FEDPGRL: PROTOTYPE-GUIDED REFINEMENT FOR NOISY LONG-TAILED FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) under label noise and long-tailed class distributions poses a fundamental challenge, as biased local supervision and skewed data jointly distort global optimization. This issue is especially pronounced when noisy annotations disproportionately affect rare classes, amplifying representation collapse across clients. We propose Federated Learning with Prototype-Guided Refinement Learning (FedPGRL), a unified framework built on CLIP2FL that addresses this coupling through prototype-driven sample refinement. FedPGRL leverages class prototypes as stable semantic anchors to guide three refinement stages: Prototypical Distance Selection (PDS), Prototypes Contrast Refinement (PCR), and Multi-Prototypical Mask Refinement (MPMR), which collectively generate soft labels and suppress noise while preserving minority-class structure. We further introduce Class Prototypical Regularization (CPR) to explicitly encourage inter-class separation and intra-class compactness during federated optimization. Extensive experiments on CIFAR-10, CIFAR-100, and Tiny-ImageNet validate that FedPGRL consistently improves convergence stability and generalization under both noisy and imbalanced FL settings.

1 INTRODUCTION

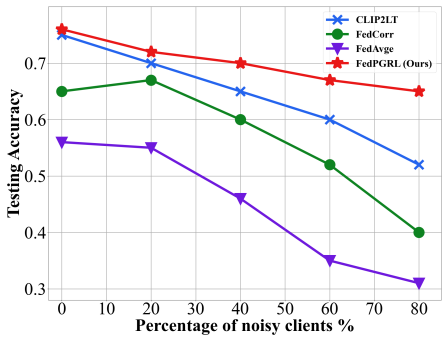
Current research in Federated Learning (FL) mainly tackles two major challenges: *noisy labels* and *long-tail data distributions* Ji et al. (2024); Xu et al. (2022); Shang et al. (2022); Shi et al. (2024). To handle label noise, many methods combine personalized and global model updates or apply client-side noise correction, such as FedCorr Xu et al. (2022), which reduces bias in the loss function and separates clean samples from mislabeled ones. However, when noisy labels and long-tailed distributions appear together, their combined biases make noise correction harder, often resulting in lower model performance and poor generalization.

Conversely, FL methods designed to handle long-tailed distributions, such as CReFF Shang et al. (2022) and CLIP2FL Shi et al. (2024), typically aggregate class-wise gradients from clients to retrain a bias-aware classifier that compensates for class imbalance. CLIP2FL adds a CLIP-based knowledge divergence to reduce dataset bias across clients. While these methods work well under clean data conditions, they generally assume that client data is noise-free. When label noise is present, the aggregated class gradients can become corrupted, weakening the retraining process. Although CLIP2FL's knowledge divergence helps to some extent, it does not fully address the combined impact of noisy labels and class imbalance. As shown in Figure 1a, the performance of these methods drops sharply as the proportion of noisy clients grows.

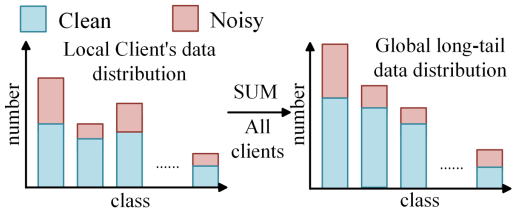
We tackle the dual challenges of noisy labels and long-tailed data distributions in federated learning—two critical issues that are often addressed separately but frequently occur together in practice. For example, in large-scale manufacturing quality inspection, data is collected from multiple production lines and locations under strict privacy constraints, making FL an obvious choice. However, annotation quality can vary greatly due to differences in inspector expertise, leading to frequent mislabeling. At the same time, defective samples are much rarer than normal ones, creating a pronounced long-tail distribution. Similar situations arise in healthcare, such as when patient cases are transferred between hospitals. To better reflect these real-world conditions, this paper introduces a more realistic FL setting that models both label noise and data imbalance jointly (see Figure 1b).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Figure 1: Illustration of the noisy long-tailed federated learning setting and the corresponding performance comparison.



(a) Testing accuracy under IF=100.



(b) Problem definition.

To cope with the combined challenges of noisy labels and long-tailed data distributions in FL, we propose **Federated Learning with Prototype-Guided Refinement Learning (FedPGRL)**. FedPGRL leverages the relationships among class prototypes to better filter and reinterpret training samples, improving both label quality and gradient reliability. Unlike FedCorr Xu et al. (2022), which focuses only on correcting label noise, FedPGRL explicitly addresses the compounded bias from noise and imbalance, reducing miscorrection risks. While CLIP2FL uses CLIP-based knowledge to enhance robustness to label noise, it is still vulnerable to client-specific class biases common in decentralized settings. FedPGRL overcomes this by recycling noisy samples through prototype-guided transformations, producing more accurate and class-representative gradient updates.

Our FedPGRL framework consists of three key stages that progressively refine data quality and model representations (explained in details later with Figure 3). (1) In the first stage, **Prototypical Distance Selection (PDS)** identifies clean and suspicious samples based on their distance to class prototypes. (2) In the second stage, **Prototypes Contrast Refinement (PCR)** uses feature similarities among clean samples to adjust suspicious ones, generating soft labels that improve feature extraction and classification robustness. (3) In the final stage, **Multi-Prototypical Mask Refinement (MPMR)** uses these soft labels to further enhance label quality and feature learning, leading to cleaner class gradients and less noise interference. In addition, we propose the **Class Prototypical Regularization (CPR)**, which encourages semantically similar classes to cluster together in the feature space while pushing apart dissimilar classes. This helps the model better capture inter-class relationships, boosting performance under noisy labels and long-tailed data.

Our main contributions are summarized as follows:

- We propose FedPGRL, a federated learning (FL) framework that effectively addresses noisy labels and long-tailed data to enable robust classification in real-world scenarios.
- Our Prototype-Guided Refinement Learning (PGRL) method distinguishes clean from noisy data, creates soft labels to enhance feature extraction, and optimizes federated features to better capture class-specific characteristics.
- Our Class Prototypical Regularization (CPR) strategy encourages similar classes to cluster together while pushing different classes apart in the feature space.
- We conduct extensive experiments showing that FedPGRL consistently outperforms the state-of-the-art approaches on CIFAR-10, CIFAR-100, and Tiny-ImageNet under both simulated and real-world FL settings.

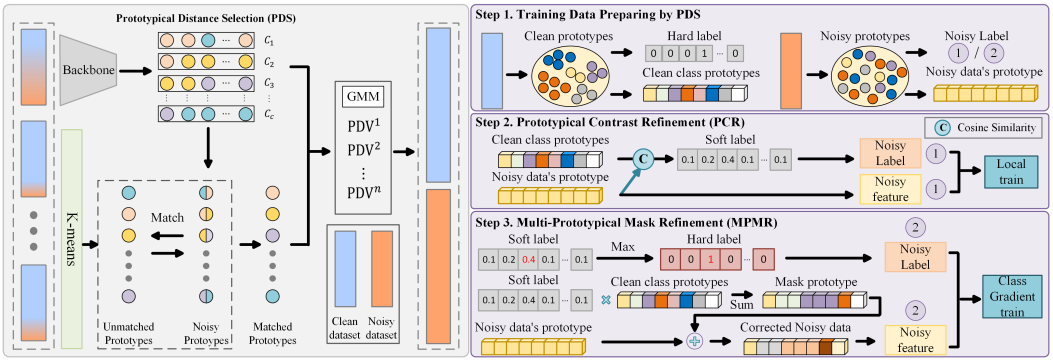


Figure 2: Detailed architectures of the Prototype-Guided Refinement Learning (PGRL), which consists of three steps: (1) PDS uses K-means clustering and GMM to classify data as clean or noisy based on Prototypical Distance Values (PDVs); (2) PCR refines noisy data into soft labels using clean prototypes and cosine similarity for stable local training; (3) MPMR converts soft labels into hard labels, creating multi-prototypical data to enhance class gradient training.

2 METHOD

2.1 SETTING AND NOTION

Our **FedPGRL** framework tries to tackle FL challenges arising from both long-tailed class distributions and label noise. Consider a large dataset $\mathbf{D}$, consisting of M samples spanning C classes, distributed among I clients. The dataset is partitioned as $\mathbf{D} = \bigcup_{i=1}^{I} \mathbf{D}^i$, where each local dataset $\mathbf{D}^i = \{(x_{i,j}, \hat{y}_{i,j})\}_{j=1}^{N_i}$ includes N_i samples indexed by j, and the data distribution for each client follows $(X_i, Y_i) \sim \mathbf{D}^i$. Here, $\hat{y}_{i,j}$ represents the noisy version of the true label $y_{i,j}$, defined as $\hat{y}_{i,j} = \boldsymbol{h}(y_{i,j})$, where $\boldsymbol{h}$ is a function producing label noise Ji et al. (2024).

The global dataset $\mathbf{D}$ exhibits a long-tailed class distribution Shang et al. (2022); Shi et al. (2024). For each class $c \in \{1, 2, \ldots, C\}$, we define $\mathbf{D}_c^i = \{(x, \hat{y}) \in \mathbf{D}^i : \hat{y} = c\}$ as the subset of samples in class c on client i, where n_i^c denotes the number of samples for class c on client i. The total sample count for class c across all clients is $N_c = \sum_{i=1}^{I} n_i^c$. Classes are ranked in descending order by their total sample counts: $N_1 > N_2 > \cdots > N_C$, where N_1 is the most frequent class and N_C is the rarest.

In real-world scenarios, the dataset $\mathbf{D}$ faces challenges from both imbalanced class distributions and varying levels of label noise across clients. In our **FedPGRL** framework, both the client-side and server-side models consist of a feature extractor $\mathbb{F}_\theta$ (parameterized by θ) and a linear classifier $\mathbb{G}_\phi$ (parameterized by ϕ). For an input sample x, the feature extractor produces a d-dimensional representation $z = \mathbb{F}_\theta(x)$, which the classifier then uses to generate logits $s = \mathbb{G}_\phi(z)$, representing the predicted label probabilities for x. The full model parameters are denoted as $\mathbf{w} = \{\theta, \phi\}$. In each training round, a subset of clients denoted by $\mathbf{B}$ is randomly selected to participate.

2.2 THE FRAMEWORK OF OUR PROPOSED FEDPGRL

FedPGRL is a federated learning framework designed to address noisy labels, client heterogeneity, and long-tailed data distributions. As shown in Fig. 3, FedPGRL adopts a collaborative training paradigm in which the client and server models are optimized separately. On the client side (the left part of 3), the Prototype-Guided Refinement Learning (PGRL) first differentiates clean and noisy samples through Prototypical Distance Selection (PDS), then stabilizes feature learning via Prototypical Contrast Refinement (PCR), and finally applies the Class Prototypical Regularization (CPR) loss for robust optimization. Multi-Prototypical Mask Refinement (MPMR) is further employed to enhance class-wise gradient signals during training. On the server side (the right part of Fig. 3), aggregated model weights and gradients are used to refine global feature representations, resulting in a more balanced classifier. In the following, we provide the detailed formulation of the client and server models.

2.3 PROTOTYPE-GUIDED REFINEMENT LEARNING (PGRL)

Filtering and reusing noisy data is crucial for managing complex noisy datasets. This paper proposes the PGRL to address this problem, as detailed in Fig. 2. PGRL includes three stages: PDS, PCR, and MPMR. PDS separates clean and noisy data. PCR and MPMR further refine noisy data and optimize local model training and class gradient training, respectively. In addition, we propose the CPR strategy to encourage similar classes to cluster together while pushing different classes apart in the feature space. We also incorporate prior knowledge from large multimodal models (*e.g.*, CLIP) to improve training efficiency.

Step 1 – Data preparation with PDS. During data preparation, we propose the Prototypical Distance Selection (PDS) module to separate clean and noisy samples in each client's dataset (the left part of Fig. 2). The key idea is to extract class prototypes and compute representative centers to detect noisy samples. However, label noise in local data can lead to inaccurate prototypes when using class centers based on labels. To address this, we apply K-means clustering Krishna & Murty (1999) to group data based on its structure, without relying on labels. This produces more reliable prototype clusters, reducing label noise impact and improving clean data selection. Since the cluster centers are unlabeled, we use the original noisy labels to compute reference clusters and associate them with classes. The relationship between each cluster center and the noisy class prototypes is then measured using L2 distance. The details of the PDS strategy for the i-th client are as follows:

For the i-th client, there are J_i noisy prototypes; that is, $\mathbf{P}^i_{noisy} = \{p^i_j\}$. The K-means clusters all the J_i noisy prototypes into C classes to obtain the set of cluster centers $\mathbf{Q}^i = \{Q^i_c\}$, where the initial center of Q^i_c is the mean of the c-th class obtained from the noisy labels of p^i_j. Then, the distance between p^i_j and Q^i_c is measured by the form:

$$d(p^i_j, Q^i_c) = 1 - \mathbf{sim}(p^i_j, Q^i_c) = 1 - \frac{(p^i_j)^\top \cdot Q^i_c}{\|p^i_j\| \|Q^i_c\|}. \tag{1}$$

The best class $\hat{c}$ is assigned to p^i_j if $d(p^i_j, Q^i_{\hat{c}})$ is the smallest:

$$\hat{c} = \arg\min_c d(p^i_j, Q^i_c). \tag{2}$$

This assignment process ensures that each cluster center is aligned with its most closely related class, establishing a reliable foundation for separating clean data from noisy data. After matching clusters, we can get the clean prototype set $\mathbf{P}^i_{clean}$. According to the class type of each clean prototype in $\mathbf{P}^i_{clean}$, we can divide it into C classes of prototypes, *i.e.*, $\{\mathbf{P}^{i,c}_{clean}\}$. The mean of $\mathbf{P}^{i,c}_{clean}$ is served as the clean cluster center $Q^{i,c}_{clean}$ of the c-th class in the i-th client. Then, we calculate the Prototypical Distance Values (PDV) to quantify the relationship between each data $x \in \mathcal{D}^c_i$ and its corresponding cluster center $Q^{i,c}_{clean}$ Snell et al. (2017) and use these values to indicate whether a sample is close to its corresponding original label cluster. If it is close, the label is considered correct; otherwise, it is regarded as a noisy label. The PDV is defined as:

$$PDV^i_c(x) = d(x, Q^{i,c}_{clean}). \tag{3}$$

Then, we get the $PDV^i(x) = \{PDV^i_c(x)\}$ distibution in the i-th client. We then use a Gaussian Mixture Model (GMM) to distinguish between noisy and clean datasets, denoted as $\mathbf{D}^{i,c}_{noisy}$ and $\mathbf{D}^{i,c}_{clean}$, respectively. The GMM models the distribution of $x \in \mathbf{D}^i_c$, enabling precise separation of clean and noisy data. The data farther from their class cluster centers are labeled as noisy, while those closer are considered as clean.

After selecting the noisy and clean datasets, we cannot simply discard the noisy data, as it may contain valuable learnable information that can enhance the feature extractor. Therefore, we propose two innovative methods in the next steps: Prototypes Contrast Refinement (PCR) and Multi-Prototypical Mask Refinement (MPMR). These refinement methods help process noisy data, allowing it to be effectively reused in different training stages within FedPGRL.

Step 2 – Prototypical Contrast Refinement (PCR). PCR (the middle-right part of Fig. 2) handles noisy labeled data by replacing hard labels with soft labels, reducing the impact of label noise on model learning. Noisy labels can cause the model to learn incorrect decision boundaries, harming generalization. To address this, PCR uses prototype learning to adjust data distribution and

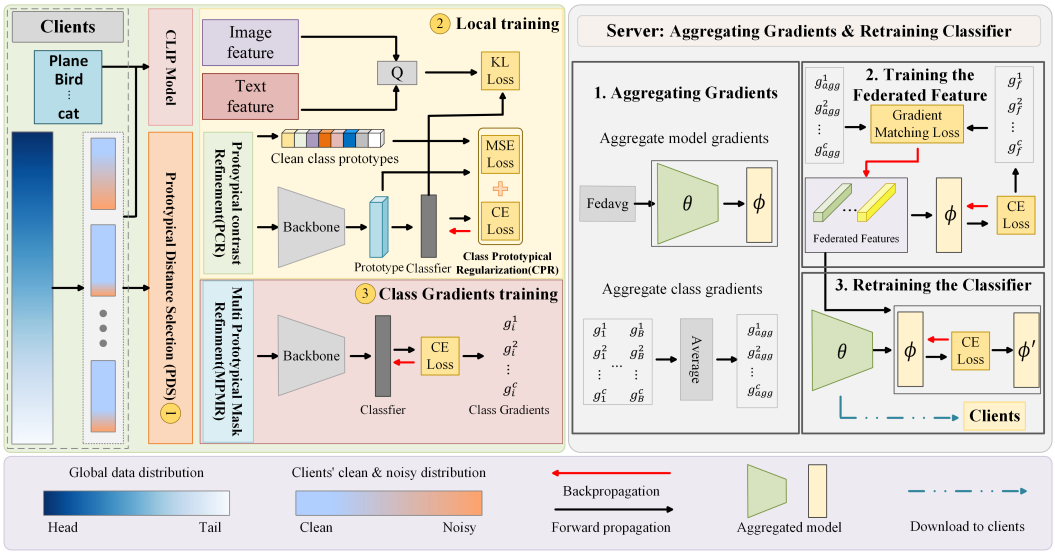


Figure 3: FedPGRL addresses long-tail noisy label federated learning. On clients, PGRL tries to filter and reuse noisy data. It separates clean and noisy data using PDS, refines samples with PCR for stable feature learning, and applies CPR loss for optimization. MPMR further improves class gradients during training. On the server, aggregated weights and gradients are used to retrain federated features, which leads to a balanced classifier. Here, CE= Cross Entropy, MSE = Mean Squared Error, and KL = Kullback–Leibler.

improve feature learning. PCR constructs class prototypes using clean data and re-evaluates noisy samples based on these prototypes, aligning their label distributions with the true class. Using a prototype-based contrastive learning mechanism, PCR guides the model toward the correct prototypes while minimizing the influence of noisy labels on gradient updates, thus enhancing stability and generalization.

The following is the process: for each noisy sample $x \in \mathbf{D}_{noisy}^{i,c}$, we compute its similarity with every clean class center $Q_{clean}^{i,c}$ as follows:

$$\mathbf{sim}(x, Q_{clean}^{i,c}) = \frac{x^\top \cdot Q_{clean}^{i,c}}{\|x\| \|Q_{clean}^{i,c}|}. \tag{4}$$

Next, the similarity scores are normalized to generate soft labels by applying the softmax function across the scores for all classes. The soft label $\bar{y}$ is then computed as:

$$\bar{y}_c^i(x) = \frac{\exp(\mathbf{sim}(x, Q_{clean}^{i,c}))}{\sum_{c=1}^{C} \exp(\mathbf{sim}(x, Q_{clean}^{i,c}))}. \tag{5}$$

Generating soft labels $\bar{y}$ that reflect the relationship between the noisy data and the clean class prototypes can reduce the impact of noisy labels. As a result, the model becomes more stable and can learn discriminative features better.

Step 3 – Multi Prototypical Mask Refinement (MPMR) In PCR, the objective is to prevent noisy samples from corrupting the feature extractor by converting noisy labels into soft labels. However, soft labels cannot be directly used for class-gradient–based classifier updates, creating a challenge for effectively leveraging noisy data. To address this, we introduce Multi-Prototypical Mask Refinement (MPMR) (Fig. 2, bottom-right). MPMR converts soft labels into hard labels while jointly refining noisy prototypes to ensure consistency with updated supervision. Since converted labels may deviate from the original prototypes, we redefine masked prototypes and integrate them with the noisy features to mitigate negative gradient effects. For refining noisy prototypes, we use clean class prototypes $Q_{clean}^{i,c}$, and integrate them with soft labels $\bar{y}$ from noisy data x to create a new mask prototype $m(x)$. The formula is given as follows:

$$m(x) = \sum_{c=1}^{C} \bar{y}_c^i(x) \times Q_{clean}^{i,c}. \tag{6}$$

After obtaining $m(x)$, we aim to preserve the original prototype's distinct characteristics. To achieve this, we combine the original noisy prototype x with the mask prototype to generate the final corrected noisy data. The mask prototype is appropriately weighted and merged with x to refine the noisy data $\hat{x}$ effectively, as defined by the following function:

$$\hat{x} = \alpha \cdot x + (1 - \alpha) \cdot m(x), \tag{7}$$

where α is a balancing factor that adjusts the contribution of mask prototypes and noisy prototypes. Its effects will be addressed in the appendices. To allow these noisy data to be used in gradient learning, we convert the soft labels into hard labels by assigning each noisy data $\hat{x}$ to the class with the highest soft label score:

$$SL(x) = \arg\max_c \bar{y}_c^i(\hat{x}). \tag{8}$$

This approach assigns noisy data to the class with the highest score in its soft label, strengthening the class prototypes most associated with the noisy data. It effectively reduces the bias in class gradients caused by noisy data.

2.4 CLIENT-SIDE LEARNING

Client training consists of two stages: **local training** and **class gradient training**. For local training, we enhance class feature separation, especially in noisy and long-tailed distributions where the Cross-Entropy (CE) loss struggles. To address this, we introduce the Class Prototypical Regularization (CPR), which combines the CE loss with the Mean Squared Error (MSE) loss between training samples' prototypes and clean class prototypes. This improves feature separation and refines inter-class relationships. The MSE loss with clean prototypes is:

$$\mathcal{L}_{\text{MSE}} := \sum_x \|x - Q_{clean}^{i,SL(x)}\|^2, \tag{9}$$

where $SL(x)$ is obtained from Eq.(8). During the loss computation, each sample calculates its loss relative to its class prototype, which promotes better feature separation across different classes. The final CPR loss function combines the CE loss and the MSE loss as follows:

$$\mathcal{L}_{\text{CPR}} = (1 - \lambda) \cdot \mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{MSE}}, \tag{10}$$

where λ is a hyperparameter that controls the balance between the CE loss and the MSE loss. The effect of λ will be addressed in the appendices. Then, the total loss for local training is then formulated as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CPR}} + \mathcal{L}_{\text{KL}}, \tag{11}$$

where $\mathcal{L}_{\text{KL}}$ is the Kullback–Leibler (KL) divergence loss. As a result, this approach improves feature separability and overall model performance, particularly in scenarios involving noisy and long-tailed data.

In the **class gradient training** stage, each client computes class gradients using only the Cross-Entropy (CE) loss and sends them to the server. The server then aggregates these gradients to retrain the classifier, ensuring robust learning across clients. By relying solely on the CE loss, it preserves each class's fundamental learning characteristics.

2.5 SERVER-SIDE LEARNING

On the server side (the right part of Fig. 3), after aggregating gradients, the clean class gradients from the clients are used to train the learnable parameters of federated features, enabling them to approximate the original class data. This approach mitigates data imbalance and enables effective retraining. It involves three main steps as follows.

Step 1 – Aggregating gradients: It involves collecting the gradients from the client-side backbone and classifiers, which are associated with their respective local models: a feature extractor (a backbone) $\mathbb{F}_\theta$ (parameterized by θ) and a linear classifier $\mathbb{G}_\phi$ (parameterized by ϕ). To aggregate the gradients from multiple clients, a weighted averaging approach is used. Let $\nabla\theta_i$, $\nabla\phi_i$, and ∇g_c^i represent the backbone, classifier, and class gradients of the i-th client, respectively. To do this, each client is assigned a weight ρ_i based on its local data contribution as follows:

$$\rho_i = \frac{M_i}{\sum_{i=1}^{B} M_i}, \tag{12}$$

where $M_i = \|\mathcal{D}_i\|$ and B is a part of selected clients in each training round. The aggregated backbone gradients $\nabla\theta$, classifier parameter gradients $\nabla\phi$, and classifier gradients $\bar{g}_c$ are computed as follows:

$$\nabla\theta = \sum_{i=1}^{B} \rho_i \nabla\theta_i, \nabla\phi = \sum_{i=1}^{B} \rho_i \nabla\phi_i, \text{and } \bar{g}_c = \sum_{i=1}^{B} \rho_i g_c^i. \tag{13}$$

This weighted averaging ensures that the contributions of clients with larger datasets are proportionally reflected in the aggregated gradients while maintaining the privacy of individual client data.

Step 2 – Training the federated feature: It introduces a learnable parameter, referred to as Federated Features, denoted as $\mathbf{F}_c^i = \{f_c^i | c \in C\}|_{i=1}^{m_c}$, where m_c denotes the number of Federated Features in class c. The features are randomly generated and fed into the aggregated classifier, enabling it to produce class gradients via the Cross-Entropy (CE) loss for initial training. The class gradients derived from the Federated Features are computed as:

$$g_c^{fed} = \frac{1}{m_c} \sum_{i=1}^{m_c} \nabla_\phi \mathcal{L}_{\text{CE}}(f_c^i, y_i), \tag{14}$$

where y_i denotes the corresponding label. A key challenge with Federated Features lies in ensuring that the generated class gradients align closely with those of real features. To address this, we employ a gradient matching loss $\mathcal{L}_{\text{grad}}$ Zhao et al. (2020), which quantifies the difference between the gradients produced by the server classifier for both the virtual and real features, minimizing this discrepancy. $\mathcal{L}_{\text{grad}}$ is defined as:

$$\mathcal{L}_{grad}(g_c^{fed}, \bar{g}_c) = \frac{1}{C} \sum_{k=1}^{C} \left(1 - \frac{g_c^{fed}[k] \cdot \bar{g}_c[k]}{\|g_c^{fed}[k]\| \times \|\bar{g}_c[k]\|} \right),$$

where $g_c^{fed}[k]$ and $\bar{g}_c[k]$ represent the k-th row of the gradient vectors for the virtual and aggregated features, respectively. This loss ensures that the Federated Features are as close as possible to the real features.

Step 3 – Retraining the classifier: Retraining the Classifier involves leveraging the learned Federated Features to refine and enhance each classifier's performance. These features are used to construct a balanced federated dataset with an equal number of features per class. This dataset mitigates class imbalance and improves classification accuracy. After aggregation, we obtain the updated backbone and classifier weights ($\Delta\theta$, $\Delta\phi$). The aggregated classifier parameters $\Delta\phi$ serve as the training target for Federated Features during retraining. The classifier weights are then updated using the following updating rule:

$$\Delta\phi' = \Delta\phi - \eta \nabla_\phi \mathcal{L}_{\text{CE}}, \tag{15}$$

where $\Delta\phi'$ is the updated classifier weights after retraining, η the learning rate controlled by the AdaM algorithm, and $\mathcal{L}_{\text{CE}}$ represents the cross-entropy loss between the predicted and true labels. This retraining process generates new classifier weights, which improve classification performance and better align with the previously updated classifier parameters $\Delta\phi$. Finally, the updated aggregated backbone parameters $\Delta\theta$ and the retrained classifier parameters $\Delta\phi'$ are distributed back to the client models, preparing them for the next round: $\hat{w}^{t+1} = \{\theta^{t+1}, \phi'^{t+1}\}$. Details of the training process are shown in Algorithm 1 in the appendix.

3 Experiments

3.1 Experiment setup

Dataset: We evaluated FedPGRL on three widely used datasets with globally long-tailed distributions and noisy labels: CIFAR-10-LT Krizhevsky et al. (2009), CIFAR-100-LT Krizhevsky et al. (2009), and Tiny-ImageNet-LT Deng et al. (2009). We generated long-tailed distributions by sampling data with imbalance factors (IF = 100, 50, 10). To simulate a realistic noisy environment, we introduced noise using parameters (γ, ρ), where γ represents the fraction of noisy clients and ρ denotes the lower bound of the noise level, following the setup in Xu et al. (2022). Consistent with CReFF, we employed a Dirichlet distribution with parameter $\delta = 0.5$ to create heterogeneous data partitions across clients. The effects of α in Eq. (7) and λ in Eq. (10) were analyzed in the appendices.

Table 1: Test accuracy (%) on simulated CIFAR-10 with different noise rate and imbalance factor. Bold indicates the highest score. HO= Heterogeneity-oriented. A: (0.0, 0.0), B: (0.4, 0.0), C: (0.4, 0.5), D: (0.6, 0.0), E: (0.6, 0.5), F: (0.8, 0.0), G: (0.8, 0.5).

Imbalance Factor (IF)		10							100						
Noise Rate (γ, ρ)		A	B	C	D	E	F	G	A	B	C	D	E	F	G
HO	FedAvg	77.45	63.04	60.01	58.86	50.31	53.56	43.01	56.17	47.88	46.10	42.24	35.50	37.44	30.72
FL	FedProx	76.53	63.82	62.01	58.45	51.17	54.44	43.31	56.92	46.50	46.51	43.53	37.56	38.34	29.06
Noisy FL	FedCorr	81.08	73.47	74.38	72.56	68.31	69.12	66.04	65.09	57.99	60.03	48.92	52.00	42.97	39.90
	FedNed	44.77	47.27	47.43	43.63	41.58	39.81	38.39	29.47	38.55	36.55	33.47	30.71	28.77	26.49
Long-tail FL	CreFF	80.71	66.54	63.98	56.79	50.89	43.84	51.40	70.55	51.02	49.70	41.11	37.56	37.72	32.27
	CLIP2FL	81.18	76.48	75.76	72.71	68.47	70.05	65.71	73.37	65.26	64.92	62.70	59.96	57.63	51.78
Ours	FedPGRL	**81.81**	**78.04**	**77.88**	**76.41**	**75.10**	**75.47**	**74.23**	**73.64**	**68.01**	**70.12**	**67.93**	**67.28**	**65.41**	**64.51**

Table 2: Test accuracy (%) on simulated CIFAR-100 with different noise rate and imbalance factor. Bold indicates the highest score. HO= Heterogeneity-oriented. A: (0.0, 0.0), B: (0.4, 0.0), C: (0.4, 0.5), D: (0.6, 0.0), E: (0.6, 0.5), F: (0.8, 0.0), G: (0.8, 0.5).

Imbalance Factor (IF)		10							100						
Noise Rate (γ, ρ)		A	B	C	D	E	F	G	A	B	C	D	E	F	G
HO	FedAvg	45.87	36.89	35.50	31.36	25.37	27.34	18.46	30.34	23.89	23.08	21.61	17.00	18.55	12.23
FL methods	FedProx	44.66	36.81	34.74	32.31	25.86	27.49	19.24	31.67	24.53	22.78	22.07	17.21	18.26	12.39
Noisy FL	FedCorr	48.18	46.06	44.54	36.77	35.77	32.15	24.93	37.45	31.85	31.54	26.28	20.16	21.27	16.87
	FedNed	26.13	27.15	26.77	23.07	22.03	20.02	18.08	17.26	18.50	18.41	16.33	16.40	14.37	12.77
Long-tail FL	CreFF	47.08	33.58	32.78	28.90	23.19	23.33	15.57	34.67	23.35	20.09	20.42	15.09	16.34	10.49
	CLIP2FL	48.20	44.80	43.91	41.37	36.10	36.51	28.97	37.56	31.73	30.73	29.49	25.83	25.78	20.69
Ours	FedPGRL	**48.77**	**46.04**	**46.52**	**43.26**	**42.12**	**42.60**	**39.31**	**38.55**	**34.43**	**32.29**	**31.86**	**29.95**	**29.76**	**28.21**

Implementation: Following CLIP2FL, we use ResNet-8 as the feature extractor for CIFAR-10/100-LT and Tiny-ImageNet-LT, and incorporate CLIP with ViT-B/32. The system consists of 20 clients, with 40% randomly selected for training. Each client uses a batch size of 32, with 100 federated features per class. Experiments run on PyTorch with an NVIDIA GeForce RTX 3090, completing 200 communication rounds for optimal results. The same environment is implemented on the methods compared, and for the special parameters in certain methods, we follow the guidelines provided in their GitHub repositories.

3.2 Experience results

Competing methods: We compare FedPGRL with six FL methods: FedAvg McMahan et al. (2017) , FedProx Li et al. (2020), FedCorr Xu et al. (2022), FedNed Lu et al. (2024), CReFF Shang et al. (2022), and CLIP2FL Shi et al. (2024). Among these, FedAvg and FedProx are heterogeneity-oriented methods, FedCorr and FedNed focus on handling noisy labels, while CReFF and CLIP2FL are designed for federated long-tail scenarios.

Results on CIFAR-10/100-LT: Tables 1 and 2 compare FedPGRL with prior FL methods on CIFAR-10-LT and CIFAR-100-LT. FedPGRL achieves the highest accuracy across all settings. Under extreme long-tailed conditions (IF = 10, 100) with varying noise rates, FedPGRL consistently outperforms baselines, improving accuracy by 1% to 13% on CIFAR-10-LT and 1% to 9% on CIFAR-100-LT. Additional efficiency comparisons and visualizations are provided in the appendices.

Results on Tiny-ImageNet-LT: In Table 5 of the appendices, we present the classification accuracy of FedPGRL compared to various FL methods on Tiny-ImageNet-LT. The dataset is divided into four categories: Many (more than 100 samples), Medium (20–100 samples), Few (fewer than 20 samples), and All (overall accuracy). FedPGRL consistently outperforms most FL methods, with a notable improvement of 8% in the Head, 4% in the Medium, and 1.5% in the Few categories compared with CLIP2FL.

Table 3: Ablation studies under imbalance factor (IF) = 100 and noisy rate $(\gamma, \rho) = (0.6, 0.5)$.

(a) Effect of PGRL and $\mathcal{L}_{\text{CPR}}$.

Methods	PGRL	$\mathcal{L}_{\text{CPR}}$	CIFAR10	CIFAR100
CLIP2FL	✗	✗	59.96	25.83
w/o PGRL	✗	✓	64.42	28.72
w/o $\mathcal{L}_{\text{CPR}}$	✓	✗	61.35	27.00
FedPGRL	✓	✓	**67.28**	**29.95**

(b) Effect of different PGRL components.

Method	PCR	MPMR	PDS	CIFAR10	CIFAR100
w/o PCR	✗	✓	✓	63.67	28.50
w/o MPMR	✓	✗	✓	65.39	29.74
w/o PDS	✓	✓	✗	65.25	28.66
FedPGRL	✓	✓	✓	**67.28**	**29.95**

3.3 Ablation Studies and Discussions

We conducted a comprehensive ablation study on CIFAR-10 and CIFAR-100 with a noise rate of $(\gamma, \rho) = (0.6, 0.5)$ and an imbalance factor of 100, analyzing the impact of each component. Table 3a brings two findings: (1) **PGRL** module effectively improves model performance, achieving an approximate 3% gain over CLIP2FL. (2) **CPR Loss** brings features of the same class closer while pushing different classes apart. It enhances the performance by around 5% compared with CLIP2FL.

PGRL analysis: To evaluate the effectiveness of the three modules (PDS, PCR, and MPMR), we conducted the ablation study in Table 3b, removing each component individually. 1. **Without PCR**, unprocessed noisy data disrupts learning, significantly degrading classification accuracy across all classes. 2. **Without MPMR**, class gradient generation becomes biased, affecting model balance and degrading the performance. 3. **Without PDS**, the model lacks proper guidance, leading to incorrect learning directions and a negative impact on performance. When all components are integrated, the model achieves the maximum improvement, resulting in a more robust and stable performance. In addition, we provide the visualization of the ablation study of different components in the appendices.

4 Conclusions

FedPGRL surpasses the state-of-the-art methods in federated learning under global long-tail distributions and noisy client settings. The PGRL mechanism effectively separates clean and noisy data by leveraging class feature relationships, while the CPR Loss enhances the classification performance by pulling similar features closer. Our experimental results confirm the effectiveness of FedPGRL in these challenging environments. Extensive experiments across CIFAR-10/100-LT and Tiny-ImageNet-LT confirmed that FedPGRL achieves state-of-the-art performance in both accuracy and stability, demonstrating significant gains over FedCorr, CReFF, and CLIP2FL. The ablation studies further validated the individual and combined effectiveness of the PGRL components and the CPR loss, showing that each stage contributes synergistically to model resilience and convergence.

Limitations: Although FedPGRL shows consistent improvements under noisy long-tailed federated learning, several limitations remain. First, our experiments are mainly conducted on standard image classification benchmarks with simulated long-tailed distributions and label noise. While this setting allows controlled evaluation, real-world federated systems may contain more complex noise patterns, open-set classes, domain shifts, and client participation dynamics. Second, FedPGRL relies on prototype quality to separate and refine noisy samples. When local data are extremely scarce or when class features are highly overlapping, the estimated prototypes may become less reliable, which can affect the quality of sample selection and label refinement. Third, the method introduces additional refinement stages and hyperparameters, such as the mask interpolation weight α and the CPR loss weight λ, which may require tuning when transferring the method to new datasets or deployment environments. Finally, although our experiments follow the same setting as the compared methods, larger-scale federated systems with more clients, higher-resolution data, or stricter communication constraints may require further optimization of computation and communication cost.

Future Work: We aim to refine the PGRL mechanism and loss functions to better handle tail data, ensuring more balanced performance across all classes. In addition, we plan to explore how does the runtime and memory footprint of PDS, PCR, and MPMR scale with the number of classes and client data sizes.

AI USE STATEMENT

We used generative AI tools for proofreading and language refinement, including correcting grammar and improving sentence structure, clarity, and readability. The authors reviewed the suggested edits to ensure that the intended scientific meaning was preserved and take full responsibility for the manuscript's content.

REFERENCES

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Chun-Mei Feng, Bangjun Li, Xinxing Xu, Yong Liu, Huazhu Fu, and Wangmeng Zuo. Learning federated visual prompt in null space for mri reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8064–8073, 2023a.

Chun-Mei Feng, Kai Yu, Nian Liu, Xinxing Xu, Salman Khan, and Wangmeng Zuo. Towards instance-adaptive inference for federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23287–23296, 2023b.

Wenke Huang, Yuxia Liu, Mang Ye, Jun Chen, and Bo Du. Federated learning with long-tailed data via representation unification and classifier rectification. *IEEE Transactions on Information Forensics and Security*, 2024.

Xinyuan Ji, Zhaowei Zhu, Wei Xi, Olga Gadyatskaya, Zilong Song, Yong Cai, and Yang Liu. Fedfixer: Mitigating heterogeneous label noise in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12830–12838, 2024.

K Krishna and M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jichang Li, Guanbin Li, Hui Cheng, Zicheng Liao, and Yizhou Yu. Feddiv: Collaborative noise filtering for federated learning with noisy labels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 3118–3126, 2024.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623*, 2021.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33:2351–2363, 2020.

Yang Lu, Lin Chen, Yonggang Zhang, Yiliang Zhang, Bo Han, Yiu-ming Cheung, and Hanzi Wang. Federated learning with extremely noisy clients via negative distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14184–14192, 2024.

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Dipankar Sarkar, Ankur Narang, and Sumit Rai. Fed-focal loss for imbalanced data classification in federated learning. *arXiv preprint arXiv:2011.06283*, 2020.

Xinyi Shang, Yang Lu, Gang Huang, and Hanzi Wang. Federated learning on heterogeneous and long-tailed data via classifier re-training with federated features. *arXiv preprint arXiv:2204.13399*, 2022.

Jiangming Shi, Shanshan Zheng, Xiangbo Yin, Yang Lu, Yuan Xie, and Yanyun Qu. Clip-guided federated learning on heterogeneity and long-tailed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14955–14963, 2024.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Addressing class imbalance in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10165–10173, 2021.

Zikai Xiao, Zihan Chen, Songshang Liu, Hualiang Wang, Yang Feng, Jin Hao, Joey Tianyi Zhou, Jian Wu, Howard Yang, and Zuozhu Liu. Fed-grab: Federated long-tailed learning with self-adjusting gradient balancer. *Advances in Neural Information Processing Systems*, 36, 2024.

Jingyi Xu, Zihan Chen, Tony QS Quek, and Kai Fong Ernest Chong. Fedcorr: Multi-stage federated learning for label noise correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10184–10193, 2022.

Yaopei Zeng, Lei Liu, Li Liu, Li Shen, Shaoguo Liu, and Baoyuan Wu. Global balanced experts for federated long-tailed learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4815–4825, 2023.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

IF & Noise Rate (γ, ρ)		100 & (0.6,0.5)			
Division		All	Many	Medium	Few
HO FL methods	FedAvg	9.43	21.25	4.68	1.16
Noisy FL methods	FedCorr	10.18	23.51	4.94	0.73
Long-tail FL methods	CreFF	8.24	17.62	5.20	0.83
	CLIP2FL	15.77	30.77	11.37	3.40
Ours	FedPGRL	**20.41**	**38.85**	**15.20**	**4.96**

Table 4: Test accuracy (%) on simulated Tiny-ImageNet with different noise rates and imbalance factors. Bold indicates the highest score. (HO:Heterogeneity-oriented)

A Related work

A.1 Federated label-noise learning

In federated learning (FL) with noisy labels, varying noise levels across clients can hinder model training. Several methods have been proposed to tackle this issue. For example, FedCorr Xu et al. (2022) uses Local Intrinsic Dimensionality (LID) to separate clean samples from noisy ones and filter out unreliable data. FedFixer Ji et al. (2024) combines personalized and global models to select clean samples and improve training with a regularizer. FedDiv Li et al. (2024) models global label noise and applies prediction-consistency sampling to relabel noisy samples, boosting performance. FedNed Lu et al. (2024) employs negative distillation to leverage noisy clients and enhance their models in extreme noise cases. While these approaches improve FL by addressing noise through filtering and optimization, they cannot properly handle long-tailed data distributions, but our method FedPGRL is able to handle them.

A.2 Federated learning with heterogeneous data

In FL, data heterogeneity Li et al. (2019) arises from diverse data distributions across clients, which negatively impacts model performance. To address this, the strategies like optimizing learning methods Li et al. (2020); Feng et al. (2023b), enhancing knowledge transfer Luo et al. (2021); Li et al. (2021); Feng et al. (2023a), and improving aggregation strategies Lin et al. (2020) have been proposed.

Class imbalance is a key challenge within data heterogeneity. Methods like Ratio Loss Wang et al. (2021) estimate data composition to reduce imbalance effects. The Fed-Focal Loss Sarkar et al. (2020) modifies the loss function with focal loss, adjusting weights for correctly classified samples and incorporating client model contributions. Together, these methods improve model adaptability, robustness, and performance in handling both data heterogeneity and class imbalance in FL. However, these methods cannot handle the cases when the data labels are noisy, but our method FedPGRL is able to handle them.

A.3 Federated long-tail learning

Long-tailed FL stems from globally imbalanced class distributions that impair minority-class learning. Existing solutions either modify client-side optimization (e.g., GBME Zeng et al. (2023), FedGraB Xiao et al. (2024), RCUR Huang et al. (2024)) or transmit client information to the server for classifier retraining or distillation (e.g., CReFF Shang et al. (2022), CLIP2FL Shi et al. (2024)), incurring additional communication or privacy overhead. In contrast, FedPGRL mitigates long-tailed bias via Prototype-Guided Refinement Learning (PGRL) and Class Prototypical Regularization (CPR), requiring no explicit server-side classifier retraining and thus offering improved communication efficiency, privacy preservation, and scalability.

In the appendices, we provide additional experiments and analyses to further illustrate the design choices and robustness of FedPRGL.

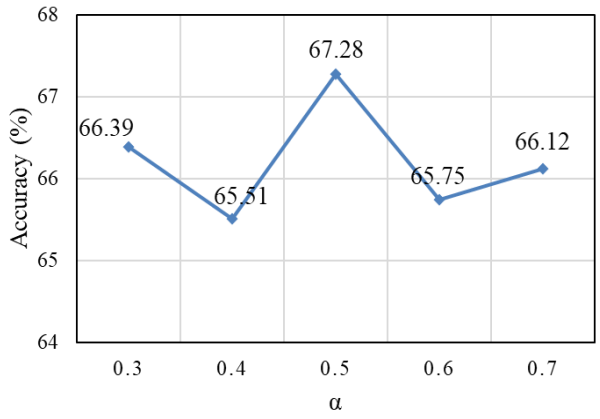


Figure 4: The figure illustrates the impact of the parameter α in the MPMR. The optimal training performance is achieved when α=0.5.

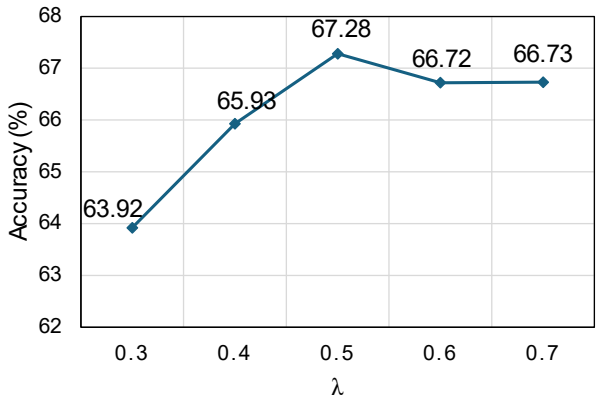


Figure 5: The figure illustrates the impact of the parameter λ in the CPR loss. The optimal training performance is achieved when λ=0.5.

B VISUALIZATION OF FEDERATED LEARNING WITH LONG-TAIL AND NOISY CLIENTS

Figure 6 presents a visualization of federated learning under long-tail and noisy client settings. We compare various methods and observe that our approach yields more detailed and focused attention maps. The visualized images are selected from the Tiny-ImageNet dataset with noise rates of 0.6 and 0.5, and an imbalance factor (IF) of 100.

C EXPERIENCE OF α PARAMETER IN MPMR

We investigated the effect of the α parameter in MPMR to understand its impact on the training process. The α parameter controls the balance between the original noisy data and the newly generated mask, influencing how the model learns from both sources. As shown in Figure 4, when (α= 0.5), the training benefits the most, indicating an optimal trade-off between preserving essential information from the original data and incorporating useful features from the generated mask. To further analyze this effect, we conducted experiments with different α values and observed their influence on model convergence, stability, and final accuracy. When α is too high, the model relies excessively on the noisy data, leading to suboptimal learning. Conversely, a low α causes the model to depend too much on the generated mask, potentially discarding valuable information. Our results

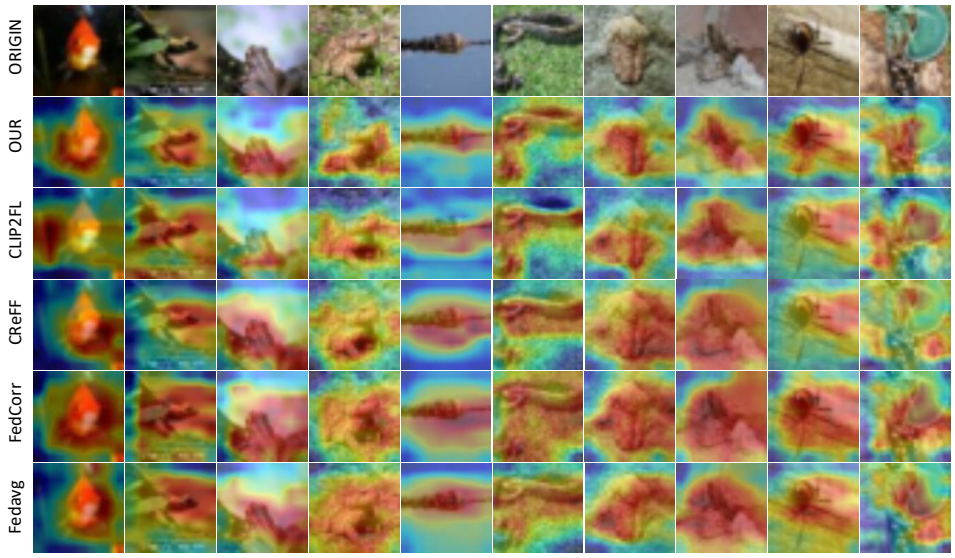


Figure 6: Visualization of different methods for comparison.

demonstrate that setting α to 0.5 achieves the best performance, ensuring a well-balanced learning process that enhances both robustness and generalization.

D EXPERIENCE OF λ PARAMETER IN CPR LOSS

The parameter λ in CPR loss plays a crucial role in our method, as it significantly influences the model's performance during training. To better understand its impact, we conducted extensive research on how to optimally set this parameter. As shown in Figure 5, we experimented with various values of λ and observed the model's response to each setting. By analyzing the trends, we found that the performance improves as λ is adjusted, with the best results consistently achieved when λ is set to 0.5. This indicates that λ is a key factor in balancing the efficiency of CPR loss, and fine-tuning it is essential for maximizing model performance. Our experiments demonstrate that choosing the optimal λ value helps in striking the right balance between the model's generalization ability and its attention to important features, leading to superior outcomes in training.

Algorithm 1 Training Process for Round t

Input: Training data $\mathcal{D} = \bigcup_{i=1}^{K} \mathcal{D}_i$, global model parameter $w^t = \{\theta^t, \phi^t\}$, re-train classifier ϕ'^t, random initialized federated features $\{F_{c,i}^t | c \in C^k\}|_{i=1}^m$
Output: A re-train model parameter $\hat{w}^{t+1} = \{\theta^{t+1}, \phi'^{t+1}\}$ for next round
// Server executes;
Send w^t to clients;
Random select a set of online clients; B^t
// Clients execute;
for $b \in B^t$ **do**
  // Do Prototype-Guide Refienment Learning (PGRL);
  // 1) Data Partition by PDS;
  Get cleaner class cluster center by Eq.1 and Eq.2;
  Divide D_b into clean dataset d_{clean} and noisy dataset d_{noisy} by PDV scores which are calculated by Eq.3 via GMM;
  // 2) Prototypical Contrast Refinement (PCR);
  // Refine noisy label to soft label for local training;
  for noisy prototype $x \in d_{noisy}$ **do**
    x compare with clean class prototypes by Eq.4;
    Refine the soft label $\bar{y}$ by Eq.5;
  end for
  Update local model w_k^{t+1} by Eq.11;
  // 3) Multi Prototypical Mask Refinement (MPMR);
  // Refine noisy prototypes and labels for class gradients training;
  for noisy prototype $x \in d_{noisy}$ **do**
    Obtain mask prototype $m(x)$ by Eq.6;
    Get Final corrected noisy data $\hat{x}$ by Eq.7;
    Convert soft label to hard label $sl(x)$ by Eq.8;
  end for
  Compute class gradients $\{g_b^c | c \in C^b\}$ by CE loss;
end for
// Server executes;
Aggregate model's parameter and class gradients' parameters by Eq.13 $\rightarrow w^{t+1}, g_{agg}^c$;
Update $w^{t+1} = \{\theta^{t+1}, \phi^{t+1}\}$ to the server;
Compute federated features g_f^c by Eq.14;
Optimize federated features to F^{t+1} by Eq.15;
Re-train the classifier ϕ'^t to ϕ'^{t+1} by Eq.16;
Send $\hat{w}^{t+1} = \{\theta^{t+1}, \phi'^{t+1}\}$ to clients;