# FedGR: Federated Learning with Gravitation Regulation for Double Imbalance Distribution

Songyue Guo[1], Xu Yang[1], Jiyuan Feng[1], Ye Ding[2], Wei Wang[3], Yunqing Feng[4], and Qing Liao[1,5(✉)]

[1] Harbin Institute of Technology, Shenzhen, China
{guosongyue,xuyang97,fengjy}@stu.hit.edu.cn, liaoqing@hit.edu.cn
[2] Dongguan University of Technology, Dongguan, China
dingye@dgut.edu.cn
[3] Huazhong University of Science and Technology, Wuhan, China
weiwangw@hust.edu.cn
[4] Shanghai Pudong Development Bank, Shanghai, China
fengyq5@spdb.com.cn
[5] Pengcheng Laboratory, Shenzhen, China

**Abstract.** Federated Learning (FL) is a well-known framework for distributed machine learning that enables mobile phones and IoT devices to build a shared machine learning model via only transmitting model parameters to preserve sensitive data. However, existing Non-IID FL methods always assume data distribution of clients are under a single imbalance scenario, which is nearly impossible in the real world. In this work, we first investigate the performance of the existing FL methods under double imbalance distribution. Then, we present a novel FL framework, called **Fed**erated Learning with **G**ravitation **R**egulation (FedGR), that can efficiently deal with the double imbalance distribution scenario. Specifically, we design an unbalanced softmax to deal with the quantity imbalance in a client by adjusting the forces of positive and negative features adaptively. Furthermore, we propose a gravitation regularizer to effectively tackle the label imbalance among clients by facilitating collaborations between clients. At the last, extensive experimental results show that FedGR outperforms state-of-the-art methods on CIFAR-10, CIFAR-100, and Fashion-MNIST real-world datasets. Our code is available at https://github.com/Guosy-wxy/FedGR.

**Keywords:** Federated learning · Double imbalance distribution · Non-IID · Softmax · Regularizer term

## 1 Introduction

Despite the success of deep learning in numerous fields [12,22], a data center training model is typically required. In some real-world applications, individual participant data cannot be located on the same device due to data privacy [1].


X. Wang et al. (Eds.): DASFAA 2023, LNCS 13943, pp. 703–718, 2023.
https://doi.org/10.1007/978-3-031-30637-2_47

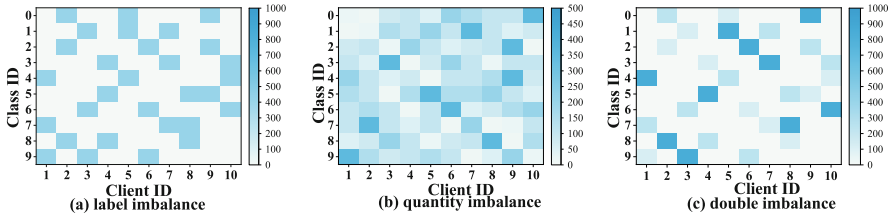


**Fig. 1.** Different imbalance distribution scenarios on CIFAR-10 dataset. (a) label imbalance, (b) quantity imbalance, (c) double imbalance.

Federated Learning (FL) [9,17,26] is designed for data privacy protection and efficient distributed training.

The advent of FL enables different clients to collectively build a robust global model without broadcasting local private data to the server. FL has demonstrated its ability to facilitate real-world applications in several domains, e.g., natural language processing [7], credit card fraud detection [28] and medical healthcare [6,25].

FL, however, also confronts the challenge of imbalance distribution [17,20]. The imbalance distribution of Non-IID data between clients brings serious performance degradation problems for FL [14,27]. This performance degradation is attributed to the phenomenon of client drift. Some recent works aim to deal with this problem, e.g., FedProx [14] included $l_2$ regularizer term to prevent local models from deviating too far from the global model, PerFedAvg [4] utilized contrastive learning, and MOON [13] used multi-task learning for fast client local adaptation to mitigate the impact of client drift. However, most existing studies focus on single imbalance distribution [21].

In this work, we focus on double imbalance distribution scenario, which is more common in the real world. We first define the imbalance distribution into two categories:

1) **Label imbalance.** According to Fig. 1(a), we simulate this scenario with 10 clients on CIFAR-10 dataset. The majority of clients have part labels of the whole, and client's labels are mostly different from others, while the quantity of each label in client is equal. For example, client 1 owns labels 4, 7, 9, but client 2 owns labels 0, 2, 8. Most recent works like FedProx [14], SCAFFOLD [10] and FedNova [23] only care about label imbalance.
2) **Quantity imbalance.** We still use 10 clients on CIFAR-10 to describe this imbalance distribution. Based on Fig. 1(b), each client owns an entire set of labels, but the quantity of each label in client varies, i.e., client 1 has 10 labels, and the number of label 9 is about 450, but the number of label 0 and 1 is approximately 0.

In this study, we focus on *double imbalance distribution* like Fig. 1(c), each client owns a partition of entire labels, and the quantity of each class in client varies, e.g., client 1 only possesses labels 4, 7, 9, and each class' sample number is imbalanced. It is clear that the double imbalance distribution scenario is more

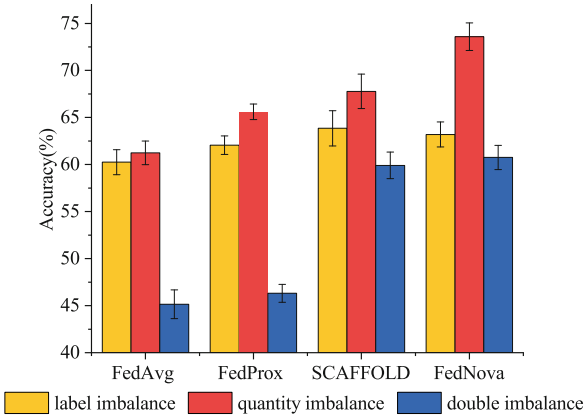


**Fig. 2.** Comparison with existing FL methods under different imbalance distribution.

in line with reality than any single imbalance scenario. However, existing works mostly omit the real scenario of double imbalance distribution scenario.

In order to investigate the performance of existing IID and Non-IID FL algorithms for double imbalance scenario, we use client distribution as Fig. 1 to design an observation experiment. We use TFCNN[1] as client's base model and implement all compared FL methods with the same model for a fair comparison. All performance results are expressed by accuracy of an average of five times. The experiment results are summarized in Fig. 2. From the performance results shown, we can find that no matter what FL algorithms gets a significant performance loss on the double imbalance, compared to the left two scenes. For example, the accuracy of FedAvg declines by about 16%. Apparently, the double imbalance scenario brings a new challenge for existing FL algorithms, which is the goal of our study tries to solve.

Motivated by the above observation experiment of double imbalance distribution, we propose a novel FL algorithm called **Fed**erated Learning with **G**ravitation **R**egulation (FedGR) to deal with this problem. We define a novel softmax function called unbalanced softmax to balance the importance of classes under quantity imbalance in clients. In addition, we propose an efficient gravitation regularizer to deal with label imbalance among clients by encouraging collaboration among clients. Combining these two components, we can correct the gradient of traditional loss function of typical FL methods. The contributions of this paper can be summarized as follows:

- We propose a novel federated learning method FedGR to effectively deal with the performance degradation problem caused by double imbalance distribution scenario.
- We design an unbalanced softmax function, which can solve the problem of unbalanced number of samples within the client by adjusting the forces of positive and negative samples on classes.

[1] https://www.tensorflow.org/tutorials/images/cnn.

- We propose a gravitation regularizer to alleviate the impact of label imbalance between clients by introducing cross-client forces to encourage the collaboration of different clients.
- Extensive experiments show that FedGR significantly outperforms the state-of-the-art federated learning methods on several benchmark datasets under double imbalance distribution scenario.

## 2 Related Work

Recently, federated learning on imbalance data distribution has drawn much interest in machine learning research. Zhao *et al.* [27] shared a limited public dataset across clients to relieve the degree of imbalance between various clients. FedProx [14] introduced a proximal term to limit the dissimilarity between the global model and local models. SCAFFOLD [10] used variance reduction to alleviate the effect of client drifting that causes weight divergence between the local and global models. FedNova [23] changed the aggregation phase by allocating different number of local steps per round to different client participants which have different computational capabilities to eliminate original objective inconsistency problem caused by imbalance data. PerFedAvg [4] used meta-learning to learn a new task quickly and effectively for quick local adaptation. Dinh *et al.* [20] proposed pFedme, introducing $l_2$-norm regularization to PerFedAvg which can control the balance between personalization and generalization performance. Li *et al.* [13] proposed MOON, which applied constrastive learning to make local representation closer to the global model's representation for better performance. Huang *et al.* [8] proposed FedAMP, an attention-based mechanism that enforces stronger pairwise collaboration amongst FL clients with similar data distributions. APFL [2] utilized model interpolation to adaptively control the mixture of global and local model. Astraea [3] created the mediator to reschedule the training of clients based on Kullback-Leibler divergence (KLD) of their data distribution for label imbalance. FedGC [18] introduced a softmax-based regularizer term to correct the loss function to be similar to the standard softmax in conventional central learning. Ghosh *et al.* proposed IFCA [5], a clustering FL framework that has many global models and assign each client to one of the $K$ clusters the global model of which achieves the lowest loss value on the client's data. FedRS [15] proposed a restricted softmax to limit the update of missing classes' parameters during the local procedure.

However, existing methods ignore considering double imbalance distribution scenario, which are not applicable on real complicated FL scenarios.

## 3 FedGR

In this section, we will first explicitly introduce the whole structure of the proposed FedGR. Then, we first define a novel softmax function to deal with quantity imbalance in client. Third, we design a gravitation regularizer in server to deal with label imbalance between clients. At the last we present the algorithm of FedGR.

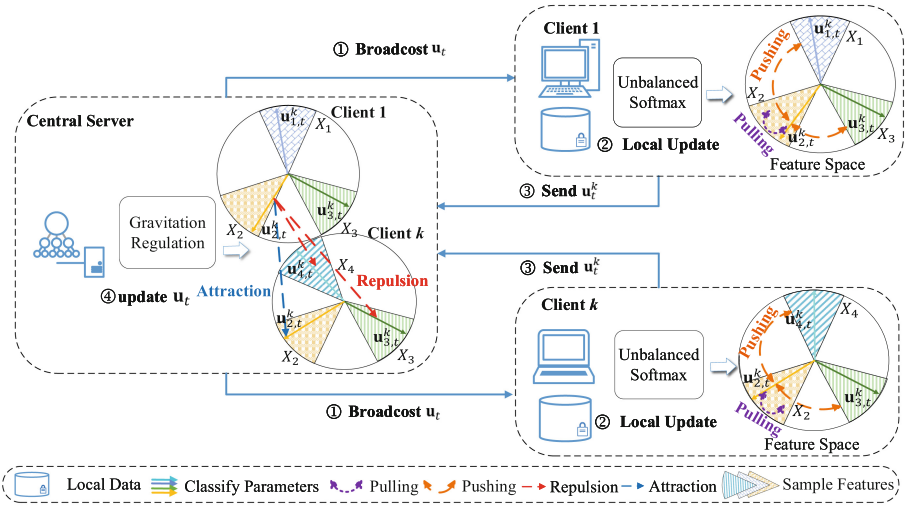


**Fig. 3.** Framework of FedGR. FedGR contains two components: unbalanced softmax in client and gravitation regularizer in server. Unbalanced Softmax aims to deal with the quantity imbalance in client. Graviation Regularizer aims to tackle the label imbalance among clients.

### 3.1 Framework of FedGR

On the basis of the aforementioned observation, we propose a method called **Fed**erated Learning with **G**ravitation **R**egulation (**FedGR**) to address the problem of double imbalance distribution. As shown in Fig. 3, FedGR have two components: clients and central server. In client, we design a novel *Unbalanced Softmax* to address quantity imbalance. Meanwhile, in server, we propose an efficient regularizer term called *Gravitation Regularizer* to solve label imbalance cross clients.

### 3.2 Unbalanced Softmax

In this subsection, we first promote the shortcoming of standard softmax faced with quantity imbalance. Then, we define a simple but efficient softmax function called unbalanced softmax to adjust the importance of classes under quantity imbalance situation. Finally, we analyze the benefits of unbalanced softmax under quantity imbalance scenario.

According to traditional FL algorithms [14,17], the cross-entropy function of client $k$ can be formulated as:

$$\mathcal{L}^k = -\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}^k} \log p^k_{i,y_i}, \tag{1}$$

where $\mathcal{L}^k$ means the loss function of client $k$, $(\mathbf{x}_i, y_i)$ denotes $i$-th sample in training dataset, $\mathbf{x}_i$ is a vector of training data, and $y_i$ is the label of $i$-th sample,

$\mathcal{D}^k$ denotes the local data, $p^k_{i,y_i}$ is the probability of $i$-th samples belonging to $y_i$ class. The probability is typically calculated by the standard softmax function by normalizing each class's score:

$$p^k_{i,y_i} = \frac{e^{\mathbf{u}^{k^T}_{y_i}\mathbf{v}^k_i}}{\sum_{y_j=1}^{C^k} e^{\mathbf{u}^{k\ T}_{y_j}\mathbf{v}^k_i}}, \tag{2}$$

where $\mathbf{u}^k_{y_i}$ denotes the classification parameters of $y_i$ in client $k$, $\mathbf{v}^k_i$ means the extracted feature of $i$-th sample, $C^k$ is the number of labels in client $k$.

However, the standard softmax may not work well when faced with quantity imbalance because standard softmax gives each class the same weight. In fact, some classes possess insufficient data samples, like the tail class in long-tailed distribution. Hence, the classification parameters of head class will pull tail class to an error feature region, so the performance of client's model directly drops.

In order to solve quantity imbalance problem in client, we introduce a balance factor $\gamma$ in unbalanced softmax to balance the importance of different classes. The unbalanced softmax can be formulated as:

$$\hat{p}^k_{i,y_i} = \frac{e^{\gamma^k_{y_i}\mathbf{u}^{k^T}_{y_i}\mathbf{v}^k_i}}{\sum_{y_j=1}^{C^k} e^{\gamma^k_{y_j}\mathbf{u}^{k\ T}_{y_j}\mathbf{v}^k_i}}, \tag{3}$$

where $\gamma^k_{y_i} = \frac{N^k}{N^k_{y_i}}$ denotes as balance factor of each class, $N^k$ means the number of total samples in client $k$, and $N^k_{y_i}$ means the number of label $y_i$ samples. We utilize this balance factor to increase the importance of tail classes in client.

**Theoretical Analysis.** In order to show the efficient of the proposed unbalanced softmax, we analyze the benefits of it for quantity imbalance in clients as below:

First, the loss function of client $k$ in FL can be rewritten by replacing the standard softmax as unbalanced softmax:

$$\mathcal{L}^k = -\sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}^k} \log \hat{p}^k_{i,y_i} = -\sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}^k} \log \frac{e^{\gamma^k_{y_i}\mathbf{u}^{k\ T}_{y_i}\mathbf{v}^k_i}}{\sum_{y_j=1}^{C^k} e^{\gamma^k_{y_j}\mathbf{u}^{k\ T}_{y_j}\mathbf{v}^k_i}}. \tag{4}$$

Second, the computation of the gradient of $\frac{\partial\mathcal{L}^k}{\partial\mathbf{u}^k_{y_i}}$ is formulated as below:

$$\frac{\partial\mathcal{L}^k}{\partial\mathbf{u}_{y_i}} = -\sum_{i=1,y\neq y_i}^{N} \gamma^k_{y_i}\hat{p}^k_{i,y_i}\mathbf{v}^k_i + \sum_{i=1,y=y_i}^{N} \gamma^k_{y_i}\left(1-\hat{p}^k_{i,y_i}\right)\mathbf{v}^k_i. \tag{5}$$

Final, we use gradient descent with learning rate $\eta$ to update the classification parameters of label $y_i$ and decompose this update process into the pushing and pulling forces:

$$\mathbf{u}^k_{y_i} = \mathbf{u}^k_{y_i} \underbrace{-\eta \sum_{i=1,y\neq y_i}^{N} \gamma^k_y\hat{p}^k_{i,y_i}\mathbf{v}^k_i}_{\text{weighted pushing force}} + \underbrace{\eta \sum_{i=1,y=y_i}^{N} \gamma^k_{y_i}\left(1-\hat{p}^k_{i,y_i}\right)\mathbf{v}^k_i}_{\text{weighted pulling force}}, \tag{6}$$

where the pulling force come from positive samples which have the same label $y_i$, while the pushing force is from negative samples whose labels are not $y_i$. The pulling force aims to pull the classification parameters close to the feature region of positive samples, while the pushing force aims to pushing the classification parameters far away from the feature region of negative samples.

From Eq. (6), it is obvious that pushing force and pulling force are weighted by our balance factor $\gamma_{y_i}^k$. If $y_i$ is a tail class in client $k$, $\gamma_{y_i}^k$ will be larger. Consequently, pushing force and pulling force are more efficient, bringing the classification parameters of tail class closer to their feature regions.

### 3.3 Gravitation Regularizer

In this subsection, we first analyze the drawbacks of traditional global optimization objective in FL when faced with label imbalance. Then, we define a novel regularizer called gravitation regularizer to encourage the collaboration of clients under label imbalance situation. Final, we analyze the benefits of gravitation regularizer under label imbalance scenario.

In typical FL scenario, traditional FL considers to train a global model by the following optimization objective:

$$\min_{\mathbf{u}} F(\mathbf{u}) \triangleq \sum_{k=1}^{K} \alpha^k \mathcal{L}^k, \tag{7}$$

where $K$ is the number of clients, $\alpha^k$ is the aggregation weight of client $k$. We define $\alpha^k$ as $\frac{N^k}{N}$, where $N$ is the number of total data samples. Then, $\sum_{k=1}^{K} \alpha^k = 1$. We denote the distributed optimization objective as *global optimization*.

However, the typical optimization objective in FL might cause performance decrease when it only considers optimization within the client and ignores optimization cross clients. For instance, client $k_1$ has label 1, 2, 3 and client $k_2$ has label 2, 3, 4. In the client $k_1$, the label 1 should be pushed far away from the label 2 and label 3. On the other hand, the label 4 should be also pushed far away from the label 2 and label 3 in client $k_2$. Hence, the feature space of label 1 might incorrectly overlap to the feature space of label 4 if the objective function ignores the cross-client optimization.

In order to introduce cross-client optimization into the objective function for handling label imbalance, we design a new regularizer term of FedGR:

$$\min_{\mathbf{u}} F(\mathbf{u}) \triangleq \sum_{k=1}^{K} \alpha^k \mathcal{L}^k + \lambda \cdot \text{Gravitation-Reg}(\mathbf{u}), \tag{8}$$

where $\lambda$ is a hyper-parameter to control the weight of the gravitation regularization term.

The gravitation regularizer term contains two components: Attraction regularizer and Repulsion regularizer, which are defined as follows:

$$\text{Gravitation-Reg}(\mathbf{u}) = -\sum_{k=1}^{K}\sum_{y_i=0}^{C^k}\Big(\underbrace{\log\frac{\sum_{z\neq k} e^{\mathbf{u}_{y_i}^{z^T}\mathbf{u}_{y_i}^{k'}}}{\sum_{z\neq k}\sum_{y_j=1}^{C^z} e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^{k'}}}}_{\text{Attraction}} + \underbrace{\log\frac{e^{\mathbf{u}_{y_i}^{k^{T\prime}}\mathbf{u}_{y_i}^{k'}}}{e^{\mathbf{u}_{y_i}^{k^{T\prime}}\mathbf{u}_{y_i}^{k'}} + \sum_{z\neq k}\sum_{y_j=1,j\neq i}^{C^z} e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^{k'}}}}_{\text{Repulsion}}\Big), \tag{9}$$

where $\mathbf{u}_{y_i}^{k'}, \mathbf{u}_{y_i}^{k^{T\prime}}$ suggests the gradient is set to be zero, which means the gradient is not required for these vectors. Attraction regularizer aims to increase similarity among the same labels' classification parameters across clients, while Repulsion regularizer aims to decrease similarity among different labels' classification parameters across clients. Hence, the Gravitation Regularizer can correct the gradient of the optimization loss function with Attraction regularizer and Repulsion regularizer.

**Theoretical Analysis.** In order to show how gravitation regularizer works, we will theoretically discuss the effort of it for label imbalance among clients as follows:

According to Eq. (8), optimization objective of FedGR equals the empirical risk respect to the loss function $\mathcal{L}_G^k$:

$$\begin{aligned} F(\mathbf{u}) &= \sum_{k=1}^{K}\alpha^k\mathcal{L}^k + \lambda\cdot\text{Gravitation-Reg}(\mathbf{u}) \\ &= -\frac{1}{N}\sum_{k=1}^{K}\sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}^k}\left(\hat{p}_{i,y_i}^k + \lambda\cdot\text{Attraction}(\mathbf{u}) + \lambda\cdot\text{Repulsion}(\mathbf{u})\right) \\ &= -\frac{1}{N}\sum_{k=1}^{K}\mathcal{L}_G^k. \end{aligned} \tag{10}$$

To show that the added gravitation regularizer complement ignored cross-client optimization, we correspondingly calculate the gradient of $\mathcal{L}_G^k$ to cross-client classification parameters $\frac{\partial\mathcal{L}_G^k}{\partial\mathbf{u}_{y_i}^z}$ and $\frac{\partial\mathcal{L}_G^k}{\partial\mathbf{u}_{y_j}^z}$ of FedGR. Then, the related gradient can be calculated as:

$$\frac{\partial\mathcal{L}_G^k}{\partial\mathbf{u}_{y_i}^z} = \left(\frac{\sum_{z\neq k} e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^k}}{\sum_{z\neq k}\sum_{y_j=1}^{C^z} e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^k}} - 1\right)\mathbf{u}_{y_i}^k, \tag{11}$$

$$\frac{\partial \mathcal{L}_G^k}{\partial \mathbf{u}_{y_j}^z} = \frac{e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^k}}{e^{\mathbf{u}_{y_i}^{k^T}\mathbf{u}_{y_i}^k} + \sum_{z\neq k}\sum_{y_j=1,j\neq i}^{C^z} e^{\mathbf{u}_{y_j}^{T}\mathbf{u}_{y_i}^k}} \mathbf{u}_{y_i}^k, (j \neq i) \tag{12}$$

Due to the ease of convergence on local data, the features of local client data are effectively trained on each client. Therefore, the distance between classification parameters $\mathbf{u}_{y_i}^k$ and features $\mathbf{v}_i^k$ tends to be zero: $\mathbf{u}_{y_i}^k \to \mathbf{v}_i^k$. Hence, we can get the following approximations:

$$\frac{\partial \mathcal{L}_G^k}{\partial \mathbf{u}_{y_i}^z} \approx \left( \frac{\sum_{z\neq k} e^{\mathbf{u}_j^{z^T}\mathbf{u}_{y_i}^k}}{\sum_{z\neq k}\sum_{j=1}^{C^z} e^{\mathbf{u}_j^{z^T}\mathbf{u}_{y_i}^k}} - 1 \right) \mathbf{v}_i^k, \tag{13}$$

$$\frac{\partial \mathcal{L}_G^k}{\partial \mathbf{u}_{y_j}^z} \approx \frac{e^{\mathbf{u}_{y_j}^{z^T}\mathbf{u}_{y_i}^k}}{e^{\mathbf{u}_{y_i}^{k^T}\mathbf{u}_{y_i}^k} + \sum_{z\neq k}\sum_{y_j=1,j\neq i}^{C^z} e^{\mathbf{u}_{y_j}^{T}\mathbf{u}_{y_i}^k}} \mathbf{v}_i^k, (j \neq i). \tag{14}$$

According to Eq. (6), Eq. (13) and Eq. (14), the gradient of $\mathcal{L}_G^k$ to $\mathbf{u}_{y_i}^z$ and $\mathbf{u}_{y_j}^z$ are similar to the pulling and pushing force of positive and negative samples respectively. Different with Eq. (6) which only considers the pulling and pushing force in client, the Eq. (13) and Eq. (14) introduce attraction and repulsion force cross client.

Finally, the updated classification parameters of $y_i$ can be decomposed as in-client force and cross-client gravitation force based on Eq. (6), Eq. (13) and Eq. (14):

$$\begin{aligned}
\mathbf{u}_{y_i,t+1}^z = \mathbf{u}_{y_i,t}^z & \underbrace{-\eta \sum_{i=1,y\neq y_i}^{N_z} \gamma_{y_i}^z \hat{p}_{i,y_i}^z \mathbf{v}_{i,t}^z}_{\text{in-client pushing force}} \underbrace{+\eta \sum_{i=1,y=y_i}^{N_z} \gamma_{y_i}^z \left(1-\hat{p}_{i,y_i}^z\right) \mathbf{v}_{i,t}^z}_{\text{in-client pulling force}} \\
& \underbrace{-\eta\lambda \sum_{k=1,k\neq z}^{K} \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}^k} \left( \frac{\sum_{z\neq k} e^{\mathbf{u}_{j,t}^{z^T}\mathbf{u}_{y_i,t}^k}}{\sum_{z\neq k}\sum_{j=1}^{C^z} e^{\mathbf{u}_{j,t}^{z^T}\mathbf{u}_{y_i,t}^k}} - 1 \right) \mathbf{v}_{i,t}^k}_{\text{cross-client Attraction force}} \\
& \underbrace{+\eta\lambda \sum_{k=1,k\neq z}^{K} \sum_{(\mathbf{x}_i,y_i)\in\mathcal{D}^k} \frac{e^{\mathbf{u}_{j,t}^{z^T}\mathbf{u}_{y_i,t}^k}}{e^{\mathbf{u}_{y_i,t}^{k^T}\mathbf{u}_{y_i,t}^k} + \sum_{z\neq k}\sum_{j=1,j\neq i}^{C^z} e^{\mathbf{u}_{j,t}^{T}\mathbf{u}_{y_i,t}^k}} \mathbf{v}_{i,t}^k}_{\text{cross-client Repulsion force}} .
\end{aligned} \tag{15}$$

where $\eta$ is the learning rate of gradient descent, $\lambda$ is the weight of regularizer term. Related forces (in-client forces and cross-client forces) can be all formed like $\beta \mathbf{v}_i^k$.

**Algorithm 1: FedGR**

**Input:** Number of clients $K$; total communication rounds $T$; learning rate $\eta$; participate rate $Q$, regularizer weight $\lambda$; dataset of client $k$ $\mathcal{D}^k$.
**Output:** model parameters $\mathbf{u}_T^k$.

1 Server initialized $\mathbf{u}_0$;
2 **for** *round* $t = 0, \ldots, T-1$ **do**
3  Server select a subset of clients $\mathcal{S}(t) \leftarrow Q \cdot K$;
  `// Clients Update:`
4  **foreach** *participate client* $k \in \mathcal{S}(t)$ **do**
5  download parameters from server $\mathbf{u}_t^k \leftarrow \mathbf{u}_t$;
6  update parameters $\mathbf{u}_{t+1}^k \leftarrow \mathbf{u}_t^k - \eta \frac{\partial \mathcal{L}^k}{\partial \mathbf{u}_{y_i}}$ (Eq. 6);
7  send $\mathbf{u}_{t+1}^k$ to central server;
8  **end**
  `// Server Update:`
9  aggragate the parameters $\tilde{\mathbf{u}}_{t+1} = \left[\mathbf{u}_{t+1}^k, \ldots, \mathbf{u}_{t+1}^K\right]^T$;
10  update parameters $\mathbf{u}_{t+1} \leftarrow \tilde{\mathbf{u}}_{t+1} - \lambda\eta\nabla_{\tilde{\mathbf{u}}_{t+1}} \text{Gravitation-Reg}\left(\tilde{\mathbf{u}}_{t+1}\right)$ (Eq.15);
11 **end**

### 3.4 Training Process of FedGR

The overall process of the proposed FedGR algorithm is illustrated in Algorithm 1. There are two main processes: clients update (line 4–7) and server update (line 9–10). In the process of clients update, each client starts their local-update process on their datasets $\mathcal{D}^k$ in parallel. First, clients download the parameters $\mathbf{u}_t$ broadcast by central server in line 5. Then they update model parameters $\mathbf{u}_t^k$ by unbalanced softmax to obtain $\mathbf{u}_{t+1}^k$ from line 6. At last, client $k$ transfers updated parameters $\mathbf{u}_{t+1}^k$ to the central server in line 7. In the process of server update, server updates the global model parameters $\mathbf{u}_{t+1}$ via gravitation regularizer in line 9–10.

## 4 Experiments

In this section, we first introduce basic experiment settings. Second, we show the ability of FedGR to deal with double imbalance distribution on several benchmark datasets, compared with start-of-the-art FL algorithms. Third, we make a ablation study of each component in FedGR. Fourth, we analyze the selection of hyper-parameter $\lambda$. Finally, we show the visualization comparisons conducted on feature level.

### 4.1 Experimental Setup

**Datasets.** We conduct experiments on three real-world image datasets: CIFAR-10, CIFAR-100 and Fashion-MNIST. The information of all datasets is listed in Table 1.

**Table 1.** Details of CIFAR-10, CIFAR-100 and Fashion-MNIST

| Datasets | Class Number | Image Size | Training Samples | Test Samples |
|---|---|---|---|---|
| CIFAR-10 [11] | 10 | 32×32 | 50,000 | 10,000 |
| CIFAR100 [11] | 100 | 32×32 | 50,000 | 10,000 |
| Fashion-MNIST [24] | 10 | 28×28 | 50,000 | 10,000 |

**Table 2.** Performance compared with state-of-the-art algorithms on CIFAR-10/100 dataset under double imbalance distribution. The best results are in **bold**, and the secondary optimal results are mark as underline. CIFAR-10 (2) means each client owns two labels, which is similar to CIFAR-10 (3), CIFAR-100 (20) and CIFAR-100 (30).

| Algorithms | CIFAR-10 (2) | | CIFAR-10 (3) | | CIFAR-100 (20) | | CIFAR-100 (30) | |
|---|---|---|---|---|---|---|---|---|
| | Acc(%) | F1(%) | Acc(%) | F1(%) | Acc(%) | F1(%) | Acc(%) | F1(%) |
| FedAvg [17] | 50.36 | 48.27 | 53.79 | 49.42 | 36.15 | 34.10 | 42.19 | 40.42 |
| FedProx [14] | 48.84 | 46.96 | 54.94 | 53.85 | 36.24 | 34.42 | 42.21 | 41.09 |
| FedNova [23] | 56.33 | 54.59 | 68.63 | 66.09 | 38.63 | 37.72 | 45.35 | 45.59 |
| SCAFFOLD [10] | 57.37 | 54.43 | 67.32 | 62.44 | 38.43 | 37.76 | 46.82 | 45.44 |
| PerFedAvg [4] | 44.67 | 42.56 | 54.87 | 53.73 | 35.98 | 34.76 | 40.14 | 40.33 |
| pFedMe [20] | 45.81 | 44.35 | 50.18 | 50.24 | 35.36 | 33.59 | 40.18 | 40.54 |
| FedOpt [19] | 62.37 | 60.68 | 70.63 | 69.79 | 42.37 | 40.68 | 49.63 | 49.79 |
| MOON [13] | 61.45 | 60.71 | 72.91 | 70.45 | 40.53 | 41.46 | 47.91 | 48.76 |
| FedRS [15] | 63.22 | 60.13 | 73.56 | 70.13 | 42.76 | 42.21 | 50.73 | 50.31 |
| FedGC [18] | 62.91 | 60.35 | 72.11 | 70.64 | 42.11 | 40.35 | 50.21 | 50.46 |
| FedGR (ours) | **67.84** | **65.62** | **77.86** | **75.32** | **45.44** | **44.85** | **53.16** | **53.32** |
| | (4.53↑) | (4.91↑) | (4.3↑) | (4.68↑) | (2.68↑) | (2.64↑) | (2.43↑) | (2.86↑) |

**Data Segmentation.** According to related works, we first followed [13] to reshape the original balanced datasets to a quantity imbalance distribution, which means the number of samples of each label on the client side follows a power-law distribution. After that, we followed [14,15,17] to simulate the label imbalance distribution by giving each client a fixed number of labels.

**Models and Hardware Settings.** We use TFCNN for CIFAR-10/100 and Fashion-MNIST as the base model. All experiments are run by PyTorch on two NVIDIA GeForce V100 GPUs. By default, we run 1000 communication rounds. We set the number of total clients at 100 and a client participate ratio 10 % in each round. For local optimization, we set the batch size is 64. For server optimization, we set the weight of gravitation regularizer $\lambda$ is 0.5. We use SGD with a learning rate 0.1 and a weight decay of 5e−4 as the optimizer for all optimization process.

### 4.2 Performance Comparison with State-of-the-Art Algorithms

We compare FedGR with several imbalance-oriented methods like FedNova [23], SCAFFOLD [10], PerFedAvg [4], pFedMe [20], FedOpt [19], MOON [13], FedRS [15] and FedGC [18] under different degrees of double imbalance.

**Table 3.** Performance compared with state-of-the-art algorithms on Fashion-MNIST under double imbalance distribution. The best results are in **bold**, and the secondary optimal results are mark as underline.

| Algorithms | Fashion-MNIST (2) | | Fashion-MNIST (3) | |
|---|---|---|---|---|
| | Accuracy(%) | Macro-F1(%) | Accuracy(%) | Macro-F1(%) |
| FedAvg (2018) [17] | 51.36 | 50.34 | 54.34 | 50.42 |
| FedProx (2020) [14] | 53.94 | 52.75 | 55.14 | 54.97 |
| FedNova (2020) [23] | 58.23 | 56.57 | 63.35 | 62.59 |
| SCAFFOLD (2020) [10] | 56.32 | 59.44 | 65.32 | 63.94 |
| PerFedAvg (2020) [4] | 48.87 | 45.73 | 56.91 | 55.41 |
| pFedMe (2020) [20] | 47.61 | 46.47 | 52.68 | 52.13 |
| FedOpt (2021) [19] | 60.72 | 60.78 | 68.93 | 68.78 |
| MOON (2021) [13] | 60.37 | 60.85 | 72.67 | 70.52 |
| FedRS (2021) [15] | 62.52 | 60.54 | 73.16 | 70.26 |
| FedGC (2022) [18] | 62.11 | 60.74 | 73.01 | 70.13 |
| FedGR (ours) | **65.62**(3.1↑) | **63.95**(3.24↑) | **76.11**(2.95↑) | **73.23**(2.97↑) |

**Table 4.** Communication rounds compared with state-of-art algorithms on CIFAR-10 and Fashion-MNIST under double imbalance distribution.

| Algorithms | CIFAR-10 (3) | | CIFAR-100 (30) | | Fahsion-MNIST (3) | |
|---|---|---|---|---|---|---|
| | #rounds | speedup | #rounds | speedup | #rounds | speedup |
| FedAvg [17] | 1000 | 1× | 1000 | 1× | 1000 | 1× |
| FedProx [14] | 800 | 1.25× | 850 | 1.17× | 860 | 1.16× |
| FedNova [23] | 600 | 1.67× | 650 | 1.53× | 540 | 1.85× |
| SCAFFOLD [10] | 860 | 1.16× | 830 | 1.20× | 810 | 1.24× |
| FedOpt [19] | 390 | 2.56× | 450 | 2.22× | 360 | 2.77× |
| FedRS [15] | 350 | 2.56× | 430 | 2.22× | 340 | 2.77× |
| FedGC [18] | 590 | 1.69× | 650 | 1.53× | 610 | 1.63× |
| FedGR(ours) | **330** | **3.03×** | **390** | **2.56×** | **300** | **3.33×** |

For each dataset, we simulate two different double imbalance scenarios for experimental generalizability. We use the average accuracy or F1 score of the last 50 rounds to represent the performance of one experiment. We also test all methods five times to reduce random errors.

Table 2, 3 show the Top-1 accuracy and macro-F1 of compared baselines on CIFAR-10, CIFAR-100 and Fashion-MINST datasets. It can be seen that our proposed *FedGR* achieves the highest performance on both accuracy and macro-F1 under different degrees of double imbalance. Compared with baselines, the highest performance gain of FedGR appears on CIFAR-10 datasets where each client only owns two labels (around 4.53%, 4.91% improvement on accuracy and F1 for the secondary best results). pFedMe achieves the lowest performance in most double imbalance scenarios, even lower than FedAvg. The possible reason is that meta-learning is not useful for dealing with serious quantity imbalance

**Table 5.** Ablation Study of FedGR.

| Datasets | Unbalanced Softmax | Gravitation Regularizer | FedGR |
|---|---|---|---|
| CIFAR-10 (3) | ✓ | | 71.34 ± 0.24 |
| | | ✓ | 70.52 ± 0.41 |
| | ✓ | ✓ | **77.86 ± 0.35** |
| Fashion-MNIST (3) | ✓ | | 71.13 ± 0.36 |
| | | ✓ | 70.31 ± 0.38 |
| | ✓ | ✓ | **76.11 ± 0.21** |

in client. FedRS achieves the secondary best on most of the scenes because it restricts the error update of missing classes, but the quantity imbalance is still a challenge for it.

We also compare the convergence speed with baselines. We choose the accuracy after a thousand rounds of FedAvg as the standard, then compare the number of communication rounds required by other methods to reach this accuracy. The results are shown in Table 4. We delete the PerFedAvg and pFedMe in Table 4 due to these two methods cannot achieve the accuracy of FedAvg. According to Table 4, FedGR can get the least communication rounds (from 300 to 390 rounds) to achieve FedAvg accuracy on all datasets, which is around two times faster than FedAvg. The reason why FedGR needs fewer rounds is that the gravitation regularizer encourages the effective collaboration of different clients by cross-client forces, which is not considered by existing FL methods.

### 4.3 Ablation Study

In this subsection, we design an experiment on CIFAR-10 and Fashion-MNIST to investigate the effect of each component of FedGR. According to Table 5, when FedGR with only unbalanced softmax, the performance is still higher than most existing FL methods because unbalanced softmax efficiently addresses the quantity imbalance problem in client. Similarly, we also find that gravitation regularizer of FedGR improves performance by enhancing the cross-client collaboration. In addition, the results show that the performance of FedGR with unbalanced softmax and gravitation regularizer obtains a significant improvement around 6% compared with FedGR with only unbalance softmax or gravitation regularizer. Therefore, our proposed FedGR actually have the capacity to handle the double imbalance distribution.

### 4.4 Parameter Analysis

One important hyper-parameter in FedGR is the weight of gravitation regularizer. We analyze the effect of $\lambda$. We evaluate its influence by experiments on CIFAR-10 and Fashion-MNIST under different degrees of double imbalance. It can be observed from Fig. 4 that FedGR gets the best performance when $\lambda =$

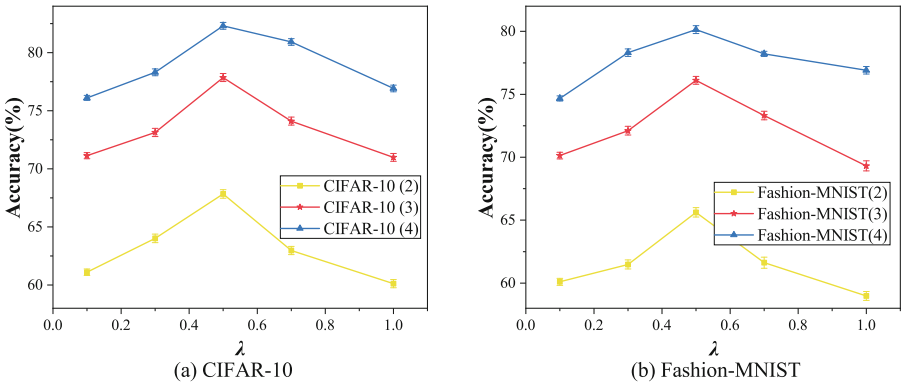


**Fig. 4.** Performance comparison with different selection of $\lambda$ on CIFAR-10 and Fashion-MNIST under different degrees of double imbalance. (a) comparison on CIFAR-10, (b) comparision on Fashion-MNIST.

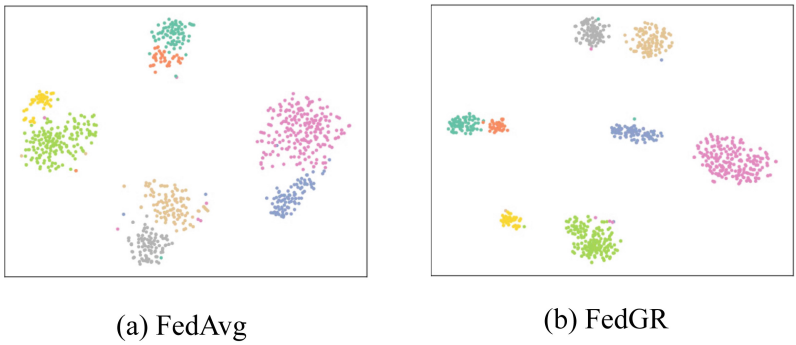


**Fig. 5.** Visualization results on feature level with t-SNE. (a) Results acquired by FedAvg, (b) Results acquired by FedGR.

0.5. When $\lambda$ approaches 0, the influence of the gravitation regularizer is diminished, and only unbalanced softmax is effective. Consequently, FedGR tends to face the label imbalance issue. When $\lambda$ is quite large (near 1), the gravitation regularizer harms the functioning of unbalanced softmax. Also, as the number of labels a client owns increases, the influence of different choices of $\lambda$ decreases as a result of the label imbalance relief.

### 4.5 Visualization Results

We visualize the samples of CIFAR-10 dataset by t-SNE [16]. In Fig. 5, points in different colors refer to the features of samples in different classes. Samples are much closer within the same class cluster means better performance. It can be seen that FedGR clearly reduces the distance between the features of samples

with the same label. This suggests that FedGR is more successful in accurate classification than FedAvg.

## 5 Conclusion

In this paper, we first show that existing FL algorithms face serious performance drop problem under double imbalance distribution. Based on this observation, we propose a novel FL algorithm called federated learning with gravitation regulation (FedGR) to deal with the problem of double imbalance distribution. We design a simple but effective unbalanced softmax by introducing a balance factor to balance the importance of classes for tackling quantity imbalance in client. Moreover, we propose a novel gravitation regularizer to call for the forces between clients for dealing with label imbalance among clients. Experiments have shown that FedGR outperforms the state-of-the-art FL methods under double imbalance distribution scenario.

**Acknowledgement.** This work was supported by National Key R & D Program of China (No. 2022YFF0606303) and National Natural Science Foundation of China (No. 62076079).

## References

1. Chen, M., et al.: Federated learning of N-gram language models. arXiv preprint arXiv:1910.03432 (2019)
2. Deng, Y., Kamani, M.M., Mahdavi, M.: Adaptive personalized federated learning. arXiv preprint arXiv:2003.13461 (2020)
3. Duan, M., Liu, D., Chen, X., Liu, R., Tan, Y., Liang, L.: Self-balancing federated learning with global imbalanced data in mobile systems. IEEE Trans. Parallel Distrib. Syst. (2021)
4. Fallah, A., Mokhtari, A., Ozdaglar, A.: Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach. In: Advances in Neural Information Processing Systems, vol. 33, pp. 3557–3568 (2020)
5. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. In: Advances in Neural Information Processing Systems, vol. 33, pp. 19586–19597 (2020)
6. Gupta, O., Raskar, R.: Distributed learning of deep neural network over multiple agents. J. Netw. Comput. Appl. **116**, 1–8 (2018)
7. Huang, H., Shang, F., Liu, Y., Liu, H.: Behavior mimics distribution: combining individual and group behaviors for federated learning. arXiv preprint arXiv:2106.12300 (2021)
8. Huang, Y., et al.: Personalized cross-silo federated learning on non-IID data. In: AAAI, pp. 7865–7873 (2021)
9. Kairouz, P., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**(1–2), 1–210 (2021)
10. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143. PMLR (2020)

11. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (2017)
13. Li, Q., He, B., Song, D.: Model-contrastive federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10713–10722 (2021)
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: Proceedings of Machine Learning and Systems, vol. 2, pp. 429–450 (2020)
15. Li, X.C., Zhan, D.C.: FedRS: federated learning with restricted softmax for label distribution non-IID data. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 995–1005 (2021)
16. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(11) (2008)
17. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
18. Niu, Y., Deng, W.: Federated learning for face recognition with gradient correction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 1999–2007 (2022)
19. Reddi, S., et al.: Adaptive federated optimization. arXiv preprint arXiv:2003.00295 (2020)
20. T Dinh, C., Tran, N., Nguyen, J.: Personalized federated learning with Moreau envelopes. In: Advances in Neural Information Processing Systems, vol. 33, pp. 21394–21405 (2020)
21. Tan, A.Z., Yu, H., Cui, L., Yang, Q.: Towards personalized federated learning. IEEE Trans. Neural Netw. Learn. Syst. (2022)
22. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
23. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 7611–7623. Curran Associates, Inc. (2020)
24. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
25. Xu, J., Xu, Z., Walker, P., Wang, F.: Federated patient hashing. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 6486–6493 (2020)
26. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
27. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. arXiv preprint arXiv:1806.00582 (2018)
28. Zheng, W., Yan, L., Gou, C., Wang, F.Y.: Federated meta-learning for fraudulent credit card detection. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, pp. 4654–4660 (2021)