FedRS: Federated Learning with Restricted Softmax for Label Distribution Non-IID Data

Xin-Chun Li, De-Chuan Zhan
State Key Laboratory for Novel Software Technology, Nanjing University
Nanjing 210023, China
lixc@lamda.nju.edu.cn,zhandc@nju.edu.cn

ABSTRACT

Federated Learning (FL) aims to generate a global shared model via collaborating decentralized clients with privacy considerations. Unlike standard distributed optimization, FL takes multiple optimization steps on local clients and then aggregates the model updates via a parameter server. Although this significantly reduces communication costs, the non-iid property across heterogeneous devices could make the local update diverge a lot, posing a fundamental challenge to aggregation. In this paper, we focus on a special kind of non-iid scene, i.e., label distribution skew, where each client can only access a partial set of the whole class set. Considering top layers of neural networks are more task-specific, we advocate that the last classification layer is more vulnerable to the shift of label distribution. Hence, we in-depth study the classifier layer and point out that the standard softmax will encounter several problems caused by missing classes. As an alternative, we propose "Restricted Softmax" to limit the update of missing classes' weights during the local procedure. Our proposed FedRS is very easy to implement with only a few lines of code. We investigate our methods on both public datasets and a real-world service awareness application. Abundant experimental results verify the superiorities of our methods.

CCS CONCEPTS

• **Computing methodologies** → **Cooperation and coordination**; **Supervised learning by classification**; **Learning under covariate shift**.

KEYWORDS

federated learning; non-iid; label distribution shift; softmax

ACM Reference Format:
Xin-Chun Li, De-Chuan Zhan. 2021. FedRS: Federated Learning with Restricted Softmax for Label Distribution Non-IID Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467254

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467254

1 INTRODUCTION

Although deep learning has experienced great success in many fields [18, 22, 30], a data center training paradigm is usually required. Due to data privacy or transmission cost, data from individual participants can not be located on the same device in some real-world applications [10, 43]. Standard distributed optimization [12, 34] provides solutions to distributed training with big data or huge model, while Federated Learning (FL) [27, 38, 51] is tailored for data privacy protection and efficient distributed training. Specifically, FL takes multiple rounds of local and global procedures [13, 38] to collaborate isolated data islands (local clients). During local procedure, a subset of clients download the global model from the central server and update it on local private data. The global procedure is taken by the central server to aggregate the local model updates. These two procedures are iterated until convergence. In FL, only model parameters are transmitted among clients and server, which brings basic privacy protection. Advanced privacy protection methods, e.g., differential privacy [1, 17, 48], can be further applied for stricter privacy protection. In another view, local clients take more computation steps, e.g., epochs of training on local data, making the decentralized training more efficient.

FL also faces many challenges, e.g., massive amounts of devices, limited communication, and non-iid property, etc [38]. The non-iid data distribution across clients is the most fundamental statistical challenge, while others are major obstacles at the system level [35]. In this paper, we mainly focus on the non-iid challenge. With heterogeneous local data, the local training procedure will diverge a lot from the global target due to the discrepancy between the local and global data distribution [57]. The stronger the heterogeneity of the local data set, the larger the distribution discrepancy, and the harder it is to aggregate a well-performed global model. Hence, some existing methods add various regularizations to restrict the local models not diverge from the global model too much [35, 44, 52].

As further studied in the recent survey of FL [27], the non-iid scenes can be subdivided into five categories: feature distribution skew, label distribution skew, concept shift with different features, concept shift with different labels, and quantity skew. This paper mainly focuses on the label distribution skew, where the prior distributions across clients $\mathcal{P}^k(y)$ may vary a lot, but $\mathcal{P}^k(x|y)$ is the same, e.g., the distribution of animal species varies across regions [27]. Existing studies show that the bottom layers of neural networks extract common features and are more transferable than the top layers [54]. Hence, the topmost layer is the most task-specific, which is widely implemented with a softmax classification layer, e.g., a combination of softmax and cross entropy loss [18, 25, 30]. Faced with label distribution shift, we advocate that this layer could be the most vulnerable. Specifically, we first in-depth analyze and show the

inherent pulling and pushing forces in standard softmax (Sect. 3.1). Then we point out that several properties will disappear faced with missing classes (Sect. 3.2), which will be encountered during local training procedure of FL with label distribution skew (Sect. 4.1). As an alternative, we slightly modify the standard softmax and introduce "Restricted Softmax" to limit the update of missing classes' weights during local procedure (Sect. 4.3). We briefly introduce our methods with three folds: (1) it is the first work diving into the softmax classification layer to solve the label distribution skew in FL; (2) it is easy to implement with only a few lines of code; (3) it is consistent with many classic methods as analyzed in Sect. 4.4.

2 RELATED WORKS

FL with Non-IID Data Various techniques have been proposed to solve the non-iid challenge in FL. A natural solution is sharing a small public data set among clients, being a compromise for privacy protection [26, 50, 53]. Some methods resort to multi-task learning [11, 45] or meta learning [9, 15] for fast local adaptation. Taking advantage of fully decentralized learning [4] or private-shared models [3, 36, 40] are solutions for better aggregation and personalization. Updating models with momentum on server can lead to stable performances [28]. The most similar work to ours is adding regularization terms during local procedure. FedProx [35] introduces a proximal term and directly restricts the local model parameters not diverge from the global model too much. FedMMD [52] aims to mitigate the discrepancy between features extracted by local and global models. Except that FedAwS [55] studies the extreme scene that each client could only access one class, relatively few considerations have been shown on the final classification layer under label distribution skew.

Label Distribution Shift There are several scenes that are directly related to label distribution shift. Dataset shift [42] is originally categorized into covariate shift and label shift. In area of transfer learning, joint distribution alignment [37], generalized domain adaptation via co-alignment [47], and partial domain adaptation [7] provide insights for solving label shift problems between source and target domains. Class imbalanced learning with long tail data poses a significant challenge for classifiers which could get biased towards frequent classes [6, 21, 24]. Class incremental learning with new classes [23] and few-shot classification for generalizing to unseen meta-test classes [16] handle the class drift problem in sequential tasks. Different from these studies, our work studies the label distribution shift problem in FL and searches solutions from aspect of the most vulnerable task-specific layer, i.e., the final softmax classification layer.

3 INTUITION AND MOTIVATION

As a major motivation, we progressively introduce the properties of softmax and the problems when faced with missing classes.

3.1 Properties of Softmax

In standard classification, all samples are centralized on the same device. We denote $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ as the training set with N samples, where $y_i \in C = \{1, 2, \ldots, C\}$ and C is the number of classes. Deep networks always contain the feature extractor $F_\theta(\cdot)$ with parameters θ and the last classification layer with weights $\{\mathbf{w}_c\}_{c=1}^C$ (we omit the bias for simplification). *Without additional declaration, we refer to the last classification layer as the classifier.* We denote $\mathbf{h}_i = F_\theta(\mathbf{x}_i) \in \mathcal{R}^d$ as the extracted feature vector of the i-th sample. Borrowing from some related works [39, 46, 56], we refer to the classification weights $\{\mathbf{w}_c\}_{c=1}^C$ as *proxies*. For the c-th proxy $\mathbf{w}_c$, we denote the features from the c-th class and other classes as *positive features* and *negative features* respectively.

The softmax operator normalizes the scores of each class (i.e., $\mathbf{w}_c^T \mathbf{h}_i$) and returns the probability:

$$p_{i,c} = \frac{\exp(\mathbf{w}_c^T \mathbf{h}_i)}{\sum_{j=1}^C \exp(\mathbf{w}_j^T \mathbf{h}_i)}, \tag{1}$$

and the cross-entropy loss is calculated as:

$$\mathcal{L} = -\sum_{i=1}^N \sum_{c=1}^C \mathcal{I}\{y_i = c\} \log p_{i,c}, \tag{2}$$

where $\mathcal{I}\{\cdot\}$ is the indication function. The gradient of $\mathbf{w}_c$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_c} = -\sum_{i=1}^N \left(\mathcal{I}\{y_i = c\} - p_{i,c}\right) \mathbf{h}_i. \tag{3}$$

We use gradient descent with learning rate η to update $\mathbf{w}_c$ and decompose this update into the *pulling* and *pushing* forces, i.e.,

$$\mathbf{w}_c = \mathbf{w}_c + \underbrace{\eta \sum_{i=1, y_i=c}^N \left(1 - p_{i,c}\right) \mathbf{h}_i}_{\text{pulling}} \underbrace{-\eta \sum_{i=1, y_i \neq c}^N p_{i,c} \mathbf{h}_i}_{\text{pushing}}, \tag{4}$$

from which we can obtain the inherent properties of softmax:

Property 1 (Properties of Softmax). *Classification with softmax has the following properties: (1) pulling proxies closer to positive features; (2) pushing proxies away from negative features.*

These properties are illustrated in Fig. 1 (A), where a demo with three classes is shown. We only show the properties of updating proxy $\mathbf{w}_1$. The data region $\mathbf{X}_1$ contains positive features, while $\mathbf{X}_2$ and $\mathbf{X}_3$ contain negative features. Hence, $\mathbf{w}_1$ is pulled closer to $\mathbf{X}_1$ and pushed away from $\mathbf{X}_2$, $\mathbf{X}_3$ simultaneously. The properties are directly related to deep metric learning [19, 39, 49, 56], where forces of pulling and pushing exist together. However, the balance could be broken in same cases.

3.2 Softmax with Missing Classes

In some cases, we can not obtain training samples of several classes, named as **missing classes**. Formally, the label set C is split into observed class set O and missing class set $\mathcal{M}$ respectively. We have $O \cap \mathcal{M} = \emptyset$ and $O \cup \mathcal{M} = C$, where $\emptyset$ is the empty set. We still denote the training set as $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, while y_i is only from O. Then, the Eq. 4 can be adapted. For $c \in \mathcal{M}$:

$$\mathbf{w}_c = \mathbf{w}_c + \underbrace{\eta \sum_{i=1, y_i=c}^N \left(1 - p_{i,c}\right) \mathbf{h}_i}_{=0} \underbrace{-\eta \sum_{i=1, y_i \neq c}^N p_{i,c} \mathbf{h}_i}_{\text{only pushing exists}}. \tag{5}$$

This shows that due to lacking corresponding training samples, the pulling force of proxy $\mathbf{w}_c, c \in \mathcal{M}$ is missing, and the proxy is only pushed away from negative features. This phenomenon is

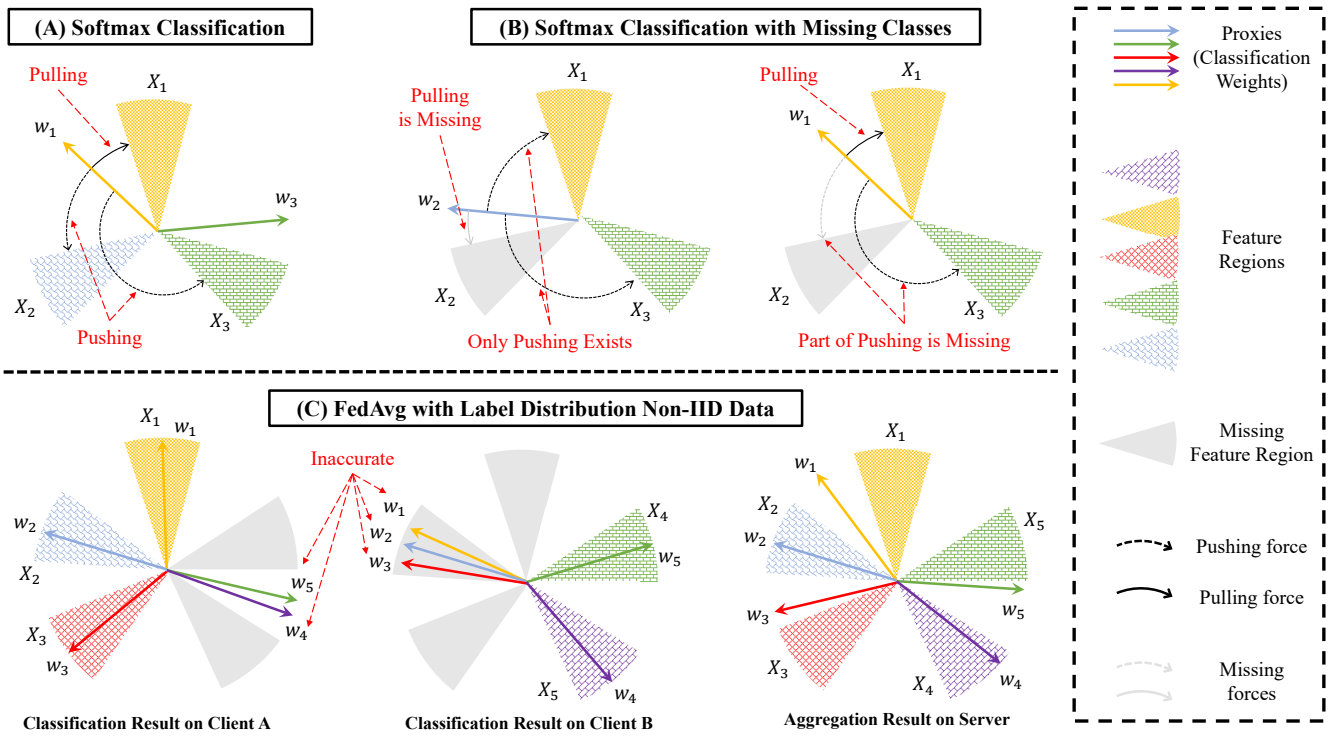

Figure 1: Illustration of the motivations. The right shows the legends. (A) Softmax classification: the proxy is pulled towards positive features and pushed away from negative features. (B) Softmax with missing classes: the second class is missing; its corresponding proxy is only pushed away from negative features (left); the proxies of observed classes lack part of pushing forces (right). (C) FedAvg with label distribution non-iid data (5-class case): the global target is to classify 5 classes, while client A and B only has access to 3 and 2 classes respectively; proxies of missing classes may become inaccurate on local clients (left and middle), leading to a poor aggregation on server (right).

illustrated in the left part of Fig. 1 (B), where $\mathbf{w}_2$ is the proxy of the missing class and the gray arc implies the pulling property is missing. For the observed class $c \in O$, we have the update of $\mathbf{w}_c$:

$$\mathbf{w}_c = \mathbf{w}_c + \eta \underbrace{\sum_{i=1, y_i=c}^{N} \left(1 - p_{i,c}\right) \mathbf{h}_i}_{\text{pulling}} - \eta \underbrace{\sum_{i=1, y_i \neq c}^{N} p_{i,c} \mathbf{h}_i}_{(|O|-1) \text{ classes}}, \tag{6}$$

where we can find that the pushing force becomes weaker due to that the negative features only come from $|O| - 1$ classes. This is illustrated in the right part of Fig. 1 (B).

These missing properties could iteratively lead to error accumulation in update of features. This can be observed through the update of $\mathbf{h}_i$, where we only show the gradient of $\mathbf{h}_i$ instead of further propagating it backward to θ for simplification:

$$\mathbf{h}_i = \mathbf{h}_i + \underbrace{\eta \left(1 - p_{i,y_i}\right) \mathbf{w}_{y_i}}_{\text{pulling}} - \underbrace{\eta \sum_{j \in O, j \neq y_i}^{C} p_{i,j} \mathbf{w}_j}_{(|O|-1) \text{ classes}} - \underbrace{\eta \sum_{j \in \mathcal{M}} p_{i,j} \mathbf{w}_j}_{\text{inaccurate}}, \tag{7}$$

where the update of features can also be decomposed into pulling and pushing forces. The pulling force makes the features closer towards corresponding proxies, while the pushing force makes the features away from proxies of other classes. Compared with the standard softmax, the pushing force is weaker due to the proxies of missing classes could be inaccurate and the effective ones are only from $|O| - 1$ classes. This could result in less compact features as declared in [8]. Due to the update of features are too complex to analyze, we only analyze the update of classifier in the following, i.e., the proxies, for simplification. Another reason for this is that the classifier is most vulnerable under label distribution shift as aforementioned. We conclude the above analysis as a problem of softmax with missing classes:

Property 2 (Problem of Softmax with Missing Classes). *With missing class set $\mathcal{M}$, the softmax classification has following problems: (1) the proxies of missing classes, i.e., $\{\mathbf{w}_c\}_{c \in \mathcal{M}}$, are only pushed away from negative features, becoming more and more inaccurate; (2) the proxies of observed classes, i.e., $\{\mathbf{w}_c\}_{c \in O}$, are only pushed away from $|O| - 1$ negative feature regions.*

A natural solution for missing classes is to discard the proxies of $\mathcal{M}$ and obtain a $|O|$-class classification problem. However, these proxies can not be directly discarded in FL with label distribution non-iid data.

4 OUR METHODS

In this section, we will first formally introduce the focused problem, i.e., FL with label distribution non-iid data, and then introduce the drawbacks of standard FL algorithms. Finally, we present our methods and provide a thorough analysis.

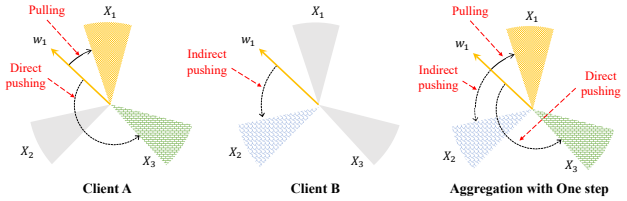

Figure 2: Illustration of the update of proxies (3-class case). We split the pushing force into direct and indirect ones. Client A has samples from the first class while client B can not. The proxy $\mathbf{w}_1$ is updated with a balance of pulling and direct pushing on client A, while it only undergoes the indirect pushing on client B. With multiple local training steps and without the balance of the pulling force, the indirect pushing on client B could diverge a lot from the oracle one, leading to a poor aggregation.

4.1 FL with Label Distribution Non-IID Data

Suppose we have K clients, and each client owns a local data distribution $\mathcal{D}^k = \mathcal{P}^k(\mathbf{x}, y)$. FL algorithms aim to optimize a combination of local losses, i.e., $\min_{\psi=(\theta,\{\mathbf{w}_c\}_{c=1}^C)} \sum_{k=1}^K p_k \mathcal{L}^k(\psi; \mathcal{D}^k)$. p_k is the weight of each client, satisfying $p_k \geq 0, \sum_{k=1}^K p_k = 1$. $\mathcal{L}^k(\psi; \mathcal{D}^k)$ is the local loss as in Eq. 2, while it is calculated based on local data $\mathcal{D}^k$. In this paper, we focus on label distribution non-iid challenge as categorized in [27], i.e., the $\mathcal{P}^k(y)$ may vary a lot among clients, while the $\mathcal{P}^k(\mathbf{x}|y)$ may be the same. Hence, we assume each client only has access to a subset of classes. Formally, the k-th client has a training set $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k}$ with N_k samples, and the label y_i^k is only from O^k, i.e., the observed class set of the k-th client. Similarly, the missing class set is denoted as $\mathcal{M}^k$. Note that the local client k actually does a $|O^k|$-class classification problem, while the global target aims to obtain a C-class classification model.

4.2 Drawbacks of Standard FL Algorithms

FedAvg [38], as the most standard FL algorithm, takes multiple rounds of local and global procedures to optimize the global model. In the beginning of each round, a subset of clients $S \subseteq \mathcal{K}$ is selected, where $\mathcal{K}$ is the set of all clients. Then, each selected client $k \in S$ executes the following local procedure in parallel.

During local procedure, the client downloads the global parameters, i.e., $\theta^k \leftarrow \theta$, $\mathbf{w}_c^k \leftarrow \mathbf{w}_c, \forall c \in C$. We use superscript "$k$" to discriminate local parameters from the global ones. Then, the client updates the downloaded model on local training set $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k}$. The downloaded global model contains the full set of proxies $\{\mathbf{w}_c\}_{c=1}^C$, while the k-th client only observes a partial set (i.e., O^k) of the whole class set. This local training procedure is just the introduced scene in Sect. 3.2, which has several problems as shown in Property. 2.

During server procedure, the server collects the updated parameters from these clients and takes a simple parameter averaging process. We denote $\hat{\theta}^k, \{\hat{\mathbf{w}}_c^k\}_{c=1}^C$ as the updated parameters of the k-th client, and the server updates the global model via: $\theta \leftarrow \frac{1}{|S|}\sum_{k=1}^{|S|} \hat{\theta}_c^k$, $\mathbf{w}_c \leftarrow \frac{1}{|S|}\sum_{k=1}^{|S|} \hat{\mathbf{w}}_c^k, \forall c \in C$. A possible problem during aggregation is that: due to the missing classes' proxies of the k-th client, i.e., $\{\mathbf{w}_c^k\}_{c\in\mathcal{M}^k}$, are updated without the pulling force, it leads to inaccurate $\{\hat{\mathbf{w}}_c^k\}_{c\in\mathcal{M}^k}$ and incurs error accumulation during aggregation. This is illustrated in Fig. 1 (C), where the server aims to build a 5-class classification model while the two clients only observe 3 and 2 classes respectively. Proxies $\mathbf{w}_4$, $\mathbf{w}_5$ on client A, and $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3$ on client B could become inaccurate, leading to a poor aggregation.

4.3 Restricted Softmax and FedRS

During local procedure, we advocate that *the update of missing classes' proxies, i.e., $\{\mathbf{w}_c^k\}_{c\in\mathcal{M}^k}$, should be restricted.* An easy way to implement this is adding "scaling factors" to softmax operation, i.e.,

$$p_{i,c}^k = \frac{\exp(\alpha_c^k \mathbf{w}_c^{kT} \mathbf{h}_i^k)}{\sum_{j=1}^C \exp(\alpha_j^k \mathbf{w}_j^{kT} \mathbf{h}_i^k)}, \tag{8}$$

which is denoted as **restricted softmax**. We set $\alpha_c^k = \mathcal{I}\{c \in O^k\} + \alpha\mathcal{I}\{c \in \mathcal{M}^k\}$, where $\alpha \in [0, 1]]$ is the only hyper-parameter. This is an asymmetric scaling way that works normally with $\alpha_c^k = 1$ for observed classes while works as a decaying method with $\alpha_c^k = \alpha$ for missing classes. Although it is a simple modification, we in-depth analyze the brought advantages from several aspects.

Restricting update of missing classes' proxies. Based on the cross-entropy loss $\mathcal{L}^k = -\sum_{i=1}^{N_k}\sum_{c=1}^C \mathcal{I}\{y_i^k = c\} \log p_{i,c}^k$, we can obtain the gradients of $\mathbf{w}_c^k$:

$$\frac{\partial \mathcal{L}^k}{\partial \mathbf{w}_c^k} = -\alpha_c^k \sum_{i=1}^{N_k} \left(\mathcal{I}\{y_i^k = c\} - p_{i,c}^k\right) \mathbf{h}_i^k. \tag{9}$$

For missing class $c \in \mathcal{M}^k$, we have $\alpha_c^k = \alpha$ and the update process of $\mathbf{w}_c^k$ is:

$$\mathbf{w}_c^k = \mathbf{w}_c^k \underbrace{-\alpha\eta \sum_{i=1}^{N_k} p_{i,c}^k \mathbf{h}_i^k}_{\text{restricted}}, \tag{10}$$

where we can find that the pushing force is restricted with $\alpha \in [0, 1]$. If we take $\alpha = 0$, it degenerates into fixed proxies (if we do not consider weight decay); if $\alpha = 1$, it is just normal softmax. Overall, the norm of missing classes' gradients is restricted.

Restricting the inaccurate terms in update of features. For the i-th instance, we have $\alpha_{y_i^k} = 1$, $\alpha_{j\in O^k} = 1$, and $\alpha_{j\in\mathcal{M}} = \alpha$. We can obtain the update process of its corresponding features $\mathbf{h}_i^k$:

$$\begin{aligned} \mathbf{h}_i^k \quad = \quad & \mathbf{h}_i^k + \eta(1 - p_{i,y_i}^k)\mathbf{w}_{y_i}^k \\ & -\eta \sum_{j\in O^k, j\neq y_i^k}^{C} p_{i,j}^k \mathbf{w}_j^k \underbrace{-\alpha\eta \sum_{j\in\mathcal{M}^k} \alpha p_{i,j}^k \mathbf{w}_j^k}_{\text{restricted}}, \end{aligned} \tag{11}$$

where we can find that the inaccurate term in Eq. 7 is restricted by the scaling factor α. This brings one benefit that the features are updated towards more accurate directions.

Leading to more accurate aggregation of proxies. We use the restricted softmax during local training and aggregate the model as usual. We name our method as "Federated learning with Restricted Softmax", i.e., **FedRS**. We present and analyze the aggregation process of proxies in detail, i.e., $\mathbf{w}_c \leftarrow \frac{1}{|S|}\sum_{k=1}^{|S|} \hat{\mathbf{w}}_c^k, \forall c \in C$. In

FedAvg, the local procedure will update the downloaded parameters on local training data for several epochs. This is too complex to analyze, and we simplify this local procedure by only taking one gradient descent step. With restricted softmax, we can obtain the following propsition:

Proposition 1 (Aggregation with Restricted Softmax). *If we only take one gradient step during local procedure and use the restricted softmax in Eq. 8, the global update of* $\mathbf{w}_c$ *can be actually decomposed as:*

$$\mathbf{w}_c = \mathbf{w}_c + \underbrace{\frac{\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I^k_{1,i,c}(1-p^k_{i,c})\mathbf{h}^k_i}_{\textit{Part1: pulling}} \underbrace{-\frac{\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I^k_{2,i,c}p^k_{i,c}\mathbf{h}^k_i}_{\textit{Part2: direct pushing}} \underbrace{-\frac{\alpha\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I^k_{3,c}p^k_{i,c}\mathbf{h}^k_i}_{\textit{Part3: indirect pushing}}, \quad (12)$$

where $I^k_{1,i,c} = \mathcal{I}\{c \in O^k, y^k_i = c\}$, $I^k_{2,i,c} = \mathcal{I}\{c \in O^k, y^k_i \neq c\}$, *and* $I^k_{3,c} = \mathcal{I}\{c \in \mathcal{M}^k\}$ *are indication functions.*

Proof. The proof can be found in Appendix. □

The above Eq. 12 shows the proxies' update process on the server and it is the aggregation of local updates (Eq. 6). For a specific class c, we categorize the clients according to whether the client has access to samples of this class or not. If the k-th client observes this class, it can pull the proxy $\mathbf{w}_c$ towards the positive features (Part1). Simultaneously, it can push the proxy $\mathbf{w}_c$ away from a subset of negative classes (Part2). Otherwise, if the k-th client does not observe the c-th class, i.e., the c-th class belongs to the missing class set of the k-th client, only the pushing force is imposed (Part3). This is illustrated in Fig. 2, where we show a demo with 3 classes. Client A has data $\mathbf{X}_1$, and the proxy $\mathbf{w}_1$ is updated with forces of *pulling* and *direct pushing*. However, the $\mathbf{X}_1$ on client B is missing and $\mathbf{w}_1$ is imposed only with *indirect pushing* force. If we only take one update step on each client, the aggregation process is accurate and can be seen as an update with access to full class set. However, FL algorithms take multiple steps on local clients to reduce communication cost. Without the balance of pulling force, the individual indirect pushing force will be more and more inaccurate. As shown in Eq. 12, our proposed FedRS can restrict the indirect pushing force.

We then analyze the "strength" of the forces in FedRS. As shown in [8], the more of the number of classes, the learned features are more compact and the proxies are more discriminative. In Eq. 12, the proxy $\mathbf{w}_c$ is pulled towards positive features of its own class (Part 1), directly pushed away from at most $C_2 \triangleq |\bigcup_{k\in S, c\in O^k} O^k| - 1$ classes (Part 2), and indirectly pushed away from $C_3 \triangleq |\bigcup_{k\in S, c\in\mathcal{M}^k} \mathcal{M}^k|$ classes (Part 3). This implies that the client selection ratio and the specific class distribution in each client determines the strength of these forces. For example, if we select all clients in each round, i.e., $S = \mathcal{K}$, C_2 achieves its maximum value and the impact of indirect pushing will become relatively smaller. Hence, we advocate that FedRS can be more effective when client selection ratio is small, which is empirically verified in experimental studies (Fig. 7). The complete pseudo code of FedRS can be found in Appendix.

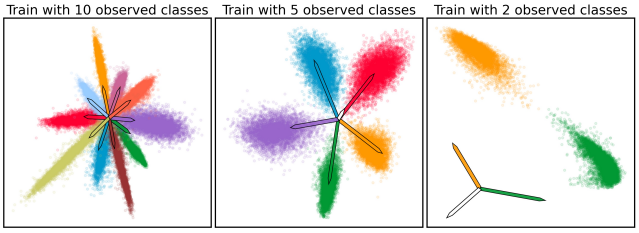

Figure 3: Visualization of learned features and proxies with 10, 5, and 2 observed classes respectively. For the latter two scenes, we also use the complete network with 10 proxies. The white arrows show the proxies of the missing classes.

4.4 Discussion from Other Aspects

Effective Learning Rate Adding scaling factors to softmax can be seen as applying an effective learning rate [2]. Our work can be seen as only decaying the learning rate of missing classes during local procedure and the effective learning rate is actually $\alpha\eta$, $\alpha \in [0, 1]$.

PC-Softmax PC-Softmax [41] proves the softmax cross entropy loss is a variational lower bound of mutual information between inputs and labels, using $\exp(\mathbf{w}_c^T\mathbf{h})/(\sum_{j=1}^C p(y=j)\exp(\mathbf{w}_j^T\mathbf{h}))$ as a probability corrected estimator for imbalanced data. If the prior distribution $p(y)$ is uniform, it differs from the normal softmax only with a constant. Our work can be seen as a similar way that correct the problem of imbalanced data from the aspect of "scores" as $p_c \propto \exp(p(y=c)\mathbf{w}_c^T\mathbf{h})$, where we take a smooth prior distribution $p(y=c) \propto \mathcal{I}\{c \in O\} + \alpha\mathcal{I}\{c \in \mathcal{M}\}$.

Transfer Adaptation The local procedure can be viewed as finetuning the downloaded global model on local data. As aforementioned, the classifier is most task-specific [54]. Our work can be seen as a careful finetuning process with more attention on the final layer faced with missing classes.

Weight Divergence The mismatch between global target and local data distributions will lead to weight divergence as shown in [53, 57]. Some approaches are proposed to constrain the update of local models, i.e., FedProx [35], FedMMD [52], etc. Our method works as diving into the final layer of the network and only constraining the missing classes' proxies instead of the whole model.

Fine-Grained Aggregation Standard FL algorithms take a simple parameter averaging for the proxies, e.g., $\mathbf{w}_c \leftarrow \sum_k p_k\hat{\mathbf{w}}^k_c$, where p_k is either set uniformly or proportional to the number of samples that the clients own. This is a coarse-grained aggregation. Considering the c-th class, the distribution among clients should be set as $p_{k,c} = N_{k,c}/\sum_k N_{k,c}$, where $N_{k,c}$ is the number of c-th class samples on k-th client. The fine-grained aggregation is $\mathbf{w}_c \leftarrow \sum_k p_{k,c}\hat{\mathbf{w}}^k_c$. We can also take a laplace smoothing, i.e., $p_{k,c} \propto N_{k,c} + \lambda$. Hence, if one client observes no samples of the c-th class, its importance is only proportional to λ due to $N_{k,c} = 0$. Our method is related to this when the α is small. For example, if we force $\alpha = 0$, the proxies of missing classes will not be updated (without weight decay) and it will not contribute to the aggregation.

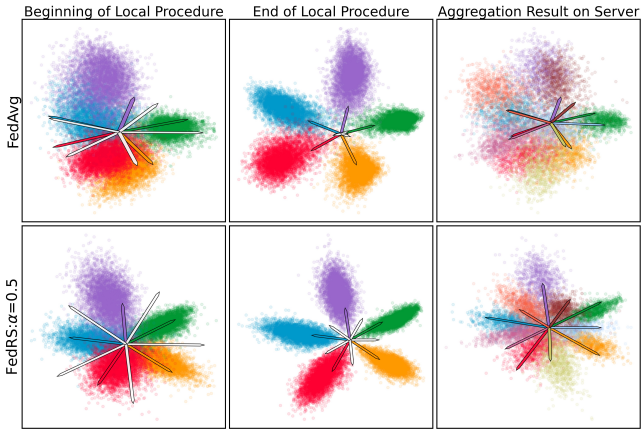

Figure 4: Visualization of learned features and proxies in FL with a 2-client label distribution shift scene (the 5-th round). We only plot for the first client. The two rows show FedAvg and FedRS ($\alpha = 0.5$) respectively. The three columns show the extracted features and proxies of the newly-downloaded model, local tuned model, and the aggregated model respectively. The white arrows show the proxies of the missing classes.

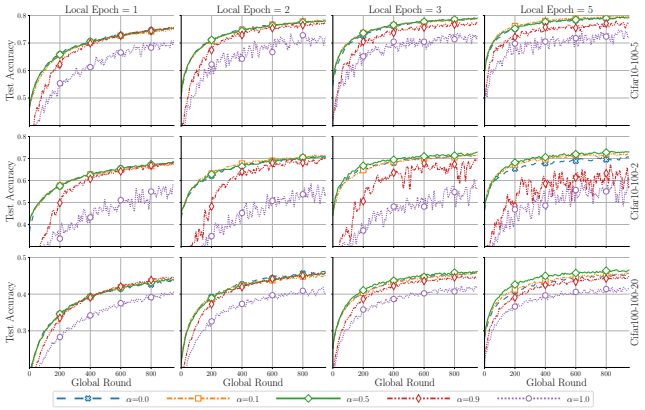

Figure 5: Performance comparisons based on TFCNN with various α in FedRS. Each row shows a scene. The four columns vary in local epochs. $\alpha = 1.0$ is just FedAvg, which performs poorly. FedRS with $\alpha = 0.5$ can almost obtain the best results.

5 EXPERIMENTS

We investigate our methods on several FL scenes with label distribution shift based on Mnist [31], Cifar10/Cifar100 [29]. We also investigate the performances of FedRS on large scale FL datasets, i.e., Shakespeare/FEMNIST in LEAF [5]. Finally, we report the effectiveness of FedRS on a real-world service awareness application. Some dataset details and hyper-parameters can be found in Appendix.

5.1 Visualization Results

We verify the motivation of our method via visualization on Mnist. We slightly modify LeNet [32] by setting the final output dimension as 2 for better visualization. First, we plot the learned features and proxies with 10, 5, and 2 observed classes respectively. We train the model for 50 epochs. The results are shown in Fig. 3, where the arrows show the learned proxies and the white ones show proxies of missing classes. We scale the norm of proxies by 10x. With all classes observed, we can find that the features are compact and the proxies are accurate. With only 5 observed classes, the regions of features become less compact and the proxies of missing classes are nearly zero. With only 2 observed classes, the proxies of missing classes are only forced to be away from the feature regions. Not so rigorously, proxies of missing classes are updated towards the negative direction of existing samples' center.

Then, we visualize the learned features and proxies in a label distribution shift scene with only 2 clients. The first client owns samples from class set $\{0, 1, 2, 3, 4\}$ while the second one has samples from $\{5, 6, 7, 8, 9\}$. We take 200 global rounds and 2 local epochs in each round. We plot the learned features and proxies at the 5-th round in Fig. 4. The two rows compare FedAvg (FedRS with $\alpha = 1.0$) and FedRS ($\alpha = 0.5$) on the first client. The first column shows the beginning of local procedure, i.e., the extracted features via the newly-downloaded global model and its proxies. The second column shows the local tuned results with only local 5 classes. We can obviously observe that proxies of missing classes are suppressed towards zero in FedAvg, while FedRS can alleviate this phenomenon. Correspondingly, the aggregated proxies become inaccurate and the extracted features become less compact in FedAvg, while FedRS can get more discriminative proxies and the features are more compact as shown in the last column. This directly shows the problems of FedAvg faced with label distribution non-iid data and verifies the superiorities of FedRS. The visualization at the 200-th round is provided in Appendix.

5.2 Performance Comparisons

Basic Settings We then compare the performances based on Cifar10/Cifar100. We construct label distribution shift scenes via label partitions as done in several previous works [33, 55, 57]. Specifically, we decentralize the data onto 100 clients with each client only has a subset of classes. We construct three scenes: Cifar10-100-5, Cifar10-100-2, and Cifar100-100-20. Take Cifar10-100-5 as an example, we split the samples of each class into 50 shards and obtain $10 \times 50 = 500$ shards in all, then we randomly allocate 5 shards to each client. Hence, each client contains 5 classes on average and 100 samples for each class. We take 1000 global rounds, a batch size of 64, and a weight decay of $5e-4$. We use SGD with momentum 0.9 as the optimizer, and use a constant learning rate of 0.03. In each global round, we randomly select 10% clients. We report the accuracy of aggregated model on the global test set, i.e., the test partition of Cifar10/Cifar100. We mainly compare FedRS under different settings of α, i.e., $\{0.0, 0.1, 0.5, 0.9, 1.0\}$. For $\alpha = 1.0$, it degenerates into FedAvg. We also investigate the impact of local epochs in each round, i.e., $\{1, 2, 3, 5\}$. We investigate the performances with different backbones, including TFCNN (Tensorflow CNN)[1] used in FedAvg [38] and VGG11 without BN in PyTorch[2]. The results with VGG11 backbone are presented in Appendix.

Results and Analysis The convergence curves based on TFCNN are plotted in Fig. 5. The three rows correspond to the three scenes and the columns vary in the local epochs. We can observe that

[1]https://www.tensorflow.org/tutorials/images/cnn
[2]https://pytorch.org/docs/stable/torchvision/models.html

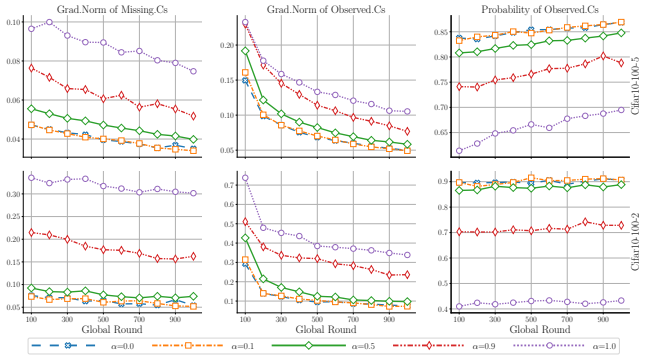

Figure 6: Statistics during optimization in two scenes. Each row shows a scene. The three columns show "gradients of missing classes' proxies", "gradients of observed classes' proxies", and "probability of observed classes" respectively.

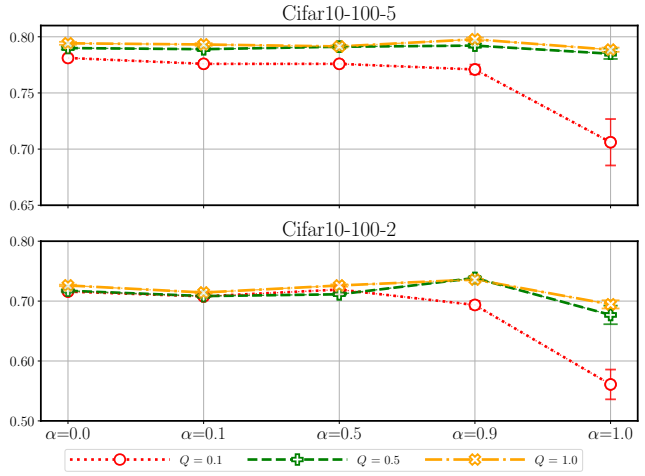

Figure 7: Performance comparisons with different client selection ratios (Q). The top and bottom shows two scenes. Each line corresponds to a setting of selection ratio. The x-axis shows the settings of α.

FedAvg (FedRS with $\alpha = 1.0$) converges slower and fluctuates a lot. FedRS with $\alpha = 0.5$ can almost obtain the best performances. Taking a closer look at FedRS with $\alpha = 0.9$ and $\alpha = 0.0$, we can find setting $\alpha = 0.9$ can obtain better performances than FedAvg, while it also oscillates slightly. In most cases, FedRS with $\alpha = 0.0$ can get comparable results with $\alpha = 0.5$, while it could become worse with more local epochs. From another aspect, with more local epochs, FedAvg converges earlier but with larger fluctuations. However, FedRS with $\alpha = 0.5$ can always obtain stable improvements.

We then plot some statistics during the training procedure of FedRS with different settings of α. We show the statistics in the Cifar10-100-5 and Cifar10-100-2. The most concerning statistic is the gradients of missing classes' proxies. We calculate norms of these gradients in each client and report the mean value as shown in the first column of Fig. 6 (denoted as "Grad.Norm of Missing.Cs"). We can find that smaller α can indeed lead to gradients with smaller norms, which prevents the missing classes' proxies being updated too much. We also care about the ones of observed classes. We plot them in the second column (denoted as "Grad.Norm of Observed.Cs"). There is a similar phenomenon that smaller α can lead to smaller norms. We explain this via the magnitude of observed classes' probabilities, i.e., $p^k_{i,c}$. Form the third column (denoted as "Probability of Observed.Cs"), we can find that smaller α leads to larger $p^k_{i,c}$. Since the gradients of observed classes' proxies have a term $\mathcal{I}\{y^k_i = c\} - p^k_{i,c}$, larger $p^k_{i,c}$ leads to slower update.

Table 1: Performance comparisons with standard FL algorithms (TFCNN). Columns correspond to three scenes. The average accuracy of the last 50 rounds and the standard deviations are reported.

	C10-100-5	C10-100-2	C100-100-20
FedAvg	71.6+ 1.524	55.9+ 3.376	41.1+ 0.684
FedMMD (0.0001) [52]	72.0+ 1.492	54.7+ 3.047	41.7+ 0.608
FedMMD (0.001) [52]	71.5+ 1.590	53.5+ 3.582	41.3+ 0.595
FedProx (0.0001) [35]	71.4+ 1.609	52.9+ 3.429	41.1+ 0.698
FedProx (0.001) [35]	73.0+ 1.123	55.1+ 3.210	41.2+ 0.644
FD [26]	73.7+ 0.772	64.2+ 1.736	42.1+ 0.227
FLDA [40]	67.4+ 0.180	58.8+ 0.521	37.6+ 0.138
FedAwS [55]	75.4+ 0.578	62.4+ 2.304	44.3+ 0.304
Scaffold [28]	72.3+ 1.338	56.6+ 3.284	41.8+ 0.569
FedRS ($\alpha = 0.5$)	**78.0**+ **0.141**	**70.8**+ **0.203**	**45.7**+ **0.201**
FedRS ($\alpha = 0.9$)	77.2+ 0.338	69.8+ 0.665	45.6+ 0.252
Scaffold [28]/RS ($\alpha = 0.5$)	**78.4**+ **0.176**	**71.5**+ **0.169**	**46.0**+ **0.168**

Studies on Client Selection Ratio We investigate the performances of FedRS with different client selection ratios (denoted as Q). We set $Q \in \{0.1, 0.5, 1.0\}$ and plot the average accuracy of the last 50 rounds and the deviations based on TFCNN in Fig. 7. We find that with smaller $C = 0.1$, the performance degradation from $\alpha < 1.0$ to $\alpha = 1.0$ is especially obvious. Larger client selection ratio can mitigate this gap progressively and make the fluctuations smaller. In real-world applications, due to large amounts of clients or limited transmission, a smaller ratio is required to deal with stragglers. This implies that our method is especially advantageous with smaller client selection ratio as analyzed in the last of Sect. 4.3.

Comparing with Other FL Methods We compare our methods with other FL methods including: FedMMD [52] and FedProx [35] based on regularization; FD [26] based on federated distillation; FLDA [40] based on private-shared model; FedAwS [55] based on spreading out; Scaffold [28] based on momentum and controllable variates. We take 2 local epochs and randomly select 10% clients in each round. Both the average accuracy of the last 50 rounds based on TFCNN and the deviations are listed in Tab. 1. For FedMMD and FedProx, we vary the coefficient of the regularization term in $\{0.0001, 0.001\}$. We only show the results of FedRS with $\alpha = 0.5$ and $\alpha = 0.9$, and we can find that our methods can obtain the best performances (bolded). We also find that our method can be easily combined with other methods, e.g., Scaffold, and we report the performances in the last row, which can further improve the performances. The results based on VGG11 are shown in Appendix.

5.3 Large Scale Datasets

We also investigate the settings of α on large scale FL scenes, including Shakespeare and FEMNIST from LEAF[3] [5]. The Shakespeare

[3]https://leaf.cmu.edu/

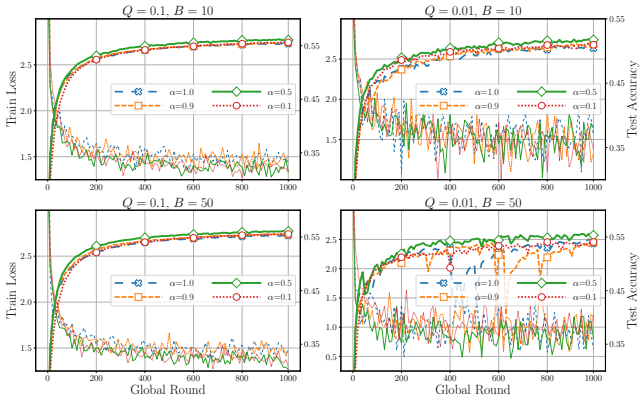

Figure 8: Performance comparisons on Shakespeare scene. Four settings of client selection ratio Q and batch size B are investigated. Each plot shows both training loss and test accuracy.

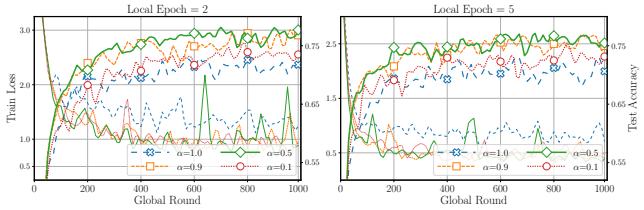

Figure 9: Performance comparisons on FEMNIST scene. Two settings of local epoch are investigated. Each plot shows both training loss and test accuracy.

scene is a next character prediction task, where each speaking role in each play is constructed as an individual client [38]. It contains 1129 clients and each client owns 3743 samples on average. There are 3550 different users in FEMNIST and each user owns 229 samples on average. For both two scenes, We split local data into 80% as local training set on this client, and the other 20% across all clients are combined as a global test set. Although these two scenes are not typical label distribution non-iid scenes, there exists obvious label imbalance on each client. For example, the fraction of classes with a sample size greater than 20 (2) in Shakespeare (FEMNIST) is only 53/81 (25/62). There are 81 and 62 classes in Shakespeare and FEMNIST respectively. Hence, for each local client, we view classes with samples less than 20 (2) in Shakespeare (FEMNIST) as missing classes. We train FedRS with different α on these two scenes. We utilize networks used in LEAF [5]. Details and hyper-parameters can be found in Appendix. The results are shown in Fig. 8 and Fig. 9. We vary different settings of client selection ratio Q, batch size B, and local epoch in these two scenes, and we can find that FedRS with $\alpha = 0.5$ can almost obtain the best performances.

5.4 Real-World Application

We finally compare FedRS with other FL methods on a real-world service awareness application, which is denoted as SA. This task aims to identify which APP is used through network byte streams. The total number of APPs is 32. There are 89 local clients. Each client only has samples from 17.5 classes on average. We utilize a CNN as the classification model. More details of SA and the network

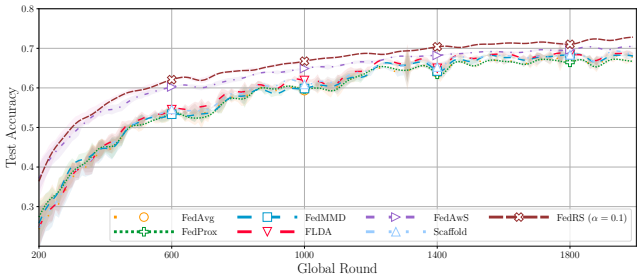

Figure 10: Performance comparisons on SA scene.

can be found in Appendix. We take 2000 rounds, and report the test accuracy on a global test set every 10 rounds. Due to this is a real-world application, we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to each uploaded parameter individually. This can lead to stricter privacy protection, and the detail can be found in Appendix. We take $\sigma = 0.1$ and report the accuracy curves in Fig. 10. We can find that FedRS with $\alpha = 0.1$ can obtain the best performances.

6 CONCLUSION

We study the label distribution non-iid challenge in FL and in-depth analyze the most vulnerable layer, i.e., softmax classification layer in deep networks. We advocate the classification weights of missing classes should be updated carefully during local procedure. We propose Restricted Softmax and FedRS to obtain a more accurate aggregation. Abundant experimental studies verify the superiorities of our methods.

7 ACKNOWLEDGEMENTS

This research was supported by National Natural Science Foundation of China (Grant Nos. 61773198, 61632004 and 61921006), and NSFC-NRF Joint Research Project under Grant 61861146001. Professor De-Chuan Zhan is the corresponding author.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. 308–318.

[2] Atish Agarwala, Jeffrey Pennington, Yann N. Dauphin, and Samuel S. Schoenholz. 2020. Temperature check: theory and practice for training models with softmax-cross-entropy losses. *CoRR* abs/2010.07344 (2020).

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated Learning with Personalization Layers. *CoRR* abs/1912.00818 (2019).

[4] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. 2018. Personalized and Private Peer-to-Peer Machine Learning. In *International Conference on Artificial Intelligence and Statistics*, Vol. 84. 473–481.

[5] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. LEAF: A Benchmark for Federated Settings. *CoRR* abs/1812.01097 (2018).

[6] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. 2019. Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss. In *Advances in Neural Information Processing Systems 32*. 1565–1576.

[7] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. 2018. Partial Transfer Learning With Selective Adversarial Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2724–2732.

[8] Binghui Chen, Weihong Deng, and Haifeng Shen. 2018. Virtual Class Enhanced Discriminative Embedding Learning. In *Advances in Neural Information Processing Systems 31*. 1946–1956.

[9] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. 2018. Federated Meta-Learning for Recommendation. *CoRR* abs/1802.07876 (2018).

[10] Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays, and Michael Riley. 2019. Federated Learning of N-Gram Language Models. In *Proceedings of the 23rd Conference on Computational Natural Language Learning*. 121–130.

[11] Luca Corinzia and Joachim M. Buhmann. 2019. Variational Federated Multi-Task Learning. *CoRR* abs/1906.06268 (2019).

[12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew W. Senior, Paul A. Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *Advances in Neural Information Processing Systems 25*. 1232–1240.

[13] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. 2020. Personalized Federated Learning with Moreau Envelopes. In *Advances in Neural Information Processing Systems 33*.

[14] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.

[15] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Advances in Neural Information Processing Systems 33*.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*. 1126–1135.

[17] Robin C. Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. *CoRR* abs/1712.07557 (2017).

[18] Ross B. Girshick. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision*. 1440–1448.

[19] Samantha Guerriero, Barbara Caputo, and Thomas Mensink. 2018. DeepNCM: Deep Nearest Class Mean Classifiers. In *6th International Conference on Learning Representations*.

[20] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. 2020. FedBoost: A Communication-Efficient Algorithm for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning*. 3973–3983.

[21] Munawar Hayat, Salman H. Khan, Waqas Zamir, Jianbing Shen, and Ling Shao. 2019. Max-margin Class Imbalanced Learning with Gaussian Affinity. *CoRR* abs/1901.07711 (2019).

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[23] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a Unified Classifier Incrementally via Rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition*. 831–839.

[24] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. 2020. Rethinking Class-Balanced Methods for Long-Tailed Visual Recognition From a Domain Adaptation Perspective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7607–7616.

[25] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition?. In *IEEE 12th International Conference on Computer Vision*. 2146–2153.

[26] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data. *CoRR* abs/1811.11479 (2018).

[27] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and Open Problems in Federated Learning. *CoRR* abs/1912.04977 (2019).

[28] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning*. 5132–5143.

[29] Alex Krizhevsky. 2012. Learning Multiple Layers of Features from Tiny Images. (2012).

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*. 1106–1114.

[31] Yann LeCun. 1998. The mnist database of handwritten digits. (1998). http://yann.lecun.com/exdb/mnist/

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[33] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. *CoRR* abs/1910.03581 (2019).

[34] Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola. 2013. Parameter server for distributed machine learning. In *Big Learning NeurIPS Workshop*, Vol. 6. 2.

[35] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems*.

[36] Paul Pu Liang, Terrance Liu, Ziyin Liu, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think Locally, Act Globally: Federated Learning with Local and Global Representations. *CoRR* abs/2001.01523 (2020).

[37] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S. Yu. 2013. Transfer Feature Learning with Joint Distribution Adaptation. In *IEEE International Conference on Computer Vision*. 2200–2207.

[38] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 1273–1282.

[39] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. 2017. No Fuss Distance Metric Learning Using Proxies. In *IEEE International Conference on Computer Vision*. 360–368.

[40] Daniel Peterson, Pallika Kanani, and Virendra J. Marathe. 2019. Private Federated Learning with Domain Adaptation. *CoRR* abs/1912.06733 (2019).

[41] Zhenyue Qin and Dongwoo Kim. 2019. Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator. *CoRR* abs/1911.10688 (2019).

[42] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil Lawrence. 2009. Dataset Shift in Machine Learning. (01 2009).

[43] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji Prediction in a Mobile Keyboard. *CoRR* abs/1906.04329 (2019).

[44] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. 2019. Overcoming Forgetting in Federated Learning on Non-IID Data. *CoRR* abs/1910.07796 (2019).

[45] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. 2017. Federated Multi-Task Learning. In *Advances in Neural Information Processing Systems 30*. 4424–4434.

[46] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems 30*. 4077–4087.

[47] Shuhan Tan, Xingchao Peng, and Kate Saenko. 2019. Generalized Domain Adaptation with Covariate and Label Shift CO-ALignment. *CoRR* abs/1910.10320 (2019).

[48] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[49] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. 2005. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems 18*. 1473–1480.

[50] Xi-Zhu Wu, Song Liu, and Zhi-Hua Zhou. 2019. Heterogeneous Model Reuse via Optimizing Multiparty Multiclass Margin. In *Proceedings of the 36th International Conference on Machine Learning*. 6840–6849.

[51] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 12:1–12:19.

[52] Xin Yao, Chaofeng Huang, and Lifeng Sun. 2018. Two-Stream Federated Learning: Reduce the Communication Costs. In *IEEE Visual Communications and Image Processing*. 1–4.

[53] Xin Yao, Tianchi Huang, Rui-Xiao Zhang, Ruiyu Li, and Lifeng Sun. 2019. Federated Learning with Unbiased Gradient Aggregation and Controllable Meta Updating. *CoRR* abs/1910.08234 (2019).

[54] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in Neural Information Processing Systems 27*. 3320–3328.

[55] Felix X. Yu, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. 2020. Federated Learning with Only Positive Labels. In *Proceedings of the 37th International Conference on Machine Learning*. 10946–10956.

[56] Andrew Zhai and Hao-Yu Wu. 2018. Making Classification Competitive for Deep Metric Learning. *CoRR* abs/1811.12649 (2018).

[57] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated Learning with Non-IID Data. *CoRR* abs/1806.00582 (2018).

A EXPERIMENTAL DETAILS

We first report the details of the investigated scenes from several aspects: the number of clients K, the total number of classes C, the observed classes of each client on average Avg.$|\mathcal{O}^k|$, the total training samples of each client on average Avg.$|N_k|$, and the number of global test samples Test.N. The details are shown in Tab. 2. The Mnist scene is used for visualization and others are used for performance comparison. For the real-world SA dataset, we plot the class distributions of the 89 clients in Fig. 11. We then report the default hyper-parameters of these scenes including: number of global rounds R, number of local epochs E, client selection ratio Q, learning rate η, momentum μ, batch size B, and the network. Without additional declaration, we use these as default and list them in Tab. 3. Finally, we report the details of utilized networks including: the total number of parameters in feature extractor (Num.Ps.Feat), the dimension of features d, and the total number of layers L (we do not count the pooling or activation layers owing to they do not contain parameters). These are shown in Tab. 4.

B STRICTER PRIVACY PROTECTION IN SA

We resort to differential privacy [1, 14, 48] (DP) for stricter privacy protection in SA. Formally, the (ϵ, δ)-DP is defined as:

Definition B.1 ((ϵ, δ)-DP [14]). A randomized mechanism $\mathcal{M}: \mathcal{X} \to \mathcal{R}$ with domain $\mathcal{X}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP, if for all measurable sets $\mathcal{S} \subseteq \mathcal{R}$ and for any two **adjacent** datasets $\mathcal{D}_i, \mathcal{D}_i' \in \mathcal{X}$,

$$Pr\left[\mathcal{M}(\mathcal{D}_i) \in \mathcal{S}\right] \leq e^{\epsilon} Pr\left[\mathcal{M}(\mathcal{D}_i') \in \mathcal{S}\right] + \delta. \tag{13}$$

The $\delta > 0$ is a relaxation term that allows a smaller probability of privacy protection in some cases. With an arbitrarily given δ, a larger ϵ gives a clearer distinguishability of **adjacent** datasets and hence a higher risk of privacy violation. The (ϵ, δ)-DP can be guaranteed via adding Gaussian noise, e.g., $\mathcal{N}(0, \sigma^2)$. The noise scale should satisfy $\sigma \geq c\Delta s/\epsilon$, where c is a constant that should satisfy $c \geq \sqrt{2\ln(1.25/\delta)}$, and Δs is the **sensitivity** of the function s given by $\Delta s = \max_{\mathcal{D}_i, \mathcal{D}_i'} \|s(\mathcal{D}_i) - s(\mathcal{D}_i')\|$. In FL, the sensitivity of uplink is $\Delta s = \frac{2L}{m}$ [48], where L is the maximum norm of the uploaded parameters and m is the minimum number of local samples. We clip the norm of the uploaded parameters via $\Delta\psi = \min(L, \|\Delta\psi\|)\frac{\Delta\psi}{\|\Delta\psi\|}$, where $\psi = (\theta, \{\mathbf{w}_c\}_{c=1}^{C})$ denotes the full set of parameters, and $\Delta\psi$ is the model update. We take $L = 20$ and $m = 400$ in SA. We set $\delta = 0.01$ and add the noise with scale $\sigma = 0.1$. Hence, we can theoretically obtain a $(3, 0.01)$-DP in SA.

Table 2: Details of the investigated scenes. Each row shows a scene.

	K	C	Avg.$\|\mathcal{O}^k\|$	Avg.$\|N_k\|$	Test.N
Mnist	2	10	5	27.5k	10k
C10-100-5	100	10	5	500	10k
C10-100-2	100	10	2	500	10k
C100-100-20	100	100	20	500	10k
Shakespeare	1129	81	53	2994	845k
FEMNIST	3550	62	25	181	161k
SA	89	32	17.5	576	18k

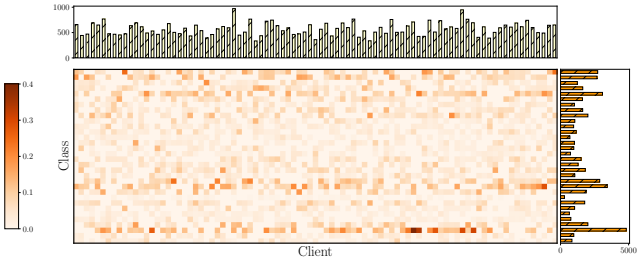

Figure 11: The label distributions in SA. Each column in the middle shows the label distribution of a single client. There are 89 clients and 32 classes in total. The top shows the number of samples in each client, and the right shows the number of samples in each class.

Table 3: Default hyper-parameters for each scene.

	R	E	Q	η	μ	B	Network
Mnist	200	2	1.0	0.1	0.9	64	LeNet
C10-100-5	1000	2	0.1	0.03	0.9	64	TF/VGG11
C10-100-2	1000	2	0.1	0.03	0.9	64	TF/VGG11
C100-100-20	1000	2	0.1	0.03	0.9	64	TF/VGG11
Shakespeare	1000	1	0.01	1.47	0.0	10	CharLSTM
FEMNIST	1000	1	0.001	0.004	0.0	10	FeCNN
SA	2000	5	0.1	0.01	0.9	64	SACNN

Table 4: Details of utilized networks.

	Num.Ps.Feat	d	L
LeNet	33,654	2	5
TFCNN (TF)	56,320	1024	4
VGG11	9,220,480	512	9
CharLSTM	799,368	256	4
FeCNN	6,476,672	2048	4
SACNN	15,984	256	7

Algorithm 1 FedRS

ServerProcedure:

for global round $r = 0, 1, 2, \ldots, R$ **do**
  $S_t \leftarrow$ sample $\max(Q \cdot K, 1)$ clients
  for $k \in S_t$ **do**
    $\hat{\psi}_t^k \leftarrow$ ClientProcedure(k, ψ_t)
  end for
  $\psi_{t+1} \leftarrow \sum_{k=1}^{|S_t|} \frac{1}{|S_t|}\hat{\psi}_t^k$
end for

ClientProcedure(k, ψ_t):

$\psi_t^k \leftarrow \psi_t$
for local epoch $e = 1, 2, \ldots, E$ **do**
  for each batch with B samples from $\mathcal{D}^k$ **do**
    Apply restricted softmax as in Eq. 8 and calculate cross entropy loss, update ψ_t using, e.g., SGD with momentum
  end for
end for
Return: the updated model $\hat{\psi}_t^k$

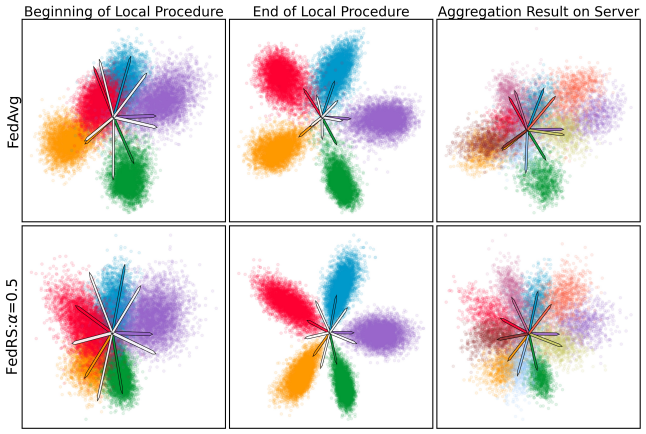

Figure 12: Visualization of learned features and proxies in FL with a 2-client label distribution shift scene (the 200-th round).

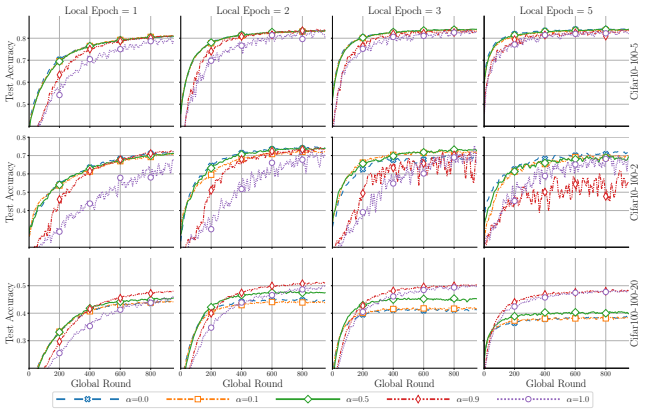

Figure 13: Performance comparisons based on VGG11 with different settings of α in FedRS.

C PSEUDO CODE

We present the pseudo code of FedRS as in Algo. 1.

D MORE EXPERIMENTAL RESULTS

We first present the visualization results based on Mnist at the 200-th global round. The results are shown in Fig. 12, where we can obtain similar results as in Fig. 4. Then we present the performances on the three cifar scenes based on VGG11 as in Fig. 13 and Tab. 5. We can observe that our methods can still obtain better performances than compared methods, while the improvements is slightly weaker than the results based on TFCNN (Tab. 1). *This may be owing to that VGG11 has much more parameters in the feature extractor, and the impact of the final layer is not as large as in TFCNN. Hence, we guess that FedRS could boost performances more with smaller backbones.* Overall, FedRS can lead to better performances.

E PROOF OF THE PROPOSITION

We present the proof of the Proposition. 1. For each selected client $k \in S$, we first download the global parameters, i.e., $\theta^k \leftarrow \theta$,

Table 5: Performance comparisons with standard FL algorithms (VGG11).

	C10-100-5	C10-100-2	C100-100-20
FedAvg	82.0+ 1.104	69.6+ 3.274	49.8+ 0.417
FedMMD (0.0001) [52]	82.6+ 1.005	70.6+ 2.311	50.1+ 0.479
FedMMD (0.001) [52]	82.6+ 0.674	70.4+ 2.993	50.2+ 0.363
FedProx (0.0001) [35]	82.7+ 0.971	68.4+ 2.551	50.4+ 0.439
FedProx (0.001) [35]	82.0+ 0.982	65.8+ 3.100	50.1+ 0.470
FD [26]	82.9+ 0.532	72.2+ 0.482	49.8+ 0.429
FLDA [40]	72.8+ 0.186	57.7+ 0.597	33.6+ 0.199
FedAwS [55]	82.8+ 0.440	71.5+ 2.080	49.7+ 0.242
Scaffold [28]	82.6+ 0.587	71.8+ 2.466	50.1+ 0.357
FedRS ($\alpha = 0.5$)	83.4+ 0.136	**73.9+ 0.182**	47.5+ 0.170
FedRS ($\alpha = 0.9$)	**83.5+ 0.370**	73.1+ 0.794	**51.0+ 0.274**
Scaffold [28]/RS ($\alpha = 0.5$)	**83.8+ 0.285**	73.0+ 0.669	50.6+ 0.191

$\mathbf{w}_c^k \leftarrow \mathbf{w}_c, \forall c \in C$. Then a local training step can be obtained as:

$$\hat{\mathbf{w}}_c^k = \mathbf{w}_c^k + \eta \alpha_c^k \sum_{i=1}^{N_k} (\mathcal{I}\{y_i^k = c\} - p_{i,c}^k)\mathbf{h}_i^k \tag{14}$$

$$= \mathbf{w}_c + \eta \alpha_c^k \sum_{i=1}^{N_k} (\mathcal{I}\{y_i^k = c\} - p_{i,c}^k)\mathbf{h}_i^k, \tag{15}$$

and then the aggregation process is:

$$\begin{aligned}
\mathbf{w}_c &= \frac{1}{|S|}\sum_{k=1}^{|S|} \hat{\mathbf{w}}_c^k \\
&= \frac{1}{|S|}\sum_{k=1}^{|S|}\left(\mathbf{w}_c + \eta\alpha_c^k \sum_{i=1}^{N_k}(\mathcal{I}\{y_i^k = c\} - p_{i,c}^k)\mathbf{h}_i^k\right) \\
&= \mathbf{w}_c + \frac{\eta}{|S|}\sum_{k=1}^{|S|}\left(\alpha_c^k \sum_{i=1}^{N_k}(\mathcal{I}\{y_i^k = c\} - p_{i,c}^k)\mathbf{h}_i^k\right) \\
&= \mathbf{w}_c + \frac{\eta}{|S|}\sum_{k=1}^{|S|}\left(\mathcal{I}\{c \in O^k\}\sum_{i=1}^{N_k}(\mathcal{I}\{y_i^k = c\} - p_{i,c}^k)\mathbf{h}_i^k\right. \\
&\quad \left. -\mathcal{I}\{c \in \mathcal{M}^k\}\alpha \sum_{i=1}^{N_k} p_{i,c}^k \mathbf{h}_i^k\right) \\
&= \mathbf{w}_c + \frac{\eta}{|S|}\sum_{k=1}^{|S|}\left(\mathcal{I}\{c \in O^k\}\sum_{i=1}^{N_k}\left(\mathcal{I}\{y_i^k = c\}(1 - p_{i,c}^k)\mathbf{h}_i^k\right.\right. \\
&\quad \left.\left. -\mathcal{I}\{y_i^k \neq c\}p_{i,c}^k \mathbf{h}_i^k\right) - \mathcal{I}\{c \in \mathcal{M}^k\}\alpha \sum_{i=1}^{N_k} p_{i,c}^k \mathbf{h}_i^k\right) \\
&= \mathbf{w}_c + \underbrace{\frac{\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I_{1,i,c}^k (1 - p_{i,c}^k)\mathbf{h}_i^k}_{\text{part1}} \\
&\quad \underbrace{-\frac{\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I_{2,i,c}^k p_{i,c}^k \mathbf{h}_i^k}_{\text{part2}} \underbrace{-\frac{\alpha\eta}{|S|}\sum_{k=1}^{|S|}\sum_{i=1}^{N_k} I_{3,c}^k p_{i,c}^k \mathbf{h}_i^k}_{\text{part3}}
\end{aligned}$$