Benchmarking Semi-supervised Federated Learning

Zhengming Zhang[1*], Zhewei Yao[2*], Yaoqing Yang[2], Yujun Yan[3],
Joseph E. Gonzalez[2], Michael W. Mahoney[2]
[1] Southeast University,
[2] University of California, Berkeley,
[3] University of Michigan, Ann Arbor

Abstract

Federated learning is an effective way to use the computational power of edge devices and maintain user data privacy. Although this approach is promising, current frameworks often make the impractical assumption that the data stored in the user devices have ground truth labels. In this work, we are the first to consider a more realistic scenario, where only the server has a limited amount of labeled data, while users have only unlabeled data. In this semi-supervised federated learning (SSFL) setting, the data distribution can be non-iid, and a portion of the users are allowed not to communicate with the server at every update. Due to the lack of consistent definitions of non-iidness in prior work in the area, we define a metric to measure non-iidness within this SSFL framework. To *benchmark our* SSFL *framework*, the following factors are considered: the non-iidness R, the communication period T, the number of users K, the amount of labeled data in the server N_s, and the number of users $C_k \leq K$ that communicate with the server in each communication round. We evaluate our SSFL framework on Cifar-10, SVHN, and EMNIST. We find that a simple consistency loss-based method, along with group normalization, can already achieve better generalization performance, even compared to previous supervised federated learning setting. Furthermore, a novel grouping-based model average is proposed to improve convergence efficiency. Our results show that our novel averaging method can boost the performance by up to 10.79% on EMNIST compared to the non-grouping based method. We have open-sourced the SSFL framework[1].

1 Introduction

Current state-of-the-art machine learning models often require high training costs as well as a huge amount of labeled data [1, 2]. Federated learning (FL), which uses smart mobile devices and local user data for training, has generated increasing interest due to the extra available computing power and the locally-available data. Much of the data stored in mobile devices can only be accessed locally, e.g., to preserve users' privacy. Thus, FL does not require users to upload their local data to the server, but it only requires the communication of the trained (intermediate) models. This can still allow the server to discover the necessary information hidden in the user data [3, 4].

Generally, for a typical FL pipeline, each user/device first updates a global shared model for multiple steps locally and then uploads the updated model to the server. After aggregating all the models from users, the server takes an averaging step over all the models (e.g., FedAvg [4]), and then it sends the averaged model back to users [3, 5]. This process preserves privacy since the server does not access the private user data at any point. However, one critical problem here is that prior work often makes the unrealistic assumption that the data stored on the local device are fully annotated with ground-truth labels. In fact, the private data at the local device are often unlabeled, since annotating data requires both time and domain knowledge [6, 7].

*equal contribution
[1] https://github.com/jhcknzzm/SSFL-Benchmarking-Semi-supervised-Federated-Learning

In this paper, we study a novel and more realistic federated learning setting, the *semi-supervised federated learning framework* (SSFL), where the users only have access to unlabeled data, and the server only has a small amount of labeled data. Our main contributions are the following.

1. We introduce the SSFL framework, and we propose a principled way to study non-iid data distributions. Although non-iidness has been explored in many previous works [8, 9, 10], a consistent definition is still lacking. Here, we introduce a traditional probability distance measurement (total distance, denoted as R) to evaluate the non-iidness.
2. We conduct experiments on the effects of different normalization (batch normalization (BN) [11] versus group normalization (GN) [12]) and the importance of loss choice (traditional self-training loss versus consistency regularization loss (CRL) [13, 14]). Our result shows that GN combined with CRL leads to the best generalization performance under SSFL setting.
3. With the best solution (GN with CRL), we extensively evaluate our SSFL framework concerning: different non-iidness R; the number of users K; the communication period T; the number of labeled data points in the server N_s; and the number of users $C_k \leq K$ that communicate with the server in each communication round. In addition to establishing state-of-the-art results, we also benchmark the more realistic SSFL setting, as compared to previous work.
4. We propose a novel grouping-based model averaging method to resolve the problem when the number of users is large. Particularly, we divide users into different sub-groups, and we use different FedAvg [4] models for different sub-groups. This grouping-based method outperforms FedAvg by 0.55%/0.16%/10.79% on Cifar-10/SVHN/EMNIST, respectively.
5. We compare our proposed method with other supervised FL algorithms and supervised distributed training algorithms. Even if our communication frequency is lower than the supervised distributed training algorithms and/or the degree of our non-iidness is higher than other supervised FL, our approach still achieves superb results.

Our approaches can also be applied to the (more common) supervised federated learning setting. However, a detailed exploration of this is outside the scope of this paper, and it is likely of less interest than the (more realistic) setting we consider.

2 Related work

Federated Learning. Federated learning (FL) [3, 4, 8, 15, 9, 10, 16, 17, 7, 17, 18] is a decentralized computing framework that enables multiple users to learn a shared model while potentially protecting privacy of users (although recent work [19] shows this may not be the case). Federated Averaging (FedAvg) [4], which is the most popular FL algorithm, shows good performance when the data distribution across users is iid. However, in the non-iid case, the performance can significantly degrade. In fact, dealing with non-iid distributions is deemed by many to be one of the most critical challenges in FL [9, 10, 8]. In [9], a data-sharing method is proposed to improve the final accuracy. However, sharing massive data among all users requires both large storage space, as well as stable connections between users and the server. Note that all of these methods require the data stored by the local users to come with ground-truth labels (in order to perform model updates locally).

In addition to the challenge of the non-iidness of the data distribution and the need for local ground truth labels, communication efficiency is another critical problem in FL [20, 16, 21, 22, 23]. One way to relieve the communication burden of FL is to increase the period (the number of local gradient descent iterations) between consecutive communication stages. However, when this communication period increases, the diversity between different models increases, and the fusion of these models by the server may lead to accuracy degradation. To handle this problem, [21] proposes FedProx, which adds a proximal term in the user local loss function to restrict the update distance between the local model and the global model. Other work considers gradient compression and model compression to reduce the communication cost [16, 18, 23]. For example, [16] proposes atomic sparsification of stochastic gradients, which leads to significantly faster distributed training.

In the SSFL setting, [7, 17] are most relevant to our work. However, unlike [7], the data at the users in

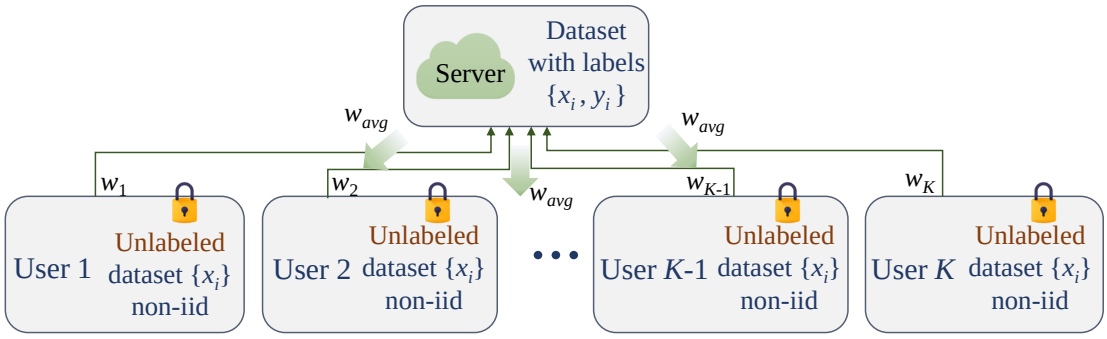

Figure 1: *Semi-supervised federated learning (*SSFL*). Only the server has access to labeled data, i.e., the data stored in local users are unlabeled. Furthermore, the data distributions across different users are non-iid.*

SSFL are fully unlabeled, and one cannot use a supervised learning technique to train the users' data; and unlike [17], we consider the non-iid case with unlabeled user data, and we study the effect of the changing number of communicating users, which is more realistic.

Semi-supervised Learning. Semi-supervised learning (SSL) is a classical problem when only a small fraction of data is labeled [24, 6, 25, 26, 27, 28, 14, 29]. In [29], a self-training method is introduced, which improves the state-of-the-art accuracy on ImageNet [30], even compared to supervised learning [31, 32]. In [14], a consistency-based loss is proposed to improve the performance of semi-supervised learning further.

In this work, we study SSL in the federated setting. The main challenges, compared to standard SSL, are two-fold: (i) the non-iid data distribution; and (ii) the communication constraints. In particular, for the non-iid data distribution: all of the labeled data are only stored in the server, as opposed to classical SSL setting where labeled data are shared with all computational nodes; and the data stored in different local users have an unbalanced class distribution.

3 A Framework of Semi-supervised Federated Learning

In our SSFL framework, there exists a cloud server and K users/devices. We denote the labeled dataset at the server by $D_s = \{(x_i, y_i)\}_{i=1}^{N_s}$, and the unlabeled dataset stored at the k-th user by $D_k = \{x_i\}_{i=1}^{N_k}$. Here, N_s (N_k) is the number of labeled (unlabeled) samples available at the server (k-th user). Note that no data is exchanging between the server and the users. That is to say, the server can only train on dataset D_s, and the k-th user can only train the local model using the local dataset D_k. In this work, we consider image classification as a representative SSFL task. Our SSFL framework is illustrated in Fig. 1.

3.1 A metric on the level of non-iidness

Although non-iid distributions in FL have been explored [9, 8], a consistent definition to quantitatively measure non-iidness is lacking. Here, we use the average total variation distance [33] to define a non-iid metric in the label skew case (a user has more data for one (couple) class(es) than others).

Definition 1 (Metric R for non-iid level)**.** *Denote the probabilistic mass function of the data D_k at the k-th user by $P_k \in \mathbb{R}^d$, where d is the number of classes.*[2] *Then, the non-iid metric R is defined as:*

$$R = \frac{2}{K(K-1)} \sum_{1 \le k < m \le K} \|P_k - P_m\|_1 / 2, \tag{1}$$

where $\|\cdot\|_1$ is the L_1 norm.

[2]Here, $\sum_{j=1}^{d} P_k[j] = 1$ for all $1 \le k \le K$.

It is not hard to see that $0 \leq R \leq 1$ [33]. When data are distributed uniformly at each user, we have $P_k = [1/d, ..., 1/d]$, for all k, and $R = 0$; and in another extreme, when $K = d$ and each user only has access to one class, we have $R = 1$. (In § 4.1, we study the impact of different non-iid levels.)

3.2 Our framework of semi-supervised federated learning

Similar to the traditional FL setting [4, 3], in our SSFL framework, at each communication round, both the server and each local user perform the training process on their own datasets.[3] In particular, we denote the loss function used by the server (resp., k-th user) as L_s (resp., L_k). Inspired by the traditional self-training method, we can define L_s and L_k as follows [34, 29]:

$$L_s = \frac{1}{N_s} \sum\nolimits_{(x_i, y_i) \in D_s} l\left(y_i, f_s(\alpha(x_i); w_s)\right), \tag{2}$$

$$L_k = \frac{1}{N_k} \sum\nolimits_{x_i \in D_k} \mathbf{1}\left(\max(\bar{y}_i) \geq \tau\right) l\left(\arg\max(\bar{y}_i), f_k(\alpha(x_i); w_k)\right), \tag{3}$$

where w_s (w_k) is the weights of server model f_s (k-th user model f_k), $l(\cdot, \cdot)$ is the cross-entropy loss, $\alpha(\cdot)$ is the data augmentation function (for traditional self-training, it is flip-and-shift augmentation), $\bar{y}_i$ is the prediction of the model f_k on $\alpha(x_i)$, $\mathbf{1}(\cdot)$ is the indicator function, and τ is the threshold hyperparameter which helps the model to decide which samples have high confidence to be trained. A simple and popular method to optimize Eq. 2 and 3 is SGD with momentum over a mini-batch.

After multi-step SGD updates, local users will send their models to the server. Then, the server will compute an averaged model using all the received models (including its own)[4]

$$w_{avg} \overset{FedAvg}{=} \left(w_s + \sum\nolimits_{k=1}^{K} w_k\right) / (K + 1). \tag{4}$$

After this, the server will broadcast this w_{avg} to all the users for next round of training.

In this work, we extensively explore the different factors affecting the learning process. In particular, the following factors are considered,

1. Non-iidness R: the non-iid metric of data distributions; see Definition 1.
2. Communication period T: during two consecutive communications, the number of training steps for local users and the server.
3. User number K: the total number of users.
4. Server data number N_s: the number of labeled data in the server (since the entire dataset is fixed, the remaining data will be distributed among the client devices).
5. Number of participating clients C_k: at each communication, the number of clients who send their models to the server. When $C_k < K$, the averaged model is computed by:

$$w_{avg}^{C_k} = \left(w_s + \sum\nolimits_{k \in U_k} w_k\right) / (C_k + 1), \tag{5}$$

where U_k is a user set with C_k users.

3.3 Main factors affecting the performance of semi-supervised federated learning

Training loss. In [14], the authors use two different types of data augmentations, which are the standard flip-and-shift augmentation $\alpha(\cdot)$ and the RandAugment [35] $A(\cdot)$. Here, the latter RadAugment uses two different augmentation methods (i.e., shift and crop) out of twelve possible augmentation methods (e.g., rotate, shift and solarize) for one image. We refer the interested reader to [35] for a detailed explanation.

[3] Note that, in standard FL, the server does not involve any training [4, 17].

[4] This is different from traditional FL since traditional FL does not have a solely trained server model.

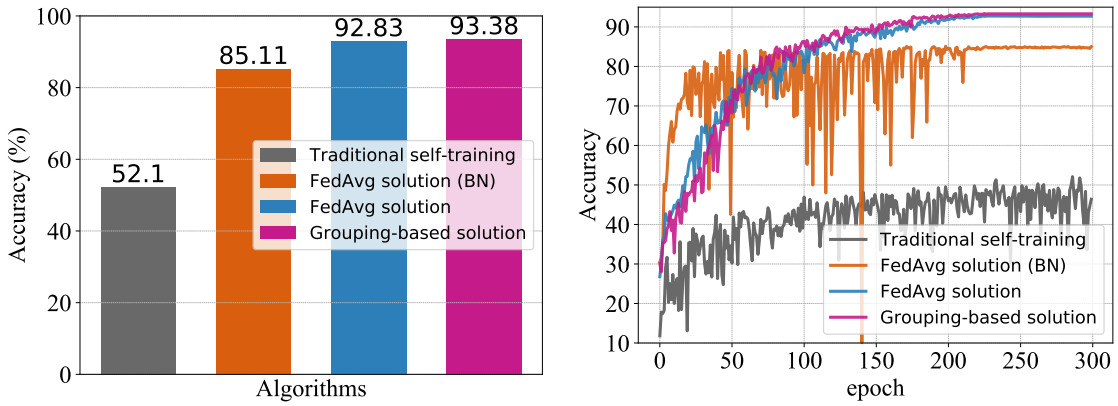

Figure 2: *(Left) Accuracy of different methods on Cifar-10. (Right) The corresponding convergence behaviors.*

Inspired by [14], we also use these two data augmentation methods $\alpha(\cdot)$ and $A(\cdot)$ to train the local user's model,

$$L_k = \frac{1}{N_k} \sum\nolimits_{x_i \in D_k} \mathbf{1}\left(\max(\bar{y}_i) \geq \tau\right) l\left(\arg\max(\bar{y}_i), f_k(A(x_i); w_k)\right), \tag{6}$$

where $\bar{y}_i = f_k(\alpha(x_i); w_k)$. In [14, 13], the authors argued that the predictions of the same image with two data augmentations should be consistent with each other, and they refer to Eq. 6 as the consistency regularization loss (CRL).

This CRL has been proved effective in the semi-supervised setting [14], but its impact in SSFL is unexplored.[5]

Normalization. Since the data distributions across users are non-iid in SSFL, normalization methods (e.g., BN) using statistical properties of data distributions, will have statistical discrepancies between different users. This leads the average model to obtain worse generalization performance. Recent works [36, 37] found out that in supervised FL with non-idd data distributions, the performance of group normalization (GN) is usually much better than that of batch normalization (BN). Here, we also study these two different normalization methods (BN versus GN) in our SSFL setting.

Model Averaging. As show in § 4.3, if we use naive FedAvg, given in Eq. 4, the large model diversity across different users significantly slows down the training process. To overcome this, we propose a novel grouping-based model averaging, which divides C_k communication users into S groups and does the average group-wise. Specifically, after collecting all C_k communication users, the server divides them into S groups $\{G_i\}_{i=1}^{S}$, and updates the w_{avg} according to:

$$\begin{cases} w_{avg,i} = \left(w_s + \sum_{w_k \in G_i} w_k\right) / (|G_i| + 1), & i \in \{1, 2, \ldots, S\} \\ w_{avg} = \sum_{i=1}^{S} w_{avg,i} / S. \end{cases} \tag{7}$$

After this, $w_{avg,i}$ is broadcast to the user group G_i, and w_{avg} is the weights used for server training.

Experimental comparison of different factors. We use ResNet-18 [31] on Cifar-10 to study the effectiveness of different choices mentioned above. Here, for the rest factors mentioned in § 3.2, we set $T = 16$, $K = 10$, $C_k = 10$, $R = 0.4$, and $N_s = 1000$. The threshold τ used in Eq. 2 and Eq. 6 is chosen to be 0.95, as in [14]. The number of groups S in Eq. 7 is set as 2. In Fig. 2, we compare the results of different approaches. In particular, we abbreviate:

1. "Traditional self-training": we use traditional self-training loss Eq. 3, BN, and FedAvg, Eq. 4.
2. "FedAvg solution (BN)": we use Eq. 6, BN, and FedAvg, Eq. 4.
3. "FedAvg solution": we use Eq. 6, GN, and FedAvg, Eq. 4.
4. "Grouping-based solution": we use Eq. 6, GN, and grouping-based model averaging, Eq. 7.

[5] In the semi-supervised setting: (i) both labeled and unlabeled are used for each training iteration; (ii) there is no non-iidness; and (iii) there is usually no user-server model communication constraint.

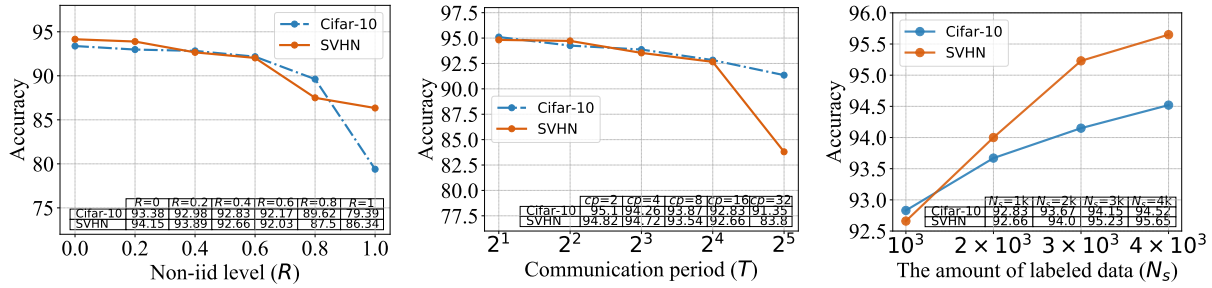

Figure 3: *(Left) Comparison between different non-iid levels (R) on Cifar-10 and SVHN. (Middle) Accuracy versus communication period T. (Right) Accuracy versus labeled data points in the server (N_s).*

As one can see, without any modification of the model/loss/averaging method, the accuracy of traditional self-training is only 52.10%. Replacing the user loss from Eq. 3 to Eq. 6, the performance boosts to 85.11%. Further correcting the normalization method from BN to GN helps the accuracy increase to 92.83%. The best performance is achieved by our grouping-based model averaging, which has 0.55% better accuracy than the FedAvg solution. The convergence behaviors of different methods are also shown in the right of Fig. 2.

Later on, without specification, the results reported in all of our experiments use the FedAvg solution setting. (A detailed comparison between FedAvg and group-based solution is discussed in § 4.3.)

4 Results

We consider three datasets, Cifar-10, SVHN, and EMNIST in our experiments. We use ResNet-18 [31] as the training model on both Cifar-10 and SVHN datasets. For EMNIST, we use the "balanced" training dataset. That is to say, we truncated the training dataset to have 2400 data points per class and drop the rest. For the test dataset, we keep the original one. We use the same CNN model as [36] on EMNIST. See Appendix A for more details.

4.1 Impact of non-iidness R, communication period T, and labeled data number N_s

We conduct experiments on Cifar-10 and SVHN to illustrate the effect of the non-iid level R. Particularly, for Cifar-10/SVHN, we set user number $K = 10/10$, the amount of labeled data $N_s = 1000/1000$, the number of communicate users $C_k = 10/10$, and the communication period $T = 16/16$, respectively. The results are shown in the left of Fig. 3. When $R = 0$ ($R = 1$), the user will have a uniform data distribution across different classes (single class data). As can be seen, the accuracy decreases as the non-iid level R increases (from 93.38%/94.15% to 79.39%/86.34% on Cifar-10/SVHN). This is in accord with our intuition that iid data distribution typically leads to the best result.

We also illustrate the effect of the communication period T on Cifar-10 and SVHN. Particularly, we set $R = 0.4/0.4$, $K = 10/10$, $N_s = 1000/1000$ and $C_k = 10/10$ for Cifar-10 and SVHN, respectively. The middle of Fig. 3 presents our results. Increasing T (i.e., communicate less frequently) leads to a worse generalization performance. This can be explained since the local model can be overfitted when T is large. In addition, the converges behaviors on Cifar-10 at different T can be found in Fig. B.1.

To investigate the impact of the amount of labeled data in the server, we design our experiment on Cifar-10 and SVHN. Particularly, we use $T = 16/16$, $R = 0.4/0.4$, $K = 10/10$ and $C_k = 10/10$ in all experiments. Our result is shown in the right of Fig. 3. Note that increasing the amount of labeled data in the server can significantly improve final generalization performance. For example, with 4k labeled data, the accuracy on Cifar-10/SVHN is 1.69%/2.99% higher as compared to 1000 labeled data. This is intuitive since the increase of the amount of labeled data can make the model trained by the server more accurate, which helps the users to obtain more accurate pseudo-labels and to improve performance. In the extreme case, if the server has the entire labeled training dataset, the situation degrades to a centralized supervised learning setting.

4.2 Impact of the number of communicated users C_k

In the real scenario, the number of connected users varies at different time due, e.g., to the internet quality or privacy concerns. To simulate this drop-and-reconnected case, for every communication, we here assume that only $C_k \leq K$ users out of the entire K users are connecting to the server. Namely, for every communication, only C_k users will upload their local trained model into the server. This setting also has another advantage, i.e., that the communication volume remains the same as we increase the user number K, while holding C_k fixed.

Table 1: *Accuracy versus amount of communicated user C_k on Cifar-10 and SVHN. Here, "*" here means we train SVHN for $E = 120$ epochs instead of $E = 40$ epochs for normal SVHN training.*

	$K = 10$, $C_k = 10$	$K = 20$, $C_k = 20$	$K = 30$, $C_k = 30$
Cifar-10	92.83%	92.13%	91.87%
SVHN	94.36%	94.83%	74.71% (93.94%*)
	$K = 10$, $C_k = 10$	$K = 20$, $C_k = 10$	$K = 30$, $C_k = 10$
Cifar-10	92.83%	93.06%	92.78%
SVHN	94.36%	94.78%	93.37%

Table 2: *The results of FedAvg setting versus grouping-based setting on Cifar-10, SVHN and EMNIST.*

Dataset	FedAvg	Grouping-based
Cifar-10	92.83%	**93.38**%
SVHN	94.83%	**94.99**%
EMNIST	70.84%	**81.63**%

We set $T = 16/16/16$, $R = 0.4/0.4/0.4$, and $N_s = 1000/1000/4700$ for the three datasets (Cifar-10, SVHN, and EMNIST). The results of our FedAvg method on Cifar-10 and SVHN are shown in Tab. 1, and the result of EMNIST is presented in Tab. C.1.

On the top of Tab. 1, we set $C_k = K$. As can be seen, increasing the number of users has a marginal effect (<1%) on the final performance, from $K = 10$ to $K = 30$. One notable thing here is that with $K = 30$, if we use the standard training epochs (40 epochs) on SVHN, it only has 74.71% accuracy, which is 19.65% lower than $K = 10$. With a larger number of users, the diversity of the models across different users increases significantly. This will add noise to the federated averaging and will make the convergence slow (discussed below). The training curve (as shown in Fig. D.1) verifies our conjecture. Therefore, we increase the training epochs from 40 to 120 for $K = 30$ on SVHN, and the final accuracy is 93.94%.

The bottom of Tab. 1 shows the result with fixed $C_k = 10$ and various K (from 10 to 30). Counterintuitively, the results have consistently better performance, compared to $C_k = K$. Particularly, the $K = 30$, $C_k = 10$ case outperforms $C_k = 30$ by 0.91%/18.66% on Cifar-10/SVHN, respectively.

As mentioned above ($K = C_k = 30$ case on SVHN), this phenomenon can be explained by too much diversity of models across users. Here, using $C_k < K$ actually acts as an implicit regularization and alleviates the effect of diversity. In order to explore our conjecture, inspired by [38], we use weight diversity to analyze the diversity of models across users. The weight diversity[6] is defined as:

$$\Delta_{gd}^t(w) = \sum_{i=1}^K \left\|w_i^t - w_{avg}^t\right\|_2^2 / \|\sum_{i=1}^K (w_i^t - w_{avg}^t)\|_2^2, \tag{8}$$

where w_*^t represents the weighs at the t-th iteration for the server/user. We conduct our weight diversity experiments on the classifier layer of ResNet-18 on Cifar-10, and the results in the setting of t mod $T = 0$ are shown in Fig. 4. Note that when $C_k = K$, the convergence speed of $K = 30$ is much slower than $K = 20$ (red curves in the top of Fig. 4). However, with fixed $C_k = 10$, the effect of increasing K from 20 to 30 is negligible (purple curves in the top of Fig. 4). The bottom of Fig. 4 show the weight diversity under different settings. It is obvious that the diversity of $C_k = 10$ is much less, compared to $C_k = K$ for both $K = 20$ and $K = 30$. From a finer-grained view, when $C_k = K$, the weight diversity (purple curves) does not vanish until epoch 20. However, for $C_k < K$, the weight diversity (red curves) quickly converges to a flat region. This is consistent with the training curve shown on the left of Fig. 4.

[6]Here, we can think of this weight diversity as the same as gradient diversity [38] since multi-step updates in a local user can be considered as the gradient accumulation.

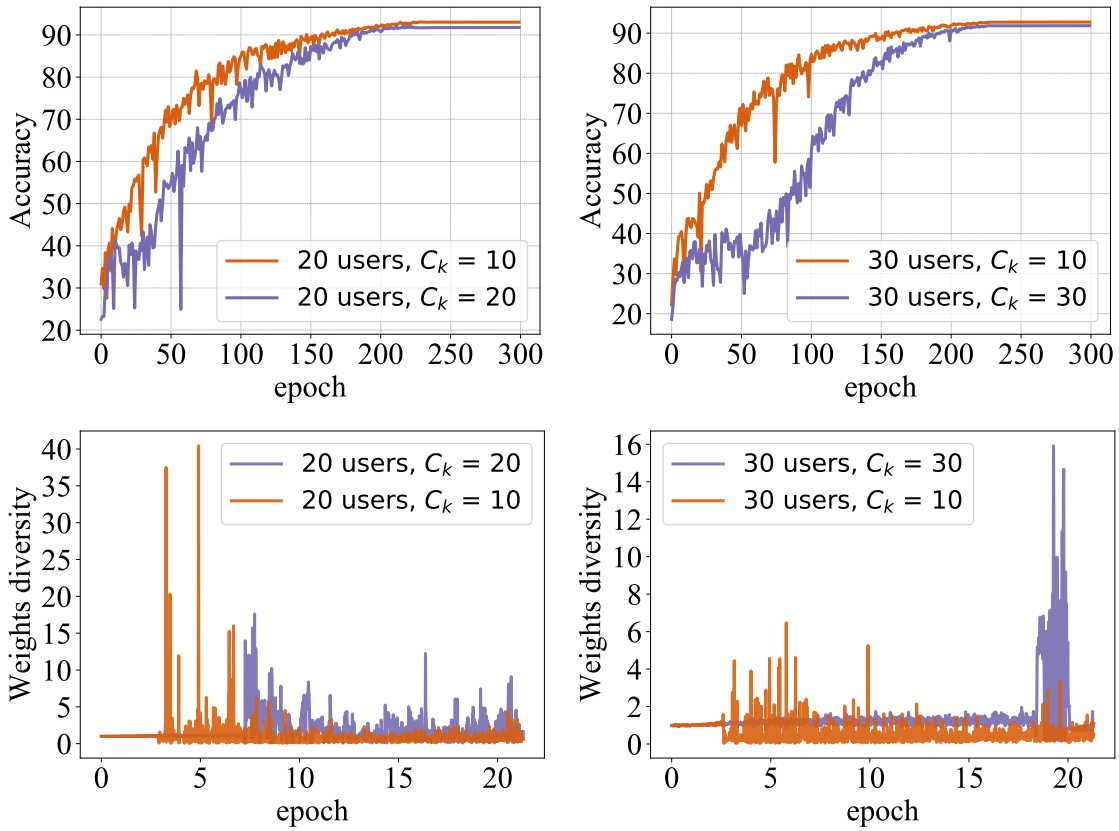

Figure 4: *(Top) The convergence of $K = 20/30$ on Cifar-10. (Bottom) The weight diversity of $K = 20/30$ on Cifar-10.*

Table 3: *Comparison with supervised FL. Our proposed framework achieves better results even when the non-iid level is high and users have no labeled data. Here, "*" is calculated according to the setting in DataSharing [9].*

Scenario	Supervised FedAvg [4] (users have labels)	DataSharing [9] (users have labels)	Our FedAvg solution (users have no label)
Cifar-10, Non-iid	78.52% ($R = 0.29$)	81.82% ($R = 0.29^*$)	91.35% ($R = 0.4$)

4.3 Impact of model averaging methods

To verify the effectiveness of our grouping-based model averaging algorithm, we perform experiments on all three datasets. In particular, we set $K = 10/20/47$, $R = 0.4/0.4/0.4$, $C_k = 10/20/47$, $T = 16/16/16$, and $N_s = 1000/1000/4700$ for Cifar-10/SVHN/EMNIST, respectively. For the grouping-based algorithm, we use group number $S = 2/2/5$ on Cifar-10/SVHN/EMNIST, respectively.

Our results are shown in Tab. 2. It can be seen that the accuracy of the grouping-based setting is 0.55%/0.16%/10.79% higher than FedAvg on three datasets. Note that our grouping-based structure can reduce the effect of the $C_k = K$ problem (i.e., weight diversity issue), as discussed in § 4.2 on EMNIST. See the EMNIST results in Tab. 2 and Tab. C.1 Besides, the convergence behavior of the grouping-based method can be found in Appendix E.

Table 4: *Comparison with EASGD [39] and OverlapSGD [1] on Cifar-10.*

Scenario	EASGD [39] (users have labels)	OverlapSGD [1] (users have labels)	Our FedAvg solution (users have no label)	Our Grouping-based solution (users have no label)
Non-iid, $T = 2$	91.12%	91.63%	93.83%	**94.22**%
Non-iid, $T = 8$	88.88%	91.45%	92.52%	**93.58**%
Non-iid, $T = 32$	–	–	91.28%	**91.92**%

4.4 Comparison with supervised results

We compare our FedAvg method with other FL algorithms in Tab. 3. Due to the lack of semi-supervised setting in the current literature, we choose two supervised FL methods for comparison, Supervised FedAvg [4] and DataSharing [9]. We set $K = 10$, $C_k = 10$ and $T = 32$, and we use ResNet-18 [31] to be the model for training. The non-iid setting of DataSharing [9] corresponds to the scenario where we set $R = 0.29$. For our FedAvg solution, we set $N_s = 1000$ and $R = 0.4$. From Tab. 3 we see that the performance of the our FedAvg method ($R = 0.4$) on Cifar-10 is still better than supervised FedAvg ($R = 0.29$) and DataSharing methods ($R = 0.29$).

We also compare our solutions (both FedAvg and grouping-based model averaging) with EASGD [39] and OverlapSGD [1] which are communication efficient algorithms under supervised settings. We set $K = 16$, $R = 0.4$, $C_k = 16$ and $N_s = 1000$ on Cifar-10 for our methods, and the results are shown in Tab. 4. As one can see, our result is much better than both EASGD and OverlapSGD. Particularly, even with $T = 32$, our grouping-based solution has 0.80%/0.29% better performance, as compared to EASGD/OverlapSGD in the setting of $T = 2$, respectively. Note that both EASGD and OverlapSGD are supervised algorithms, which means they have all the data labeled.

5 Conclusions

We have proposed the first semi-supervised federated learning (SSFL) framework, where only the server has a small amount of labeled data, and where the users have only unlabeled data. We defined a metric to measure the non-iid level quantitatively; and we studied our framework under different factors, including the non-iidness R, the communication period T, the number of users K, the number of labeled data N_s, and the number of users C_k communicating to the server with each communication round to benchmark our SSFL framework. Extensive experiments conducted on Cifar-10, SVHN, and EMNIST demonstrated that the simple consistency loss-based method, along with group normalization, could already achieve higher generalization performance than previous supervised federated learning settings. In addition, with the proposed novel grouping-based model averaging, we can further improve the generalization performance. It is worth emphasizing that our method has a certain generality, and it can be easily extended to other federal learning scenarios, such as traditional federated learning framework [4] and label-centralized and distributed federated learning scenarios introduced in [6].

Acknowledgments

We would like to thank Jianyu Wang and Daniel Rothchild for their valuable feedback. We would like to acknowledge DARPA, NSF, and ONR for providing partial support of this work.

References

[1] J. Wang, H. Liang, and G. Joshi, "Overlap local-SGD: An algorithmic approach to hide communication delays in distributed SGD," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8871–8875, 2020.

[2] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186, Springer, 2010.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[5] J. Konečnỳ, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[6] 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.

[7] A. Albaseer, B. S. Ciftler, M. Abdallah, and A. Al-Fuqaha, "Exploiting unlabeled data in smart cities using federated learning," *arXiv preprint arXiv:2001.04030*, 2020.

[8] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," *arXiv preprint arXiv:1911.02054*, 2019.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[10] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

[13] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Advances in neural information processing systems*, pp. 1163–1171, 2016.

[14] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.

[15] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[16] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Advances in Neural Information Processing Systems*, pp. 9850–9861, 2018.

[17] N. Guha, A. Talwlkar, and V. Smith, "One-shot federated learning," *arXiv preprint arXiv:1902.11175*, 2019.

[18] J. Xu, W. Du, R. Cheng, W. He, and Y. Jin, "Ternary compression for communication-efficient federated learning," *arXiv preprint arXiv:2003.03564*, 2020.

[19] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[20] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

[21] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "Federated optimization for heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 1, no. 2, p. 3, 2018.

[22] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, IEEE, 2018.

[23] H. B. McMahan, D. M. Bacon, J. Konecny, and X. Yu, "Communication efficient federated learning," Nov. 7 2019. US Patent App. 16/335,695.

[24] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[25] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in neural information processing systems*, pp. 3546–3554, 2015.

[26] 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*, pp. 1195–1204, 2017.

[27] 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*, pp. 5050–5060, 2019.

[28] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *arXiv preprint arXiv:1911.09785*, 2019.

[29] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le, "Self-training with noisy student improves imagenet classification," *arXiv preprint arXiv:1911.04252*, 2019.

[30] 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*, pp. 248–255, Ieee, 2009.

[31] 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*, pp. 770–778, 2016.

[32] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[33] M. Basseville, "Distance measures for signal processing and pattern recognition," *Signal processing*, vol. 18, no. 4, pp. 349–369, 1989.

[34] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," *arXiv preprint arXiv:1908.02983*, 2019.

[35] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical data augmentation with no separate search," *arXiv preprint arXiv:1909.13719*, 2019.

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

[37] K. Hsieh, A. Phanishayee, O. Mutlu, and P. B. Gibbons, "The non-iid data quagmire of decentralized machine learning," *arXiv preprint arXiv:1910.00189*, 2019.

[38] D. Yin, A. Pananjady, M. Lam, D. Papailiopoulos, K. Ramchandran, and P. Bartlett, "Gradient diversity: a key ingredient for scalable distributed learning," *arXiv preprint arXiv:1706.05699*, 2017.

[39] S. Zhang, A. E. Choromanska, and Y. LeCun, "Deep learning with elastic averaging sgd," in *Advances in neural information processing systems*, pp. 685–693, 2015.

[40] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[42] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017.

[43] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[44] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

A Experiment details

A.1 Datasets

Cifar-10 [40] consists of images with 3 channels, each of size 32×32 pixels. Each pixel is represented by an unsigned int8. This dataset consists of 60000 color images from 10 classes, with 6000 images in each class. There are 50000 training images and 10000 test images.

SVHN [41] is obtained from images of house numbers in Google Street View images. It has 99289 digits from 10 classes. There are 73257 digits for training and 26032 digits for testing. All digits have been resized to a fixed resolution of 32×32 pixels.

EMNIST [42] is a set of handwritten digits which have been resized to a fixed resolution of 28×28 pixels. It is an unbalanced dataset that has 814255 digits from 62 classes, including A-Z, a-z, and 0-9. However, since the uppercase and lowercase of some handwritten letters are difficult to distinguish, for these letters, the uppercase and lowercase classes are combined into a new class. There are 15 merged letters in total, including [C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z]. Thus, there are 47 classes left. To make sure every user has almost the same amount of data, we truncated the training dataset to have 2400 data points per class and drop the rest. That is to say, there are 112800 digits for training, and we remain the test dataset as the original one (18800 digits for testing).

A.2 Data distribution

For a dataset with d classes, we assume the distribution across different classes is (probabilistic mass function) $Q = [q_1, \ldots, q_d]$. For instance, for Cifar-10 dataset, $Q = [0.1, \ldots, 0.1]$. We follow the following procedures to distribute the data.

Step 1: We assign N_s/d labeled training samples from each class to the server. Recall that N_s is the total number of samples in the server.

Step 2: Assume there are n_j samples left in class j after distributing the data to the server. We assume there are m_j users (denoted as User_1^j, User_2^j, ..., $\text{User}_{m_j}^j$) who use class j as the main class (the number of samples in the main class is greater than the number of samples in other classes). For simplicity, we use User_1^j as an example to illustrate the data assignment (the same for other users):

(a) We first assign $n_j * R/m_j$ unlabeled training samples for the main class j.

(b) After that, for each class i from $\{1, \ldots, d\}$, we assign $n_i * q_j * (1 - R)/m_j$ unlabeled samples.

In this case, the total distance based on Definition 1 is R.

A.3 Optimizer

The optimizer used in all experiments is SGD with momentum. For the learning rate schedule, we use the cosine learning rate decay [43] shown in Eq. 9 below, which is a commonly used schedule in semi-supervised learning [14]:

$$\gamma_t = \gamma \times \max\left\{\cos\left(\pi \times c \times \frac{t - eM/B}{EM/B - eM/B},\right), \varepsilon\right\}, \tag{9}$$

where γ is the base learning rate, c is the periodic coefficient, E is the number of training epochs, B is the batch size, t is the current iteration, M is the number of training samples used in one epoch, e is the number of epochs for warmup [44], and ε is a small constant. The hyperparameters used for different datasets can be found in Tab. A.1.

We believe that if one further tunes those hyperparameters, the performance of our FedAvg and grouping-based methods will be further improved. For example, for SVHN dataset, in the setting of $T = 16$, $R = 0.4$, $N_s = 1000$ and $K = C_k = 10$, when $E = 120$ our FedAvg method can get 96.17% accuracy which is 2.71% higher than $E = 40$ as reported in Tab. 1.

Table A.1: *Optimizer hyperparameters used on different datasets.*

Dataset	E	M	γ	e	ε	weight decay	momentum	c	B
Cifar-10	300	65536	0.146	5	1e-4	1e-4	0.9	2.3	64
SVHN	40	65536	0.146	5	1e-4	1e-4	0.9	2.3	64
EMNIST	100	65536	0.03	0	1e-4	1e-4	0.9	0.4375	64

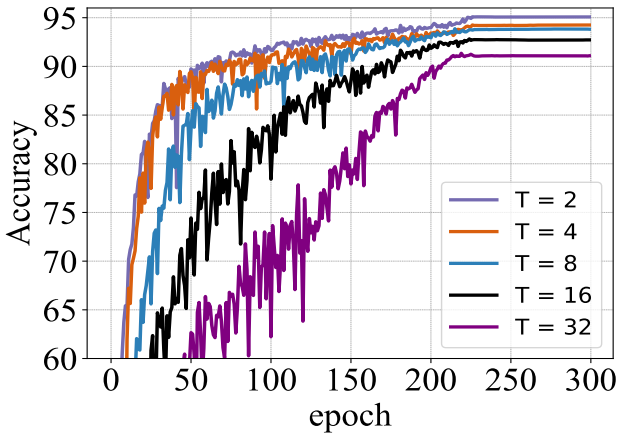

Figure B.1: *Accuracy curves of our FedAvg method of different communication periods on Cifar-10*

B Convergence speed of different communication period T on Cifar-10

The accuracy curves of different communication period T on Cifar-10 are shown in Fig. B.1. The experimental settings are presented in § 4.1. From Fig. B.1, we can see that when T is small, the FedAvg method converges fast, otherwise, it converges slow.

C Impact of the number of communication users C_k on EMNIST dataset

We set $K = 47$, $T = 16$, $R = 0.4$ and $N_s = 4700$. In this setting, the performances of our FedAvg method on EMNIST with different C_k are shown in Tab. C.1. As can be seen, a similar conclusion as we shown in § 4.2 holds. That is when K is large, a small C_k can improve the performance.

D Performance of different user number K on SVHN

As shown in Tab. 1, the accuracy of SVHN with $K = C_k = 30$ and $E = 40$ is much lower than $K = C_k = 10$ and $E = 40$. Increasing the number of training epochs to $E = 120$ can significantly improve the performance

Table C.1: *Accuracy versus the amount of communicated user C_k on EMNIST dataset*

	$K = 47$, $C_k = 10$	$K = 47$, $C_k = 30$	$K = 47$, $C_k = 47$
EMNIST	83.49%	79.05%	70.84%

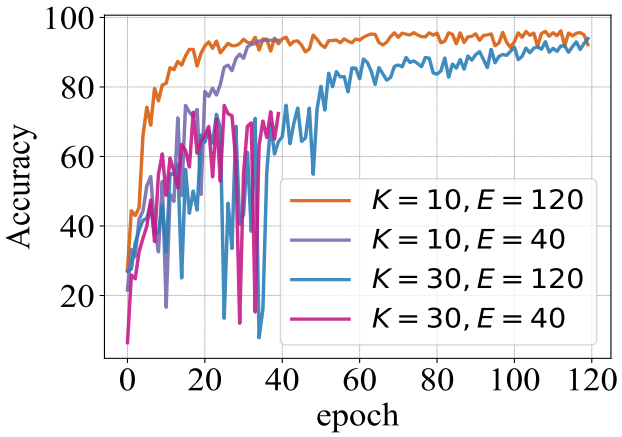

Figure D.1: *Accuracy curves of our FedAvg method for different K and E on SVHN.*

Table F.1: *Accuracy versus different ratios of communication users on Cifar-10 and SVHN*

	$u = 1/10$ ($K = 30$, $C_k = 3$)	$u = 1/5$ ($K = 30$, $C_k = 6$)	$u = 1/3$ ($K = 30$, $C_k = 10$)	$u = 1$ ($K = 30$, $C_k = 30$)
Cifar-10	80.98%	88.66%	92.78%	91.87%
SVHN	92.15%	93.12%	93.37%	74.71%

of $K = C_k = 30$. The comparison is shown in Fig. D.1. In the meantime, as we mentioned in Appendix A.3, increasing training epochs can also benefit $K = C_k = 10$ (the accuracy curve is also shown in Fig. D.1).

E Convergence behaviors of FedAvg and grouping-based averaging methods

As can be seen from Fig. E.1, when the the number of users that communicate with the server is large, the grouping-based averaging method can accelerate the convergence compared to the FedAvg method. From Fig. E.1, we also see that if we use naive FedAvg, the large model diversity across different users significantly slows down the training process. However, the grouping-based structure can effectively alleviate the weight diversity issue and improve the convergence speed.

F Impact of the ratio C_k/K

In real-life FL scenarios, the number of connected users can vary during training. Here, we define the ratio of connected users as $u = C_k/K$, and we study the impact of u on Cifar-10 and SVHN datasets.

We set $T = 16/16$, $R = 0.4/0.4$, and $N_s = 1000/1000$ for the two datasets Cifar-10 and SVHN. We use ResNet-18 [31] as the model for training, and we train 300/40 epochs on Cifar-10/SVHN. The results of our FedAvg method on Cifar-10 and SVHN are shown in Tab. F.1. It can be found that as u increases from 1/10 to 1/3, the performance of our FedAvg method gradually improves. This result is intuitive since a smaller u means fewer models participate in the model averaging, only a small ratio of the models can be utilized. It should also be noted that, as shown in § 4.2 when u increases to a large value (e.g. $u = 1$), the diversity of models across users will be large, and the performance will decrease. Therefore, properly increasing u can improve the performance but increasing u to much can also decrease the performance. However, more work needs to be done to explore what are the most effective ratios for different datasets.

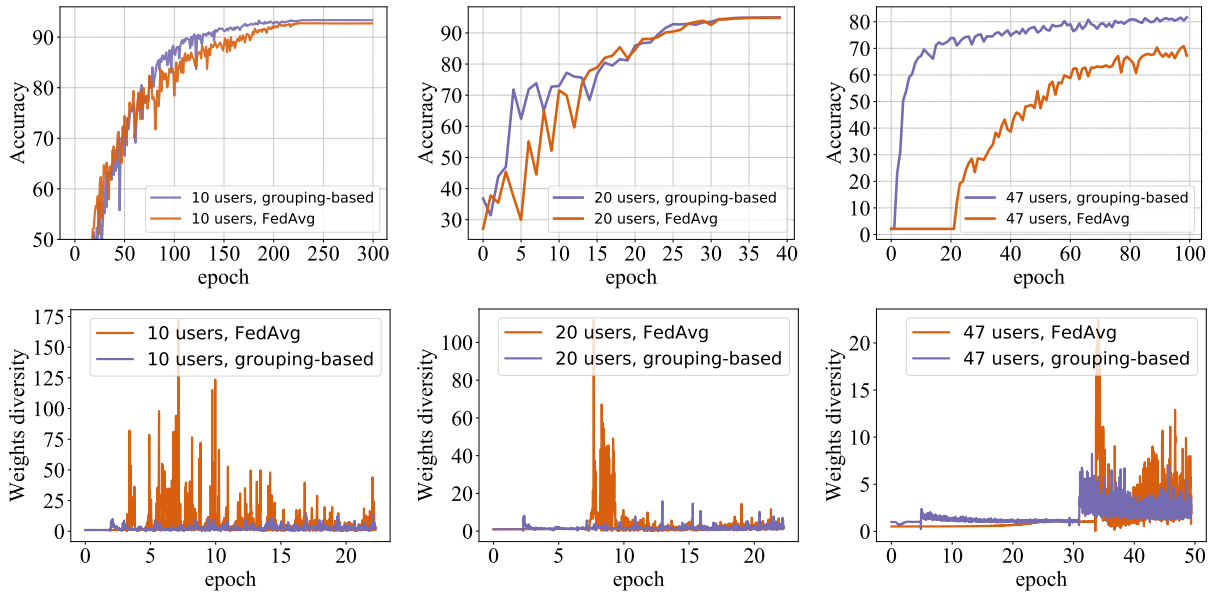

Figure E.1: *(Top) The convergence behaves of FedAvg method and grouping-based method on (left) Cifar-10, (middle) SVHN and (right) EMNIST. (Bottom) The weight diversities of FedAvg method and grouping-based method on (left) Cifar-10, (middle) SVHN and (right) EMNIST.*