of labeled data is limited.

However, in Fig. 3, we observe that the trend of training loss is not consistent with that of test accuracy. For example, given the same scale of labeled data, the model trained with additional unlabeled data achieves worse training loss but higher test accuracy. Since the ultimate goal of model training is to make predictions for unseen data, training loss is not a good metric to measure the performance of a trained model. Instead, we adopt a validation dataset to evaluate the training model and guide the decisions of P and C.

Moreover, we conduct another set of experiments on 10,000 labeled data to analyze the impacts of P and C on training performance. In Fig. 4, we present the time cost and communication cost to achieve 60% accuracy given different values of P. When achieving the same test accuracy, larger P always leads to faster convergence but results in more communication cost. Thus, we should determine P carefully to balance the trade-off between training cost and model accuracy regarding the desired cost preference.

Furthermore, we conduct experiments with different values of P and C to analyze their combined influence. Note that $C = 1.0$ means the model is only trained on labeled data since the confidence of pseudo-labels cannot exceed 1.0. The results are shown in Fig. 5, where the missing test accuracy indicates the model fails to converge given the corresponding values of P and C. We observe that, with small P (*e.g.*, 0.1), the test accuracy is more sensitive to the changes of C than that with large P (*e.g.*, 0.2 and 1.0). Conversely, when all clients participate in the training (*i.e.*, $P = 1.0$), the test accuracy is more robust to the change of C. This set of experiments shows that the values of P and C should be optimized simultaneously so as to achieve satisfactory model accuracy.

2.4 Problem Definition

In FL systems, the clients are usually equipped with limited and heterogeneous capabilities of computation and communication. Let $t_{k,m}$ denote the time cost of client v_m at round k, which includes the time for model broadcasting, updating and uploading. Since the operations of computation and communication at clients can be executed in parallel, the time cost of clients depends on the slowest participating client (*i.e.*, the straggler). Thus, the time cost at round k can be calculated as:

$$t_k = \max_{v_m \in \mathcal{V}_k} \{t_{k,m}\} + t_{k,p}, \tag{8}$$

where $t_{k,p} = t_p |\mathcal{D}_{k,U}|$ is the time cost for model training on the in-cloud pseudo-labeled data, and t_p is the time for processing a single data sample at the PS. We ignore the time cost for generating pseudo-labels, which will be elaborated in Section 3.

Since the clients are usually connected with the PS via cellular network, with the increasing number of participating clients, the network may get congested and the communication cost will increase. Given the size, *i.e.*, W, of the local model, the total communication cost can be expressed as:

$$b_k = \lceil P_k M \rceil W, \tag{9}$$

where $\lceil P_k M \rceil$ is the number of participating clients at round k. As shown in Fig. 4 and also demonstrated in the work [18], more communication cost usually leads to faster model convergence, *i.e.*, less time cost, and vice versa. Considering that different FL tasks have diverse cost preferences (*e.g.*, fast convergence or low communication cost), we consider the weighted cost of the both as in [18]:

$$\Phi_k = \alpha t_k + (1 - \alpha) b_k, \tag{10}$$

where $\alpha \in [0, 1]$ is the bias factor to adjust the preference towards time cost and communication cost. $\alpha = 0$ means that only the communication cost is taken into consideration while $\alpha = 1$ indicates that the model is expected to converge as fast as possible without considering com-

munication cost. Note that the setting of cost preference is based on the properties of FL tasks [18]. For example, in a cellular network, traffic consumption is probably the main concern for the clients participating in FL. In contrast, in a search-and-rescue task which aims to collaboratively learn a search scheme as quickly as possible, achieving timely result would be the first priority. Thus, the cost preferences are mainly determined by the requirements of FL tasks and our algorithm can adapt to different cost preferences online, which is demonstrated in Section 5.

In the most prior works [25], [28], the optimization objective of model training aims to minimize the loss function over training data, *i.e.*, the objective defined in Eq. (3). However, as demonstrated in Section 2.3, training loss fails to exactly evaluate the prediction ability of the model on unseen data (*i.e.*, generalization ability), especially when the scale of training dataset is varying. Instead, the validation dataset can be used to provide an unbiased evaluation of the model during the training [29]. At round k, we denote the accuracy of the global model on the validation dataset as u_k.

As stated in Section 2.3, P_k and C_k have a significant impact on the performance of model training and need to be judiciously determined. As a result, the optimization problem is formulated as follows:

$$\min_{P_k,C_k,K} \sum_{k=1}^{K} \alpha t_k + (1-\alpha) b_k \tag{11}$$
$$\textbf{s.t.} \begin{cases} u_K \geq \epsilon, \\ \sum_{k=1}^{K} t_k \leq T, \\ \sum_{k=1}^{K} b_k \leq B, \\ P_k, C_k \in [0,1], \forall k, \end{cases}$$

where ϵ is the target accuracy on validation dataset. T and B are the time and communication budgets for federated model training, respectively.

3 Algorithm Description

Since P_k and C_k play an important role in the model training, we propose an adaptive federated semi-supervised learning system (termed as Ada-FedSemi) to utilize both on-device labeled data and in-cloud unlabeled data efficiently. Specifically, given the desired cost preference and limited resource budgets, Ada-FedSemi employs a multi-arm bandit (MAB) based online algorithm to adaptively determine the participating fraction (*i.e.*, P_k) of clients and the confidence threshold (*i.e.*, C_k) of pseudo-labels at each round.

3.1 Overall Training Process of Ada-FedSemi

The overall training process of Ada-FedSemi is described in Alg. 1. Our goal is to achieve the target model accuracy while minimizing training cost. At round k, based on the value of P_k, the PS first randomly selects a subset of clients to participate in FL, and then aggregates the local models to derive the intermediate model $w_{k+\frac{1}{2}}$ at the end of local updating (Line 4-7). On the basis of the intermediate model, the teacher model $\tilde{w}_k$ is updated (Line 8).

As suggested in [30], averaging the models across different rounds can generate a more accurate and reliable model than directly using the latest model. This is because models tend to forget past learned knowledge and fit the recent training data [31]. For example, at a certain round, if only one client is chosen to participate in the federated model training and the data on that client is highly skewed (*e.g.*, all data belong to only one class), the trained model will prefer to classify the input data as that class. Thus, we adopt the exponential moving average of the intermediate models across rounds as the teacher model, which is updated as follows and can achieve reliable performance improvement:

$$\tilde{w}_k = \gamma w_{k+\frac{1}{2}} + (1-\gamma)\tilde{w}_{k-1}, \tag{12}$$

where $\gamma \in (0,1]$. Unlike some existing methods, where the teacher is a well-trained model, our teacher model will be gradually improved during the training process without incurring additional training cost [30].

Subsequently, the teacher model $\tilde{w}_k$ is used to generate pseudo-labels for unlabeled data (Line 9-10). As generating predictions for massive unlabeled data is time-consuming, we propose two strategies to reduce the cost for pseudo-labeling. (1) Pseudo-labeling can be executed periodically (*e.g.*, every R rounds) since the prediction ability of the teacher model will not improve significantly in several successive rounds. Furthermore, (2) pseudo-labeling can be performed in parallel with other steps like model training, broadcasting and uploading. As a result, the time cost of pseudo-labeling can be ignored. In terms of the threshold C_k, we select the high-confidence pseudo-labeled data, upon which the intermediate model is further trained to produce the global model w_{k+1} for next round (Line 11-12).

Since it is inevitable to generate incorrect pseudo-labels for unlabeled data, the model trained on these data will accumulate errors (also known as *confirmation bias*) [32]. In other words, the model keeps learning from incorrect pseudo-labels, and thereby the confidence of wrong predictions by the model continuously increases. In order to prevent error accumulation, we propose to adjust the learning rate periodically (Line 13). Specifically, we use the cosine anneal learning rate [33] to schedule the training process, which can help models jump out of local optimum and explore other regions [34]. Concretely, the learning rate is scheduled as follows:

$$\eta_k = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{k \bmod \tilde{K}}{\tilde{K}}\pi)), \tag{13}$$

where η_{min} and η_{max} are the minimum and the maximum learning rates, respectively. $\tilde{K}$ is the restart interval and k is the current training round. In each interval, the learning rate is initialized as η_{max} at the beginning and then gradually decreased to η_{min}. As a result, models trained with the trick of learning rate restart always achieve better accuracy as demonstrated in Section 5.

3.2 MAB based Decision Making

To adapt to system dynamics and different cost preferences, we need to adaptively determine the values of P_{k+1} and C_{k+1} (Line 14-17 of Alg. 1). As shown in Section 2.3, more participating clients (*i.e.*, larger P) always contribute to higher model accuracy but also result in more training cost. Besides, with the increasing value of C, the number of selected data decreases and the quality of pseudo-labels increases, since the high-confidence pseudo-labels are more

Algorithm 1 Training Process of Ada-FedSemi

Input: Client sets $\mathcal{V}$ with their on-device data and in-cloud unlabeled data

Output: The well-trained model w

1: Initialize MAB agents with action sets S_P and S_C, global model w_k, participating fraction P_k, confidence threshold C_k, update interval R, target accuracy ϵ, current accuracy $u_k = 0$, $k = 1$
2: **while** $u_k < \epsilon$ **do**
3: **Processing at the Parameter Server**
4: Select $\lceil P_k M \rceil$ clients into the set $\mathcal{V}_k$ randomly
5: Broadcast w_k to $v_m \in \mathcal{V}_k$
6: Collect local models $w_{k,m}, \forall v_m \in \mathcal{V}_k$
7: Obtain the intermediate model $w_{k+\frac{1}{2}}$ as Eq. (5)
8: Update the teacher model $\tilde{w}_k$ as Eq. (12)
9: **if** k mod $R = 0$ **then**
10: Update pseudo-labels of unlabeled data
11: Select pseudo-labeled data into $\mathcal{D}_{k,U}$ based on C_k
12: Train $w_{k+\frac{1}{2}}$ on $\mathcal{D}_{k,U}$ as Eq. (7), and obtain w_{k+1}
13: Adjust learning rate η_k as Eq. (13)
14: Validate $w_{k+\frac{1}{2}}$ and record the accuracy as $u_{k,P}$
15: Validate w_{k+1} and record the accuracy as $u_{k,C}$
16: Calculate the change of accuracy $\Delta u_{k,P}$ and $\Delta u_{k,C}$
17: Update MAB agents, and determine P_{k+1} and C_{k+1}
18: $u_k = u_{k,C}$, $k = k + 1$
19: **Processing on Each Client** v_m
20: **if** Receive w_k from the PS **then**
21: Update the local model as Eq. (4)
22: Upload the trained model $w_{k,m}$

likely to be the ground-truth labels in comparison to the low-confidence pseudo-labels. Moreover, the values of P and C are demonstrated to have an impact on each other. With more clients participating in FL (*i.e.*, larger P), the quality of predictions by the teacher model will increase and thus the number of errors in pseudo-labels will be reduced, which will affect the decision of C. Therefore, P and C are expected to get optimized simultaneously.

However, due to the complex influence factors of federated model training (*e.g.*, model architecture, datasets, optimizer and number of clients), it is infeasible to obtain the optimal values of P and C in advance of the training. Therefore, we propose a multi-armed bandit (MAB) based online learning algorithm to determine P and C without any prior knowledge of the FL system. In each round, the MAB algorithm chooses an action, *i.e.*, arm, from an action set and collects a reward. Then, based on the action decisions and the corresponding rewards across different rounds, the strategies for choosing actions are updated.

We take the optimization problem in Eq. (11) as a classic MAB problem, where the values of P and C can be regarded as actions. The MAB algorithm is originally developed for discrete decision spaces. However, the values of P and C are continuous, which are within $[0, 1]$. Thus, we need to partition the continuous decision space into discrete action sets S_P and S_C, respectively. In fact, the decision space can be further zoomed in to focus on a much smaller range. For example, in Fig. 5, we find that the test accuracy of the trained model always decreases when the value of C is below 0.8. Therefore, we only consider the decision space of C in the range of 0.8 to 1.0. The decision making process of our MAB algorithm is summarized in Alg. 2. In each round, the MAB agent at the PS first makes the decision about which action is performed and then obtains a reward in response to the action (Line 3-4). According to the rewards, the MAB agent updates probabilities for the corresponding actions (Line 5-6).

We adopt a validation dataset to evaluate the accuracy of the trained models and calculate the rewards (*i.e.*, accuracy improvement) of different actions. At round k, we denote the accuracy of the models $w_{k+\frac{1}{2}}$ and w_{k+1} as $u_{k,P}$ and $u_{k,C}$, respectively. The accuracy improvement of the two models is $\Delta u_{k,P}$ and $\Delta u_{k,C}$:

$$\Delta u_{k,P} = u_{k,P} - u_{k-1,C}, \tag{14}$$

$$\Delta u_{k,C} = u_{k,C} - u_{k,P}. \tag{15}$$

Since the intermediate model $w_{k+\frac{1}{2}}$ is aggregated from local models, we recognize the improvement of this model, *i.e.*, $\Delta u_{k,P}$, as the outcome of the decision of P_k. Meanwhile, the model w_{k+1} is trained on the pseudo-labeled data selected by C_k, and thus we recognize $\Delta u_{k,C}$ as the outcome of the decision of C_k. Since the algorithm for determining P_k and C_k is the same, we use Δu_k and S for simplicity to introduce our algorithm in the following. We define the reward of the decision at round k as follows:

$$r_k = \begin{cases} \frac{\Delta u_k}{\Phi_k}, & \text{if } \Delta u_k \geq 0, \\ \Delta u_k \cdot \Phi_k, & \text{otherwise.} \end{cases} \tag{16}$$

As the reward design is the key to the success of MAB algorithms [35], herein, we explain the rationality of the reward function. The intuition of our reward design has two folds and the goals of the two-fold reward functions are consistent, *i.e.*, improving the model performance in a cost-efficient way. (1) When achieving the same accuracy improvement (*i.e.*, $\Delta u_k \geq 0$), the decisions which consume less training cost should be given higher rewards. In another word, we expect high accuracy improvement and small training cost. (2) While some inappropriate actions may degrade accuracy, *i.e.*, $\Delta u_k < 0$, and we still use $\Delta u_k / \Phi_k$ as the reward, a smaller training cost Φ_k will lead to a higher penalty (penalty means negative reward). This is not consistent with our design goal, *i.e.*, efficient training. Thus, in case of $\Delta u_k < 0$, we denote $\Delta u_k \cdot \Phi_k$ as the reward.

Traditional MAB algorithms estimate the actual reward of an action by averaging its received rewards across rounds. However, in this paper, the reward distribution of actions is not identical across different rounds. Firstly, the improvement speed of model accuracy is not the same during the training process. In general, the increase of model accuracy is fast at the beginning of the training and becomes slow as training progresses. Besides, the optimal decision may change over time since the quality of pseudo-labels will improve and the cost preference may vary during the training. Therefore, this is a non-stationary MAB problem [36], and it is not rational to simply average rewards of each action across rounds as traditional MAB algorithms do. Instead, we concentrate more on the recent rewards which are assigned with larger weights, and gradually decay the weights for the past rewards [37]. At round k, for each action a in the action set S, its estimated reward $\hat{r}_{k,a}$ is calculated

Algorithm 2 MAB Agent

1: Action set S and $\hat{r}_{k,a}$, $p_{k,a}$ for each action
2: **for** round $k \in \{1, \ldots, K\}$ **do**
3: Select an action based on the probability $p_{k,a}, \forall a \in S$
4: Receive the reward r_k based on Eq. (16)
5: Update estimated rewards $\hat{r}_{k,a}$ as Eq. (17)
6: Calculate the probability $p_{k,a}$ of action as Eq. (18)

as follows:

$$\hat{r}_{k,a} = \begin{cases} \hat{r}_{k-1,a} + \beta(r_k - \hat{r}_{k-1,a}) & , \text{ if } a_k = a, \\ \hat{r}_{k-1,a}, & \text{otherwise}, \end{cases} \quad (17)$$

where $\beta \in (0, 1]$ is the decay factor and a_k is the action chosen for round k.

Thus, the goal of our MAB algorithm is to maximize the total received rewards via a judicious trade-off between exploration and exploitation. Exploitation means pulling the best action known so far while exploration aims to explore different actions to find better solutions. Specifically, we adopt the Boltzmann exploration strategy [38], which is wildly used for balancing exploration and exploitation. The probability of choosing action $a \in S$ at round k is calculated as follows:

$$p_{k,a} = \frac{e^{\psi \hat{r}_{k,a}}}{\sum_{a' \in S} e^{\psi \hat{r}_{k,a'}}}. \quad (18)$$

Particularly, with $\psi = 0$, the actions are uniformly chosen all the time, while $\psi \to \infty$ means that the MAB agent will always output the action with the highest reward without any exploration.

4 Adaptation to Heterogeneous Clients

As indicated in Eq. (8), the time cost of a round depends on the slowest client (*i.e.*, the straggler). Since clients typically consist of various types of devices equipped with different hardwares and bandwidth, they exhibit significant divergence on the capabilities of both computation and communication. According to AI-Benchmark[1], the computation capabilities vary significantly across different mobile CPUs. For instance, when executing the same tasks, HiSilicon Kirin 9000 is 10× faster than Snapdragon 625. Besides, according to the report of Cisco [39], by 2023, there are still about 30% mobile devices that communicate through 2G or 3G (the bandwidth is less than 10 Mbps) while the 5G connection (the bandwidth is more than 500 Mbps) only accounts for 10%. Due to the computation and communication heterogeneity among clients, the stragglers may slow down the training process, resulting in a long waiting time of fast clients and thus inefficient utilization of available resources.

In general, we expect that heterogeneous clients can train local models with different sizes in accordance with their capabilities. However, it is usually difficult to aggregate models of different architectures [40]. Herein, we propose a simple yet effective approach, called HeteroAda-FedSemi, where local models are generated from the same global model but of different depths. Specifically, inspired by recent researches on multi-exit models [41], we add auxiliary classifiers in the intermediate layers of a conventional (*i.e.*, single-exit) model. Then, each exit and its preceding layers form an independent model, which can be trained on clients without relying on the subsequent layers. In this way, local models of different depths are trained on heterogeneous clients and the global aggregation can be easily performed in a layer-wise manner. In the following, we first introduce the architecture of the multi-exit model, and then present the assignment of heterogeneous local models.

1. https://ai-benchmark.com

4.1 Multi-exit Model

Multi-exit models have been proven effective to accelerate inference through early-exit techniques [42], but remain rarely explored to achieve efficient model training. In our system, we exploit the potential of multi-exit models to perform heterogeneous local training to alleviate the impact of stragglers.

A conventional model can be regarded as a function H that maps the input x into the prediction $H(x)$. We decouple it into $B+1$ parts, including $h^1, \ldots, h^B$ for feature extraction and c for classification:

$$\hat{y} = H(x) = (c \circ h^B \circ h^{B-1} \circ \cdots \circ h^1)(x), \quad (19)$$

where $\hat{y}$ is the prediction from the model. The operator $\circ$ denotes function composition, *i.e.*, $(h^b \circ h^{b'})(\cdot) = h^b(h^{b'}(\cdot))$. The output of h^b is denoted as $z^b, 1 \le b \le B$. We denote $z^0 = x$, $z^b = h^b(z^{b-1})$ and $\hat{y} = c(z^B)$. The function h^b may represent any single or several consecutive layers (*e.g.*, convolutional layers).

To transform a conventional model into a multi-exit model, we add an auxiliary classifier c^b after $h^b, 1 \le b \le B-1$. Then, we feed the output of h^b to the classifier c^b to obtain an early prediction $\hat{y}^b$:

$$\hat{y}^b = c^b(z^b), 1 \le b \le B, \quad (20)$$

where z^b $(1 \le b \le B-1)$ is the input of both c^b and h^{b+1} while z^B is the input of the original classifier, *i.e.*, $c^B = c$. The classifier c^b and its previous parts (including $c^1, \cdots, c^b$ and $h^1, \cdots, h^b$) form a submodel H^b. We use w^b to denote the parameters in the model part b (including h^b and c^b). Regarding the output of the classifier c^b, we calculate its loss on local dataset $\mathcal{D}_m$ at round k as:

$$f^b(w^b_{k,m}) = \mathbb{E}_{(x,y)\sim\mathcal{D}_m} F(w^b_{k,m}, x^b, y), \quad (21)$$

where $w^b_{k,m}$ is the parameters of the model part b on client v_m and round k, and $x^b = z^{b-1}$ is the input of the model part b (*i.e.*, the output of h^{b-1} for $b > 1$). If the client v_m trains the submodel $f^{b_{k,m}}$ at round k, its overall optimization goal is:

$$w^*_{k,m} = \arg\min_{w_{k,m}} \sum_{b=1}^{b_{k,m}} f^b(w^b_{k,m}). \quad (22)$$

Note that the transformation from a conventional model to the multi-exit model can be easily applied on any CNN-based model [41], *e.g.*, VGG [43] and ResNet [44], which have been widely used in various FL applications [45], [46].

4.2 Submodel Assignment for Heterogeneous Clients

Models of different depths require significantly diverse cost of both computation and communication. For example, we add two intermediate classifiers for VGG9, and thus three submodels (including the full model) can be generated. The required computation cost of the smallest submodel is only

Algorithm 3 Clustering clients by their capabilities in HeteroAda-FedSemi

1: Estimate $\hat{t}_{k,m}$ as Eq. (23) for $v_m \in \mathcal{V}_k$
2: **if** $|\mathcal{V}_k| > B$ **then**
3: Initialize B clusters: $U^1, U^2, \cdots, U^B$
4: Randomly select B clients, sort their estimated time and obtain centroids of clusters: $t^1, t^2, \cdots, t^B$
5: **else**
6: Return $|\mathcal{V}_k|$ clusters: $U^{B-|\mathcal{V}_k|+1}, \cdots, U^{B-1}, U^B$
7: **while** The assignment of clusters is changing **do**
8: Compute the distance between clients' estimated time and all centroids
9: Assign each client to the closest cluster (centroid)
10: Update the centroid for each cluster by averaging all clients' estimated time in that cluster
11: Return B clusters: $U^1, U^2, \cdots, U^B$

32.3% of that of the full model. Besides, the corresponding communication cost is also reduced from 13.41MB to 0.11MB. The details of the parameter size and the computation load of submodels of VGG9 are listed in Table 9.

For heterogeneous clients, we assign local models of different depths based on their capabilities. In general, we assign the full model for high-performance clients while assigning submodels for other clients equipped with the relatively low capabilities. The detailed algorithm is presented in Alg. 3.

Clients are first clustered based on their capabilities, and then a submodel is assigned for each cluster. Specifically, to indicates clients' capabilities (Line 1), we estimate the completion time of clients for training the full model as:

$$\hat{t}_{k,m} = \frac{\Pi}{\widehat{\psi}_{k,m}} + \frac{\Omega}{\widehat{\gamma}_{k,m}}, \tag{23}$$

where Π and Ω are the training load and parameter size of the full model, respectively. $\widehat{\psi}_{k,m}$ and $\widehat{\gamma}_{k,m}$ are the estimated computation capability and bandwidth, respectively. By recording the actual time of model training (*i.e.*, $t^{cp}_{k-1,m}$) and transmitting (*i.e.*, $t^{cm}_{k-1,m}$) at the last round (*i.e.*, $k-1$), the estimated computation capability and bandwidth can be calculated as:

$$\widehat{\psi}_{k,m} = \frac{\Pi_{k-1,m}}{t^{cp}_{k-1,m}}, \tag{24}$$

$$\widehat{\gamma}_{k,m} = \frac{\Omega_{k-1,m}}{t^{cm}_{k-1,m}}, \tag{25}$$

where $\Pi_{k-1,m}$ and $\Omega_{k-1,m}$ are local submodel's training load and parameter size on client v_m at round $k-1$, respectively.

If there are over B clients in the selected set $\mathcal{V}_k$, we will randomly select B clients to initialize B clusters, *i.e.*, one client in each cluster. Then, we use clients' estimated time $\hat{t}_{k,m}$ to set centroids of clusters (Line 2-4). Otherwise, $|\mathcal{V}_k|$ clusters are returned and each contains a client (Line 5-6). Using the estimated time, the distance between clients and all centroids are calculated (Line 8). Each client is assigned to the closest centroid (Line 9). The centroid of each cluster is updated by taking average of all clients' estimated time in that cluster (Line 10). The clusters are updated repeatedly until the assignment of clusters is not changing. At last, we obtain client clusters, *i.e.*, $U^1, U^2, \cdots, U^B$. The clients in a cluster are equipped with similar capabilities, and thus train a same submodel. Specifically, the clients in U^B train the full model, *i.e.*, H^B, the clients in U^{B-1} train the submodel H^{B-1} and so on. When clients finish local training, the PS will collect these updated submodels and perform layer-wise aggregation. As a result, the updated global model w_{k+1} is obtained for further training.

4.3 Convergence Analysis

Herein, we will analyze the convergence of each part in a multi-exit model, which consists of multiple model parts. The input data of the first part is the original data samples, which keep unchanged during the training. However, the input data of other parts are affected by their preceding parts, due to the optimization of model parameters [47], [48]. In other words, the distribution of the input data of part b ($b > 1$) depends on its preceding parts. At round k, we denote the distribution of the input data of part b on client v_m as $p^b_{k,m}$ and assume the converged distribution is $p^{b,*}$.

To analyze the convergence property of the training of the multi-exit model, we make following assumptions:

Assumption 1. (***L-smoothness***) For a constant $L > 0$ and the model part b, we have:

$$\|\nabla f_m(w^b) - \nabla f_m(\tilde{w}^b)\| \leq L\|w^b - \tilde{w}^b\|, \forall v_m \in \mathcal{V}, \tag{26}$$

where w^b and $\tilde{w}^b$ are two different parameters of the model part b.

Assumption 2. (***Convergence of the previous parts***) As in [47], we consider the variation distance between the local distribution and the converged distribution: $c^b_{k,m} \triangleq \int |p^b_{k,m} - p^{b,*}|dz$, where z denotes the data samples drawn from the distribution p, *i.e.*, $z \sim p$. For ease of analysis, we denote $c^b_k = \max_m c^b_{k,m}$, and assume the input of the model part b will converge:

$$\sum_k c_k^{b-1} < \infty. \tag{27}$$

Assumption 3. (***Unbiased and bounded gradient***) The stochastic gradient at each client is an unbiased estimator of the local full-batch gradient:

$$\mathbb{E}_{(x,y)}[\nabla F_m(w^b_{k,m})] = \nabla f(w^b_{k,m}), \forall m, b, k. \tag{28}$$

For a constant G, we have:

$$\|\nabla F_m(w^b_{k,m})\|^2 \leq G, \forall m, b, k. \tag{29}$$

Lemma 1. Under Assumptions 1-3, we have:

$$\mathbb{E}[f(w^b_{k+1})] \leq f(w^b_k) + \frac{LG\eta^2}{2M} - \eta(\frac{1}{2}\|\nabla f(w^b_k)\|^2 - \frac{Gc_k^{b-1}}{M}), \tag{30}$$

where η is the learning rate. The proof is presented in APPENDIX A.

Theorem 1. Under Assumptions 1-3 and Lemma 1, we have the following result at round K:

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(w^b_k)\|^2 \leq \frac{2f(w^b_0)}{K\eta} + \frac{2G}{MK}\sum_{k=0}^{K-1} c_k^{b-1} + \frac{LG\eta}{M},$$

which is proved in APPENDIX B.

TABLE 1: Performance metrics of FedSemi-P-C and FedAvg-P (*i.e.*, FedSemi-P-1.0) on CIFAR10.

(a) Test accuracy (%)

C \ P	0.1	0.2	0.5	1.0
0.6	-	-	74.5±0.1	75.8±0.1
0.7	-	80.4±0.3	77.0±0.2	82.9±0.1
0.8	-	81.3±0.2	82.9±0.1	**83.3±0.0**
0.9	68.9±0.4	**82.1±0.2**	**83.3±0.0**	83.1±0.1
0.95	73.4±0.5	81.5±0.1	81.6±0.1	81.5±0.0
0.98	**77.6±0.2**	79.1±0.2	81.4±0.1	80.5±0.1
1.0	75.1±0.3	75.4±0.1	75.6±0.2	75.3±0.1

(b) Time cost (min) for 75% accuracy

C \ P	0.1	0.2	0.5	1.0
0.6	-	-	-	419
0.7	-	382	312	251
0.8	-	311	240	196
0.9	-	296	**199**	**186**
0.95	-	**247**	211	190
0.98	**356**	296	221	194
1.0	395	346	326	314

(c) Traffic (GB) for 75% accuracy

C \ P	0.1	0.2	0.5	1.0
0.6	-	-	-	355
0.7	-	68	139	226
0.8	-	58	113	186
0.9	-	57	**98**	**183**
0.95	-	**49**	103	187
0.98	**36**	59	108	194
1.0	41	72	169	324

Corollary 1. Referring to the analyses in the works [49], [50], we define:

$$q_k = \frac{\|\nabla f(w_k)\|^2}{\|\nabla f(w_k^b)\|^2} \quad \text{and} \quad q_0 = \max_k q_k. \tag{31}$$

Besides, we also define:

$$f_0 = \max_b f(w_0^b) \quad \text{and} \quad c_0 = \max_{k,b} c_k^{b-1}. \tag{32}$$

Then, we can obtain the convergence of the entire model:

$$\frac{1}{K}\sum_{k=0}^{K-1} \|\nabla f(w_k)\|^2 \leq \frac{2q_0 f_0}{K\eta} + \frac{2q_0 G c_0}{M} + \frac{q_0 L G \eta}{M}, \tag{33}$$

which is proved in APPENDIX C.

From Theorem 1, we can see that the distribution of input data of the model part b, *i.e.*, c_k^{b-1}, has an impact on the convergence, and the distribution depends on the convergence of all the preceding model parts. Thus, in our algorithm, the part that is closer to the original data input will be trained on more clients, since its convergence will influence the training process of all its subsequent parts. In this way, heterogeneous resources can be fully utilized while model convergence can be guaranteed. Besides, we can also see that the loss value (*i.e.*, the left hand side of Theorem 1) continues to decrease as the numbers of training rounds (*i.e.*, K) and participating clients (*i.e.*, M) increase, which is consistent with prior works of FL [51], [52].

5 Experimental Evaluation

5.1 System Platform

We evaluate the performance of Ada-FedSemi through extensive experiments on an FL hardware prototype system. Specifically, an AMAX deep learning workstation, which is equipped with an Intel(R) Core(TM) i9-10900X CPU, 4 NVIDIA GeForce RTX 2080Ti GPUs and 128 GB RAM, is applied to serve as the PS. Besides, 20 NVIDIA Jetson TX2 developer kits[2] are specified as the clients. Each TX2 client has a 256-core Pascal GPU and a CPU cluster consisting of a 2-core Denver2 and a 4-core ARM CortexA57. By default, TX2 clients work in their full capabilities, *i.e.*, the mode 0 in Table 8. The PS and clients are connected via a Wi-Fi router. The implementation for model training is based on the PyTorch deep learning framework [53], and we use the socket library of Python to build up the communication between clients and the PS.

2. https://docs.nvidia.com/jetson/

5.2 Setup of Experiments

Datasets and Models: We use four benchmark datasets, *i.e.*, CIFAR10 [54], SVHN [55], STL10 [56] and IMDB [57] to evaluate the performance of Ada-FedSemi and baselines:

- **CIFAR10**: It contains 60,000 color images labeled in 10 classes with 6,000 samples per class. By default, we split the whole dataset into four datasets: i) labeled training dataset with 10,000 samples, ii) unlabeled training dataset with 40,000 samples whose labels are discarded, iii) validation dataset with 2,000 samples, and iv) test dataset with 8,000 samples.
- **SVHN**: There are 73,257 digits for training, 26,032 digits for testing, and 531,131 additional data, which are labeled in 10 classes. By default, 5% of training data, *i.e.*, 3,660 digits, are distributed to clients as labeled data. 20,000 and 6,032 digits in testing dataset are used for testing and validation, respectively. The rest digits in the training dataset and additional dataset are all placed at the PS as unlabeled data. As a result, there are 600,728 unlabeled samples at the PS.
- **STL10**: It consists of 13,000 labeled images and 100,000 unlabeled images. The resolution of these color images is 96×96 pixels and there are 10 categories in the labeled dataset. We split the labeled data into three datasets: i) labeled training dataset with 5,000 samples, ii) validation dataset with 2,000 samples, and iii) test dataset with 6,000 samples.
- **IMDB**: There are 50,000 movie review samples (25,000 for training and 25,000 for test), labeled by positive or negative sentiment. We split the whole dataset into four datasets: i) labeled training dataset with 5,000 samples, ii) unlabeled training dataset with 20,000 samples, iii) validation dataset with 5,000 samples, and iv) test dataset with 20,000 samples.

Since data are not always distributed uniformly across clients at the network edge, we will analyze training performance under both IID and non-IID settings. (1) In the IID setting, all labeled data are uniformly distributed to clients. (2) In the non-IID setting, as in [23], a fraction (ζ) of data samples assigned to a client belong to a certain class and the remaining data samples belong to other classes, which is denoted as non-IID-ζ. By default, the data distribution of CIFAR10 and SVHN is non-IID-0.5 while that of STL10 and IMDB is non-IID-0.75 and IID, respectively.

TABLE 2: Performance metrics of FedSemi-P-C and FedAvg-P (*i.e.*, FedSemi-P-1.0) on SVHN.

(a) Test accuracy (%)

C \ P	0.1	0.2	0.5	1.0
0.7	62.7±0.4	80.2±0.2	80.7±0.2	81.8±0.1
0.75	64.5±0.3	81.3±0.3	81.1±0.2	82.0±0.1
0.8	67.5±0.2	81.6±0.2	81.9±0.1	83.2±0.2
0.85	68.5±0.2	82.4±0.1	**83.4±0.2**	**84.5±0.1**
0.9	69.6±0.3	**82.8±0.1**	82.1±0.1	82.6±0.1
0.95	**72.5±0.1**	80.4±0.3	81.3±0.1	81.2±0.0
1.0	70.4±0.2	80.8±0.2	80.9±0.2	81.6±0.1

(b) Time cost (min) for 75% accuracy

C \ P	0.1	0.2	0.5	1.0
0.7	-	63	52	31
0.75	-	59	46	27
0.8	-	54	43	25
0.85	-	52	**37**	**23**
0.9	-	**51**	41	27
0.95	-	59	45	28
1.0	-	58	43	28

(c) Traffic (GB) for 75% accuracy

C \ P	0.1	0.2	0.5	1.0
0.7	-	0.77	1.21	2.07
0.75	-	0.71	1.12	1.94
0.8	-	0.69	1.09	1.76
0.85	-	0.64	**1.02**	**1.69**
0.9	-	**0.59**	1.08	1.81
0.95	-	0.76	1.13	2.20
1.0	-	0.76	1.15	2.04

TABLE 3: Performance metrics of FedSemi-P-C and FedAvg-P (*i.e.*, FedSemi-P-1.0) on STL10.

(a) Test accuracy (%)

C \ P	0.1	0.2	0.5	1.0
0.8	24.7±0.6	44.9±0.4	51.6±0.2	52.3±0.3
0.85	26.3±0.5	51.1±0.3	53.0±0.4	53.7±0.1
0.9	29.7±0.3	54.2±0.3	55.6±0.3	57.5±0.2
0.95	32.2±0.4	56.5±0.3	56.6±0.2	**59.5±0.0**
0.98	39.3±0.4	56.4±0.2	**57.8±0.2**	58.4±0.1
0.99	**41.4±0.3**	**57.7±0.2**	55.9±0.1	56.9±0.2
1.0	39.9±0.4	55.6±0.2	56.3±0.1	56.5±0.1

(b) Time cost (min) for 50% accuracy

C \ P	0.1	0.2	0.5	1.0
0.8	-	-	342	329
0.85	-	549	256	244
0.9	-	340	243	181
0.95	-	324	239	**167**
0.98	-	325	**207**	178
0.99	-	**303**	245	189
1.0	-	355	243	192

(c) Traffic (GB) for 50% accuracy

C \ P	0.1	0.2	0.5	1.0
0.8	-	-	131	247
0.85	-	82	99	187
0.9	-	53	93	146
0.95	-	49	93	**129**
0.98	-	50	**84**	138
0.99	-	**46**	95	150
1.0	-	54	96	148

TABLE 4: Optimal combination of P and C with different cost preferences (α) on CIFAR10.

Range of α ($\times 0.1$)	[0, 1.07)	[1.07, 5.06)	[5.06, 8.68)	[8.68, 10]
Optimal P and C	(0.1, 0.98)	(0.2, 0.95)	(0.5, 0.9)	(1.0, 0.9)

TABLE 5: Optimal combination of P and C with different cost preferences (α) on SVHN.

Range of α ($\times 0.01$)	[0, 2.98)	[2.98, 4.57)	[4.57, 100]
Optimal P and C	(0.2, 0.9)	(0.5, 0.85)	(1.0, 0.85)

TABLE 6: Optimal combination of P and C with different cost preferences (α) on STL10.

Range of α	[0, 0.284)	[0.284, 0.529)	[0.529, 1.0]
Optimal P and C	(0.2, 0.99)	(0.5, 0.98)	(1.0, 0.95)

On CIFAR10 and STL10, we train a VGG9 model [58] with 3.49 million parameters while a lightweight CNN model with 0.54 million parameters is trained on SVHN. For IMDB, we adopt the CNN model in [59]. Besides, the SGD optimizer with momentum is adopted in our experiments to optimize models, and the momentum is set as 0.9. The restart interval for learning rate is set as 100. The maximum and minimum learning rates are set as 0.05 and 0.0001, respectively.

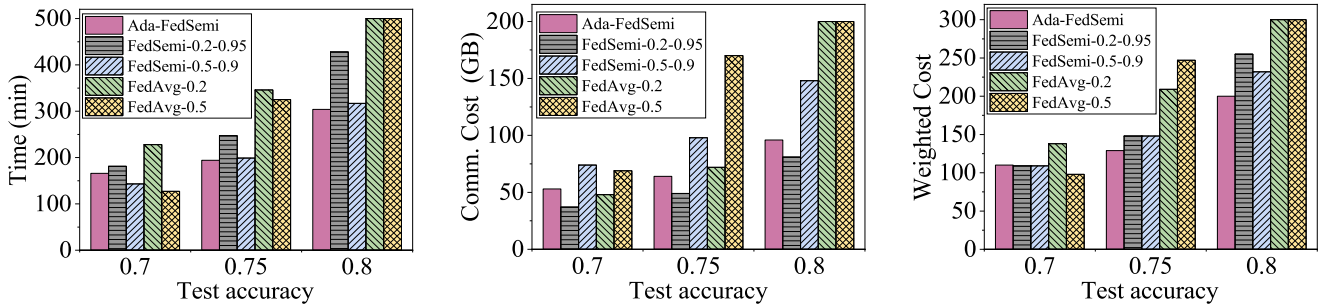
Baselines: We compare our proposed system with the following baselines.

- **FedSemi** [15], [16]: In FedSemi, the in-cloud unlabeled data and on-device labeled data are used to train models in a semi-supervised way. However, the two critical parameters, *i.e.*, P and C, are fixed during the training. Given different combinations of P and C, we denote the baselines as FedSemi-P-C, *e.g.*, FedSemi-0.5-0.9.
- **FedAvg** [4]: In FedAvg, only labeled data on clients are utilized to train models, and thus there is only one critical parameter, *i.e.*, P. We denote the FedAvg with different P as FedAvg-P, *e.g.*, FedAvg-0.2. Note that if the value of C in FedSemi-P-C is set as 1.0 (*i.e.*, FedSemi-P-1.0), FedSemi-P-1.0 is equivalent to FedAvg-P, since the confidence of pseudo-labels cannot exceed 1.0 and none of the unlabeled data is selected. For ease of presentation, we will use FedSemi-P-1.0 and FedAvg-P interchangeably in the later experiments.

Performance Metrics: In the experiments, we employ the following metrics to evaluate the performance of different FL systems: (1) Test accuracy. In each round, we will evaluate the global model on test dataset and record the accuracy. (2) Time cost. We will record the time to achieve the target test accuracy on different FL systems. (3) Communication cost. The communication cost for broadcasting and uploading models is also recorded when achieving the target test accuracy. (4) Weighted Cost. Based on the cost preference and Eq. (10), we combine time cost and communication cost to derive the weighted cost.

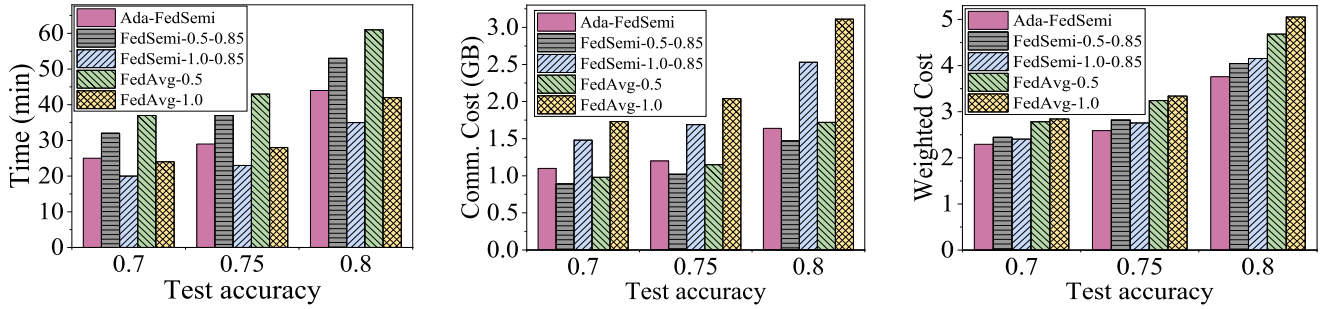
5.3 The Impacts of P and C

We first conduct experiments on the baselines with fixed P and C to analyze the impacts of the two parameters. The time budget of the training is set as 480min, 80min and 480min for CIFAR10, SVHN and STL10, respectively. The corresponding target test accuracy is set as 75%, 75%



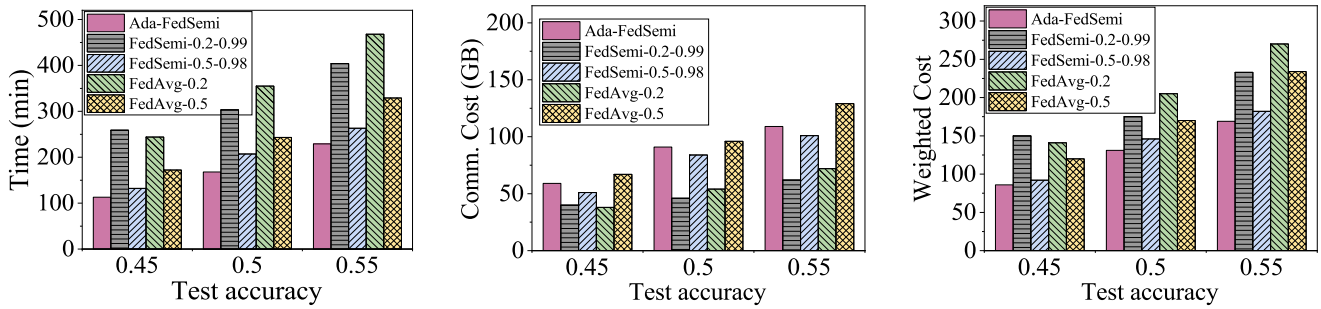
(a) Time cost (b) Communication cost (c) Weighted cost with $\alpha = 0.5$

Fig. 6: Training cost on CIFAR10.



(a) Time cost (b) Communication cost (c) Weighted cost with $\alpha = 0.05$

Fig. 7: Training cost on SVHN.



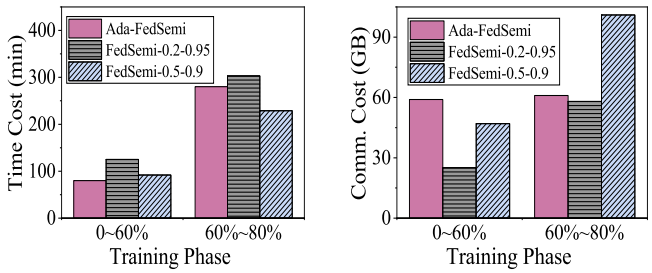
(a) Time cost (b) Communication cost (c) Weighted cost with $\alpha = 0.5$

Fig. 8: Training cost on STL10.

and 50%. For training efficiency, by default, Ada-FedSemi randomly selects 100,000 out of 600,728 (for SVHN) and 40,000 out of 100,00 (for STL10) unlabeled data to generate pseudo-labels. The impact of the scale of unlabeled dataset will be analyzed in Section 5.6. The experimental results are presented in Tables 1-3.

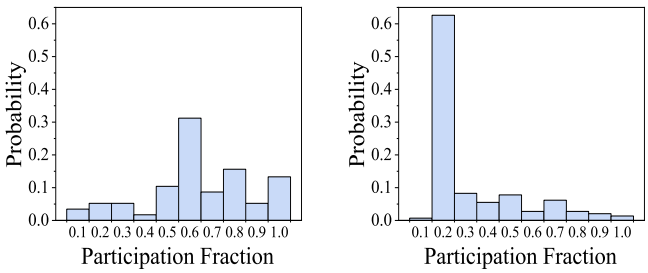
From the perspective of test accuracy, the models trained on both labeled and unlabeled data (*i.e.*, FedSemi) can achieve 1.6%-7.8% accuracy improvement on three datasets, compared with the models only trained on labeled dataset (*i.e.*, FedAvg). However, we observe that FedSemi with small C and/or small P suffers from accuracy degradation (*e.g.*, $P = 0.1$ and $C \leq 0.9$ on STL10) and even fails to converge (*e.g.*, FedSemi-0.1-0.8 on CIFAR10). Given a small P, the models will only learns knowledge from a small number of clients and labeled data, and a small C will bring in many low-confidence pseudo-labels during the training, which degrades the performance of models. Besides, the values of P have influence on the optimal values of C. For example, on CIFAR10, when all clients participate in the FL training (*i.e.*, $P = 1.0$), $C = 0.8$ achieves the highest test accuracy. However, when selecting only 10% of clients (*i.e.*, $P = 0.1$), C needs to be set as 0.98 to achieve the best accuracy. The reason lies in that the models can learn more knowledge from the labeled data with the increasing of P and thus generate pseudo-labels with higher quality. Thus, when using the same C, we can select more samples with less errors from the pseudo-labeled data.

In terms of training cost, the time cost and communication cost are usually contradictory. With the increasing number of participating clients, the time to achieve the target accuracy gets shorter and meanwhile the communication cost gets higher. For example, when training models on SVHN using FedAvg, with the value of P varying from 0.2 to 1.0,



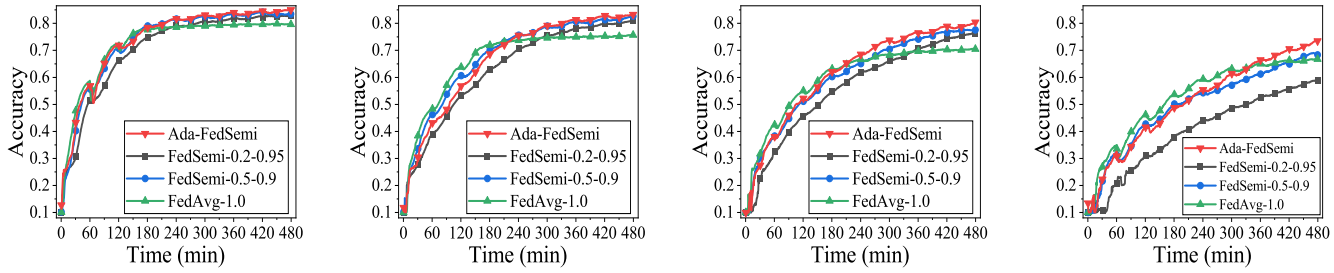
(a) Time cost (b) Communication cost

Fig. 9: Training cost on CIFAR10 with varying preference.



(a) Phase 1: 0∼60% ($\alpha = 0.9$) (b) Phase 2: 60%∼80% ($\alpha = 0.1$)

Fig. 10: Distribution of P in two training phases.



(a) IID (b) Non-IID-0.5 (c) Non-IID-0.6 (d) Non-IID-0.75

Fig. 11: Training process of Ada-FedSemi and the baselines on different data distributions.

the time cost to achieve 75% test accuracy decreases from 58min to 28min while the communication cost increases from 0.76GB to 2.04GB. It is noteworthy that training on both labeled and unlabeled data may not always achieve better training efficiency, compared with training only on labeled dataset. For example, on CIFAR10, FedSemi-1.0-0.6 spends 33% more time cost and 10% more communication cost to achieve the same target test accuracy in comparison to FedAvg-1.0. This is because a large number of low-confidence pseudo-labels mislead the optimization of model training and result in resource waste.

In Tables 1-3, we observe that a little extra time cost can help reduce the communication cost to a great extent in some cases. For example, on CIFAR10, when C is set as 0.9, the system with $P = 0.5$ spends 7% more time but saves 47% communication cost, compared with $P = 1.0$. Different FL tasks always have different cost preferences. Some tasks expect to converge fast without considering communication cost while others may prefer to perform model training in a communication-efficient way. As a result, these preferences will lead to different optimal decision of P and C. We present the optimal combination of P and C under different preferences in Tables 4-6. For example, on STL10, when the task prefers saving communication cost, *i.e.*, $\alpha < 0.284$, the minimum weighted cost can be achieved with $P = 0.2$. When the task is expected to converge fast (*i.e.*, $\alpha \geq 0.529$), all clients (*i.e.*, $P = 1.0$) should participate in the training.

5.4 Performance Comparison

In this section, we compare the performance of Ada-FedSemi and baselines. In Ada-FedSemi, the decision spaces of P and C are set as $[0.0, 1.0]$ and $[0.8, 1.0]$, respectively, and we evenly partition each of the two decision spaces into 10 discrete values. By default, we set the preference parameter α as 0.5, 0.05 and 0.5 for the training on CIFAR10, SVHN and STL10, respectively. As indicated in Table 4, on CIFAR10, $P = 0.2$ and 0.5 can achieve small weighted cost when $\alpha = 0.5$. Therefore, we choose FedAvg-0.2, FedAvg-0.5, FedSemi-0.2-0.95 and FedSemi-0.5-0.9 as baselines. Similarly, on SVHN, we take FedAvg-0.5, FedAvg-1.0, FedSemi-0.5-0.85 and FedSemi-1.0-0.85 for comparison while on STL10, FedAvg-0.2, FedAvg-0.5, FedSemi-0.2-0.99 and FedSemi-0.5-0.98 are adopted as baselines. When a system cannot achieve the target test accuracy, its cost is set as the maximum (*i.e.*, the budgets are exhausted). On CIFAR10 and STL10, the time budget and traffic budget are set as 500min and 200GB, respectively. On SVHN, those are 60min and 3GB.

The time cost, communication cost and weighted cost of Ada-FedSemi and the baselines for achieving the different test accuracy are presented in Figs. 6-8. Although baselines with fixed P and C may achieve the least time cost (*i.e.*, $P = 1.0$ on SVHN) or communication cost (*i.e.*, $P = 0.2$ on CIFAR10) in some cases, they cannot achieve the least weighted cost. On the contrary, Ada-FedSemi can always achieve the least weighted cost, indicating that Ada-FedSemi is able to balance time cost and communication cost given the specific cost preference. On CIFAR10, compared with FedAvg-P, Ada-FedSemi can save 35% ($P = 0.2$) and 48% ($P = 0.5$) weighted cost when achieving 75% test accuracy. Given the 80% test accuracy, Ada-FedSemi saves the weighted cost over FedSemi-0.2-0.95 and FedSemi-0.5-0.9 by 22% and 14%. However, FedAvg-P fails to achieve higher test accuracy (*i.e.*, 80%) without utilization of the in-cloud unlabeled data. On SVHN and STL10, compared with other baselines, Ada-FedSemi saves 6%∼25% and 7%∼43% weighted cost, respectively.

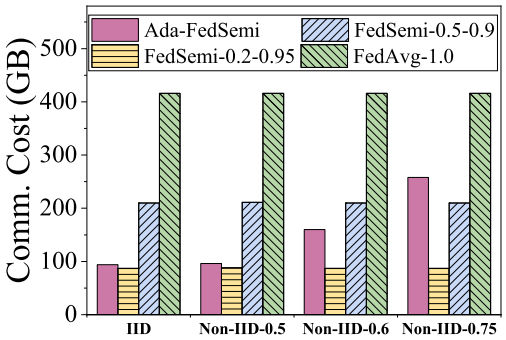
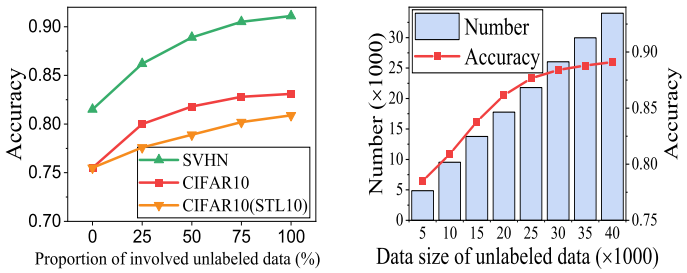


Fig. 12: Communication cost of different systems.



(a) Test accuracy

(b) Number of selected data and accuracy of pseudo-labels on CIFAR10

Fig. 13: Performance of models trained with different scale of unlabeled data.

TABLE 7: Performance comparison of different systems on IMDB dataset.

Metrics	Accuracy (%)	Time (min)	Traffic (GB)
Ada-FedSemi	74.8±0.2	76.8	52.6
FedSemi	73.6±0.3	84.7	59.2
FedAvg	70.7±0.1	104.1	69.4

Furthermore, we evaluate the systems' performance on IMDB dataset to verify that Ada-FedSemi can also perform well on the text classification task. For the baselines FedAvg and FedSemi, we set P as that of Ada-FedSemi in each round while C in FedSemi is set as 0.95. As such, we can compare the training performance between fixed C and adaptive C. In Table 7, we present the best accuracy in 120min as well as the training cost to achieve 70% target accuracy. We observe that Ada-FedSemi reaches the highest test accuracy due to its adaptive determination of C. Meanwhile, when achieving the same target accuracy, Ada-FedSemi provides a speedup of 1.1×∼1.4×, compared with baselines. Besides, compared with FedSemi and FedAvg, Ada-FedSemi reduces the network traffic by 11.2% and 24.2%, respectively. These results show remarkable performance gains of the proposed algorithm, even on the text classification task.

We conduct another set of experiments on CIFAR10 in the scenario where the FL tasks would like to achieve an acceptable test accuracy with low time cost and then further improve model performance in a communication-efficient way. Specifically, we initially set α as 0.9 to ensure fast convergence and when the test accuracy reaches 60%, α is set as 0.1 to put more emphasis on communication cost. The training process is terminated when the test accuracy reaches 80%. The communication cost and time cost of the two training phases are presented in Fig. 9. In the first training phase, Ada-FedSemi achieves the least time cost, resulting in the most communication cost since our goal in this phase is fast convergence. In the second training phase, our system results in similar communication cost with FedSemi-0.2-0.95 and saves about 50% communication cost in comparison to FedSemi-0.5-0.9. We further present the distribution of the values of P in the two training phases. As shown in Fig. 10, Ada-FedSemi always chooses the optimal P (*i.e.*, 0.6 in the first phase and 0.2 in the second phase) with the high probability, which indicates that our system is able to adaptively determine the optimal combination of P and C even when the cost preference is varying over time.

5.5 Adaptability to Data Distribution

In this section, we conduct experiments to evaluate the impact of data distributions. The models are trained on CIFAR10 with four different data distributions, *i.e.*, IID, non-IID-0.5, non-IID-0.6 and non-IID-0.75. We take FedSemi-0.2-0.95, FedSemi-0.5-0.9 and FedAvg-1.0 for comparison. The time budget is set as 480min, and the experimental results are presented in Fig. 11.

Generally, the test accuracy of the models trained on all systems decreases with the increasing skewness of data distribution. The final test accuracy of Ada-FedSemi on IID, non-IID-0.5, non-IID-0.6 and non-IID-0.75 is 85.0%, 83.3%, 80.4%, and 73.5%, respectively. For the same data distribution, Ada-FedSemi can always achieve the highest test accuracy and outperform the three baselines by 1.5% to 14.8%. We find that FedAvg-1.0 always achieves the best test accuracy at the beginning of the training. This is because, in FedAvg, models are optimized without perturbation of errors from low-confidence pseudo-labels and thus converge fast. For FedSemi, as the training progresses, the prediction ability of models is improved, and thus the teacher model generates more and more high-confidence pseudo-labels for the unlabeled data. As a result, the models trained on these high-confidence pseudo-labeled data can achieve higher test accuracy, compared with FedAvg. Nevertheless, in Fig. 11(d), the test accuracy of FedSemi-0.2-0.95 and FedAvg-1.0 is separately 59.0% and 66.7%, indicating that with highly skewed data, a small number of clients may fail to generate high-confidence pseudo-labels, and thus the incorrect pseudo-labels may mislead the model optimization.

In Fig. 12, we present the communication cost of different systems on the four data distributions. Except Ada-FedSemi, the communication cost of other three baselines is almost the same across different data distributions since they always use fixed P and C. As the skewness of training data increases, our system can adaptively increase communication cost to ensure the best model performance. For example, on non-IID-0.75 dataset, although Ada-FedSemi consumes 20% more communication cost than FedSemi-0.5-0.9, it improves

the final test accuracy from 66.7% to 73.5%. The results of the experiments demonstrate that Ada-FedSemi has the ability of adapting to different data distributions.

5.6 Impact of the Scale of Unlabeled Dataset

As mentioned in Section 5.2, the number of unlabeled data in CIFAR10, SVHN and STL10 is 40,000, 600,728 and 100,000, respectively. To explore the impact of the scale of unlabeled dataset, we fix the size of labeled dataset and conduct experiments by changing the proportion of in-cloud unlabeled data involved in the training. In addition to conducting experiments with labeled and unlabeled data from the same dataset, *i.e.*, SVHN and CIFAR10, we also conduct an experiment using labeled data from CIFAR10 and unlabeled data from STL10, denoted as CIFAR10(STL10). The experimental results are shown in Fig. 13(a). When the proportion of involved unlabeled data varies from 0% to 100%, the accuracy of models on CIFAR10 increases from 75.5% to 83.1% and that on SVHN increases from 81.5% to 91.1%. However, when labeled and unlabeled data are obtained from different datasets, the accuracy improvement is only 5.4%, *i.e.*, from 75.5% to 80.9%. This can be attributed to a large number of unlabeled data in STL10 that do not belong to any classes in CIFAR10. Hence, those data cannot contribute significantly towards improving model performance.

Furthermore, we use the trained model with $C = 0.9$ to select pseudo-labeled data of CIFAR10 and the results are presented in Fig. 13(b). With the increasing of scale of unlabeled data, the number of selected data increases from 4,868 to 34,008, and the accuracy of pseudo-labels increases from 78.5% to 89.1%. This set of experiments demonstrates that in Ada-FedSemi, the final test accuracy of trained models is positively correlated to the scale of unlabeled data. Therefore, when the scale of labeled data on clients is small, it is an effective way to collect and exploit large-scale in-cloud unlabeled data to boost the model performance, and it will not incur additional training cost for resource-constrained clients.

5.7 Impact of Learning Rate Restart

Herein, we also conduct experiments to evaluate the impact of learning rate restart. The constant learning rate 0.05, denoted as no-restart, is taken as comparison. We perform model training on CIFAR10 with FedSemi-0.2-0.9 and FedSemi-0.2-0.7, and the corresponding results are presented in Figs. 14(a) and 14(b), respectively. It shows that the test accuracy first degrades at each moment of learning rate restart and then resumes quickly. Although the models trained with the constant learning rate can achieve continuous improvement, it converges earlier and fails to reach higher test accuracy, compared with the models trained with learning rate restart. This is because the constant learning rate may make the models get trapped in local minimum, especially when there exists noise in pseudo-labels. Instead, restarting learning rate helps the models jump out of local minimum and converge to better solutions.

Although learning rate restart can reach higher final test accuracy, the performance of models during the training may vary significantly. In Fig. 15, we compare the test accuracy of teacher model and global model on two data distributions of CIFAR10, *i.e.*, non-IID-0.5 and non-IID-0.75. We find that compared to the global model, the teacher model can achieve more robust performance improvement. The reason lies in that since the teacher model is a moving average of past local models, it can produce pseudo-labels with less errors, and thus exploits unlabeled data better.

TABLE 8: Configurations of 5 computation modes of TX2.

Mode	0	1	2	3	4
Denver 2	2.0Ghz×2	0	1.4Ghz×2	0	2.0Ghz×2
ARM A57	2.0Ghz×4	1.2Ghz×4	1.4Ghz×4	2.0Ghz×4	0.3Ghz×1
GPU	1.30Ghz	0.85Ghz	1.12Ghz	1.12Ghz	1.12Ghz

TABLE 9: Parameter size (MB) and normalized computation load of submodels of VGG9.

Submodel	H^1	H^2	H^3
Parameter size (MB)	0.11	1.00	13.4
Normalized computation load	1.0	2.3	3.1

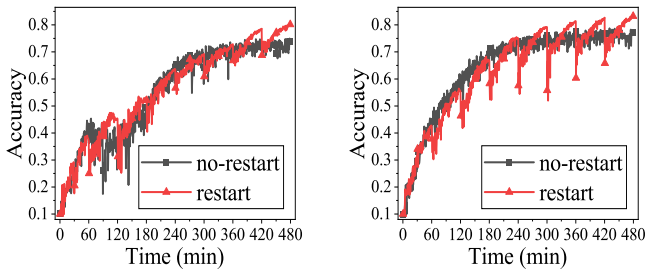
5.8 Extension to Client Heterogeneity Scenario

In this section, we evaluate the performance of our proposed algorithm, *i.e.*, HeteroAda-FedSemi, under the client heterogeneity scenario. The baseline methods are Ada-FedSemi and Semi-HFL [48]. Without considering client heterogeneity, Ada-FedSemi assigns a same model to all selected clients for local training. By utilizing multi-exit models, Semi-HFL heuristically assigns models of different sizes to heterogeneous clients and selects pseudo-labels with a fixed threshold. To make a fair comparison, we modify the implementation of Semi-HFL and apply it in our scenario, where unlabeled data are on the server.

To simulate various clients with heterogeneous computation and communication capabilities, we give the configurations of clients as follows.

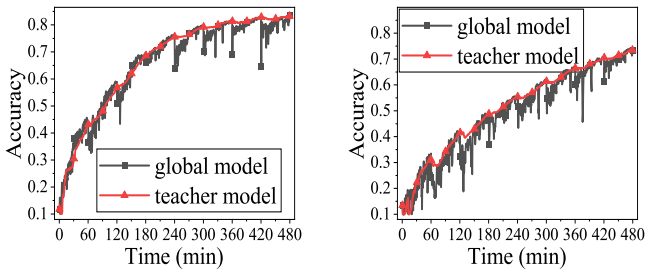
- **For Computation.** TX2 clients can work in one of five computation modes, and the details are listed in Table 8. Specifically, the fastest mode (*i.e.*, mode 0) performs training about 2× faster than the slowest one (*i.e.*, mode 1). To mimic system dynamics, we randomly change the working modes of clients every 20 rounds.
- **For Communication.** All clients are connected to the parameter server via 2.4GHz WiFi. We arrange 20 clients in 3 groups (each of which contains 10, 5 and 5 TX2 devices, respectively). Then, the clients in the groups are placed at different locations that are 2m, 10m and 20m away from the WiFi router. Due to random noises and competition among clients, the bandwidth between the parameter server and clients varies dynamically.

The VGG9 model [58] contains 6 convolutional layers and 3 fully-connected layers. To transform VGG9 into a multi-exit model, auxiliary classifiers are added after the second and the fourth convolutional layers, respectively. As a result, three submodels (including the original model) of different depths can be generated to adapt to the client heterogeneity. The parameter sizes and computation loads of different submodels are presented in Table 9.



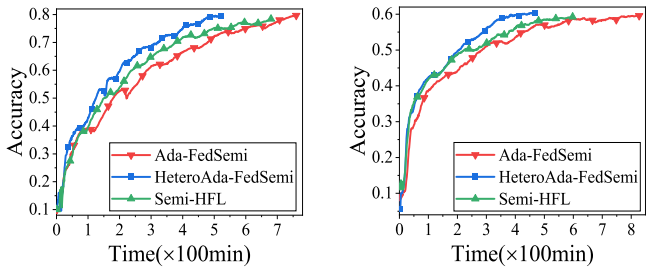
(a) FedSemi-0.2-0.7 (b) FedSemi-0.2-0.9

Fig. 14: Impact of the learning rate restart strategy.



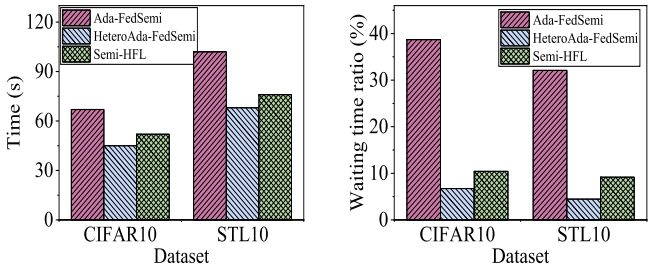
(a) Non-IID-0.5 (b) Non-IID-0.75

Fig. 15: Performance of teacher model and global model.



(a) CIFAR10 (b) STL10

Fig. 16: Training process with heterogeneous clients.



(a) Average round time (b) Waiting time ratio

Fig. 17: Comparison under client heterogeneity.

As shown in Fig. 16, on both CIFAR10 and STL10, HeteroAda-FedSemi always converges fastest. Specifically, compared to Ada-FedSemi and Semi-HFL, HeteroAda-FedSemi speeds up the model training on CIFAR10 by about 1.5× and 1.3×, respectively. The corresponding speedup on STL10 is 1.7× and 1.2×. In Fig. 17(a), we observe that on CIFAR10, the average round time of Ada-FedSemi, HeteroAda-FedSemi and Semi-HFL is 67s, 45s and 52s, respectively. The corresponding time on STL10 is 102s, 68s and 76s. The reason of the performance improvement lies in that considering heterogeneous device capabilities, HeteroAda-FedSemi assigns different submodels for clients to perform local training, thus alleviating the impact of stragglers. Although the clients in Semi-HFL also train different submodels, Semi-HFL can neither assign the most suitable submodel to each client nor determine the confidence threshold adaptively.

To further investigate how long the fast clients wait for the slow ones, we calculate the average waiting time ratio as follows:

$$ratio = \frac{\sum_{k=1}^{K}\sum_{v_m\in\mathcal{V}_k}\frac{t_{k,max}-t_{k,m}}{t_{k,max}}}{\sum_{k=1}^{K}|\mathcal{V}_k|}\times 100\%, \quad (34)$$

where $t_{k,max} = \max_{v_m\in\mathcal{V}_k}\{t_{k,m}\}$. As shown in Fig. 17(b), HeteroAda-FedSemi significantly reduces the waiting time. When training models on CIFAR10, the average waiting time ratio is 38.7%, 6.7% and 10.4% for Ada-FedSemi, HeteroAda-FedSemi and Semi-HFL, respectively. On STL10, it becomes 32.1%, 4.5% and 9.2%, correspondingly. These results demonstrate the efficiency of HeteroAda-FedSemi when training models under the client heterogeneity.

6 Related Works

FL aims to train DL models over networks cooperatively without transferring any users' personal raw data. Since the clients at the network edge are usually resource-constrained (*e.g.*, limited computation and communication capabilities), many existing works focus on training efficiency in FL. However, few works consider the scarcity of labeled data on clients, which is a more practical scenario for federated model training. In what follows, we review the existing works about FL with both labeled and unlabeled data, as well as resource-efficient FL, respectively.

6.1 Federated Learning with Unlabeled Data

The significant improvement of DL in recent years is largely attributed to the utilization of large scale labeled dataset, *e.g.*, Imagenet [60]. However, obtaining labels of data is often very costly and time-consuming in practice [5]. Thus, more and more works are paying attention to exploiting massive unlabeled data [15], [16], [61], which are easy to collect from public access. For example, with the advent of social media, over a billion of people are generating different types of data, including text, images and videos on the Internet continuously [8], which are stored in cloud and can be accessed publicly. One large scale dataset is WebVision Database [9], consisting of 2.4 million web images crawled from the Internet but their labels are always missing. Therefore, semi-supervised learning (SSL) [62] is proposed to train models on both small scale of labeled dataset and large scale of unlabeled dataset.

There are two main methods in traditional SSL. The first one is consistency regularization based algorithms [63]. These algorithms require that the predictions of unlabeled data are invariant to different data augmentations, which is implemented by a consistency regularization term in the loss function. However, as indicated in [21], training

models on the different views of the data (*i.e.*, multiple augmentations of a single data sample) significantly increases the training overhead. The other method is pseudo-labeling based algorithms [17], [21], which regard the high-confidence predictions of unlabeled data as their pseudo-labels. Then, models are trained over both labeled and pseudo-labeled data samples. However, all above methods only care about the final test accuracy of the trained model but don't take the features (*e.g.*, limited capacity of communication and computation) of FL into consideration and thus fail to achieve training efficiency.

Recently, several works try to perform SSL under FL settings. Some works try to exploit on-device unlabeled data. For example, Jeong *et al.* [11] propose to select other clients' local models for each client to help exploit local unlabeled data. Besides, Long *et al.* [12] adopt two models (*i.e.*, teacher and student models) at each client to train models on both labeled and unlabeled data. Moreover, considering the limited scale of on-device unlabeled data, works [13], [14] first distribute in-cloud public unlabeled data to clients and then perform SSL algorithms. However, all above methods try to utilize unlabeled data at the edge, which will increase training cost for clients. Since end devices at the network edge are always resource-constrained and there are large-scale in-cloud public unlabeled data, it is a more efficient way to utilize these data at the PS, without incurring any additional training cost on clients.

6.2 Resource-efficient Federated Learning

Training the DL models (especially the highly over-parameterized models) in a distributed manner usually requires intensive computation and significant communication overhead. In order to achieve efficient training of FL, many algorithms are proposed to reduce time cost and communication cost. Some recent works [19], [23] aim to optimize the training time by utilizing deep reinforcement learning based algorithms to schedule clients. Huang *et al.* [64] aim to accelerate FL while ensuring long-term fairness constraints. However, these works all employ a fixed participating fraction of clients and mainly concentrate on optimizing a single objective (*e.g.*, training time), which cannot satisfy the various cost preferences (*e.g.*, fast convergence or low communication cost) for different FL tasks. As stated in the work [18], a large participating fraction can lead to reduction of training time while a small fraction contributes to saving communication cost. However, the work [18] is designed to determine the offline optimal participating fraction before performing the FL tasks, which fails to adapt to the dynamic changes of cost preferences online.

Moreover, to reduce the volume of transmitted data and accelerate the distributed model training, many prior works [65] propose various model (or gradient) compression techniques. For example, *Quantization* saves communication cost using less bits to represent the original parameter elements (*e.g.*, from float32 to float16). *Sparsification* is also a popular compression method, which transmits a sparse vector including only a subset of the original model parameters. The transmitted parameters are selected randomly or through their magnitude. Another common technique for reducing the number of communication rounds is federated average (FedAvg) [4], which allows clients to perform multiple local updating before global aggregation. Wang *et al.* [25] propose to determine the optimal local updating steps adaptively with the constraint of available resource.

Edge devices usually exhibit capability heterogeneity, due to different types of hardwares, including CPU, GPU, storage and battery life. The clients with poor capabilities may become the system stragglers, slowing down the training process. To deal with the heterogeneity among edge devices in FL, there are a number of works that adopt different methods. Ma *et al.* [52] adjust batch size for heterogeneous clients to reduce the waiting time. Xu *et al.* [66] jointly optimize the local updating frequency and model compression ratio to speed up the training process.

Note that our work is orthogonal to above resource-efficient techniques, which can be adopted in our FL system to further reduce traffic consumption and speed up the training process.

7 Conclusions

To fully utilize the on-device labeled and in-cloud unlabeled data in FL, we propose an adaptive FSSL system called Ada-FedSemi. It employs an MAB based online learning algorithm to adaptively determine the fraction of participating clients and confidence threshold for pseudo-labeling during the federated model training. The dynamic optimization of participating fraction and confidence threshold can contribute to the trade-off between model accuracy and training efficiency given the limited resource budgets. The extensive experimental results demonstrate that Ada-FedSemi significantly outperforms the existing baselines, including FedAvg and FedSemi. Moreover, considering the client heterogeneity, the proposed HeteroAda-FedSemi can further speed up the model training.

References

[1] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A review of machine learning and iot in smart transportation," *Future Internet*, vol. 11, no. 4, p. 94, 2019.

[2] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[3] A. S. Tulshan and S. N. Dhage, "Survey on virtual assistant: Google assistant, siri, cortana, alexa," in *International symposium on signal processing and intelligent recognition systems*. Springer, 2018, pp. 190–201.

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

[5] K. Li, G. Li, Y. Wang, Y. Huang, Z. Liu, and Z. Wu, "Crowdrl: An end-to-end reinforcement learning framework for data labelling," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 289–300.

[6] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," *arXiv preprint arXiv:1706.08947*, 2017.

[7] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big data and cloud computing: innovation opportunities and challenges," *International Journal of Digital Earth*, vol. 10, no. 1, pp. 13–53, 2017.

[8] N. A. Ghani, S. Hamid, I. A. T. Hashem, and E. Ahmed, "Social media big data analytics: A survey," *Computers in Human Behavior*, vol. 101, pp. 417–428, 2019.

[9] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[10] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, "Curriculumnet: Weakly supervised learning from large-scale web images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 135–150.

[11] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," in *International Conference on Learning Representations*, 2021.

[12] Z. Long, L. Che, Y. Wang, M. Ye, J. Luo, J. Wu, H. Xiao, and F. Ma, "Fedsemi: An adaptive federated semi-supervised learning framework," *arXiv preprint arXiv:2012.03292*, 2020.

[13] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[14] I. Bistritz, A. Mann, and N. Bambos, "Distributed distillation for on-device learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[15] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *International Conference on Learning Representations*, 2017.

[16] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," in *International Conference on Learning Representations*, 2018.

[17] P. Cascante-Bonilla, F. Tan, Y. Qi, and V. Ordonez, "Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[18] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021.

[19] W. Xia, T. Q. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7108–7123, 2020.

[20] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

[21] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah, "In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning," in *International Conference on Learning Representations*, 2021.

[22] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2020.

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

[24] M. Tang and V. W. Wong, "An incentive mechanism for cross-silo federated learning: A public goods perspective," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[25] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[26] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 596–608.

[27] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[28] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 300–310.

[29] J. Gareth, W. Daniela, H. Trevor, and T. Robert, *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[30] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[31] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[32] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[33] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations*, 2017.

[34] A. Gotmare, N. S. Keskar, C. Xiong, and R. Socher, "A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation," in *International Conference on Learning Representations*, 2019.

[35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[36] O. Besbes, Y. Gur, and A. Zeevi, "Stochastic multi-armed-bandit problem with non-stationary rewards," *Advances in neural information processing systems*, vol. 27, pp. 199–207, 2014.

[37] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 2011.

[38] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[39] "Cisco (2020) cisco annual internet report (2018-2023). white paper." https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[40] S. P. Singh and M. Jaggi, "Model fusion via optimal transport," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 045–22 055, 2020.

[41] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, 2020.

[42] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[45] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai *et al.*, "Federated learning for predicting clinical outcomes in patients with covid-19," *Nature medicine*, vol. 27, no. 10, pp. 1735–1743, 2021.

[46] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6341–6345.

[47] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Decoupled greedy learning of cnns," in *International Conference on Machine Learning*. PMLR, 2020, pp. 736–745.

[48] Z. Zhong, J. Wang, W. Bao, J. Zhou, X. Zhu, and X. Zhang, "Semi-hfl: semi-supervised federated learning for heterogeneous devices," *Complex & Intelligent Systems*, pp. 1–23, 2022.

[49] A. Mohtashami, M. Jaggi, and S. Stich, "Masked training of neural networks with partial gradients," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 5876–5890.

[50] H.-P. Wang, S. Stich, Y. He, and M. Fritz, "Progfed: effective, communication, and computation efficient federated learning by progressive training," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 034–23 054.

[51] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2020.

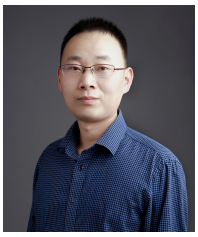
[52] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 37–53, 2023.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[54] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[55] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.
[56] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.
[57] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.
[58] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *International Conference on Learning Representations*, 2020.
[59] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.
[60] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[61] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013, p. 896.
[62] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
[63] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
[64] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Transactions on Parallel and Distributed Systems*, 2020.
[65] Z. Tang, S. Shi, and X. Chu, "Communication-efficient decentralized learning with sparsification and adaptive peer selection," *arXiv preprint arXiv:2002.09692*, 2020.
[66] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Transactions on Mobile Computing*, 2022.

Yang Xu (Member, IEEE) is currently an associate researcher in the School of Computer Science and Technology at University of Science and Technology of China. He got his Ph.D. degree in computer science and technology from University of Science and Technology of China in 2019. He got his B.S. degree in Wuhan University of Technology in 2014. His research interests include Ubiquitous Computing, Deep Learning and Mobile Edge Computing.

Lun Wang received the B.S. degree from the University of Electronic Science and Technology of China in 2019. He is currently pursuing his Ph.D. degree in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include mobile edge computing, federated learning and distributed machine learning.



Hongli Xu (Member, IEEE) received the B.S. degree in computer science from the University of Science and Technology of China, China, in 2002, and the Ph. D degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China, China. He was awarded the Outstanding Youth Science Foundation of NSFC, in 2018. He has won the best paper award or the best paper candidate in several famous conferences. He has published more than 100 papers in famous journals and conferences, including the IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interest is software defined networks, edge computing and Internet of Thing.

Jianchun Liu received the Ph.D. degree in School of Data Science from the University of Science and Technology of China in 2022. He is currently an associate researcher in the School of Computer Science and Technology at University of Science and Technology of China. His main research interests are software defined networks, network function virtualization, edge computing and federated learning.

Zhiyuan Wang received the B.S. degree from the Jilin University in 2019. He is currently a Ph.D. candidate in the School of Computer Science, University of Science and Technology of China (USTC). His main research interests are edge computing, federated learning and distributed machine learning.

Liusheng Huang (Member, IEEE) received his M.S. degree in computer science from the University of Science and Technology of China in 1988. He is currently a Senior Professor and a Ph.D. Supervisor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or co-authored six books and over 300 journal/conference papers. His research interests are in the areas of the Internet of Things, vehicular Ad-Hoc networks, information security, and distributed computing.