FedDiv: Collaborative Noise Filtering for Federated Learning with Noisy Labels

Jichang Li[1,2], Guanbin Li[1,3*], Hui Cheng[4], Zicheng Liao[2,5], Yizhou Yu[2*]

[1]School of Computer Science and Engineering, Research Institute of Sun Yat-sen University in Shenzhen, Sun Yat-sen University, Guangzhou, China
[2]Department of Computer Science, The University of Hong Kong, Hong Kong
[3]Guangdong Province Key Laboratory of Information Security Technology
[4]UM-SJTU Joint Institute, Shanghai Jiao Tong University, Shanghai, China
[5]Zhejiang University, Hangzhou, China
csjcli@connect.hku.hk, liguanbin@mail.sysu.edu.cn, chenghui_123@sjtu.edu.cn
zichengliao@gmail.com, yizhouy@acm.org

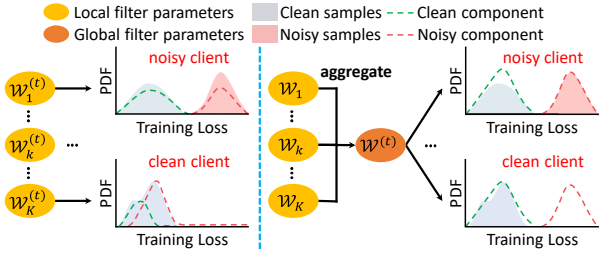
Abstract

Federated learning with noisy labels (F-LNL) aims at seeking an optimal server model via collaborative distributed learning by aggregating multiple client models trained with local noisy or clean samples. On the basis of a federated learning framework, recent advances primarily adopt label noise filtering to separate clean samples from noisy ones on each client, thereby mitigating the negative impact of label noise. However, these prior methods do not learn noise filters by exploiting knowledge across all clients, leading to sub-optimal and inferior noise filtering performance and thus damaging training stability. In this paper, we present `FedDiv` to tackle the challenges of F-LNL. Specifically, we propose a global noise filter called Federated Noise Filter for effectively identifying samples with noisy labels on every client, thereby raising stability during local training sessions. Without sacrificing data privacy, this is achieved by modeling the global distribution of label noise across all clients. Then, in an effort to make the global model achieve higher performance, we introduce a Predictive Consistency based Sampler to identify more credible local data for local model training, thus preventing noise memorization and further boosting the training stability. Extensive experiments on CIFAR-10, CIFAR-100, and Clothing1M demonstrate that `FedDiv` achieves superior performance over state-of-the-art F-LNL methods under different label noise settings for both IID and non-IID data partitions. Source code is publicly available at https://github.com/lijichang/FLNL-FedDiv.

Introduction

Compared to traditional Centralized Learning (Li et al. 2019b, 2021, 2019a; Wu et al. 2019b,a; Huang et al. 2023), Federated Learning (FL) is a novel paradigm facilitating collaborative learning across multiple clients without requiring centralized local data (McMahan et al. 2017). Recently, FL has shown significant real-world success in areas like healthcare (Nguyen et al. 2022), recommender systems (Yang et al. 2020), and smart cities (Zheng et al. 2022). However, these FL methods presume clean labels for all client's private data, which is often not the case in reality due to data complexity and uncontrolled label annotation quality (Tanno et al. 2019; Kuznetsova et al. 2020). Especially with the blessing of privacy protection, it is impossible to ensure absolute label accuracy. Therefore, this work centers on Federated Learning with Noisy Labels (F-LNL). In F-LNL, a global neural network model is fine-tuned via distributed learning across multiple local clients with noisy samples. Like (Xu et al. 2022), we here also assume some local clients have noisy labels (namely noisy clients), while others have only clean labels (namely clean clients).

Besides the private data on local clients, F-LNL faces two primary challenges: data heterogeneity and noise heterogeneity (Kim et al. 2022; Yang et al. 2022b). Data heterogeneity refers to the statistically heterogeneous data distributions across different clients, while noise heterogeneity represents the varying noise distributions among clients. It has been demonstrated by (Xu et al. 2022; Kim et al. 2022) that these two challenges in F-LNL may lead to instability during local training sessions. Previous studies in F-LNL (Xu et al.



(a) Local Noise Filter (b) Federated Noise Filter

Figure 1: (a) Local noise filtering may have limited capabilities as each client develops its own local noise filter using its own private data only. Especially on clean clients, such filters would incorrectly identify a subset of clean samples to be noisy. (b) Collaborative noise filtering proposed by us significantly improves the performance of label noise filtering on each client as a federated noise filter is learned by distilling knowledge from all clients. PDF: Probability density function.

*Corresponding Authors are Guanbin Li and Yizhou Yu.

2022; Kim et al. 2022) have demonstrated that many FL approaches, such as (McMahan et al. 2017; Li et al. 2020), are now in use to adequately address the first challenge. These approaches primarily focus on achieving training stability with convergence guarantees by aligning the optimization objectives of local updates and global aggregation. However, they do not address label noise on individual clients. Therefore, to tackle noise heterogeneity, such F-LNL algorithms propose separating noisy data into clean and noisy samples, occasionally complemented by relabeling the noisy samples. This aims to mitigate the negative effects of noisy labels and prevent local model overfitting to such label noise, avoiding severe destabilization of the local training process.

Some F-LNL methods (Kim et al. 2022; Xu et al. 2022) thus emphasize proposing local noise filtering, where each client develops its own noise filter to identify noisy labels. However, these noise filtering strategies omit the potential of learning prosperous knowledge from other clients to strengthen their capacities. Instead, they rely heavily on each client's own private data for training, thus leading to suboptimal and inferior performance. For example, as shown in Figure 1(a), limited training samples available on each client impede the accurate modeling of their local noise distributions, significantly restricting noise filtering capabilities. In addition, if noise filtering and relabeling are not handled properly, overfitting of noisy labels can inevitably occur, leading to noise memorization and thus degrading global performance upon model aggregation. Developing strategies to prevent noise memorization (Yang et al. 2022a) while enhancing training stability is critical, yet existing F-LNL algorithms have not succeeded in achieving this.

In this paper, we present a novel framework, `FedDiv`, to address the challenges of F-LNL. To perform label noise filtering per client, `FedDiv` consists of a global noise filter, called Federated Noise Filter (FNF), constructed by modeling the global distribution of noisy samples across all clients. Specifically, by fitting the loss function values of private data, the parameters of a local Gaussian Mixture Model (GMM) can be iteratively learned on each client to model its local noise distributions. These parameters are then aggregated on the server to construct a global GMM model that serves as a global noise filter, effectively classifying samples on each local client into clean or noisy. As depicted in Figure 1(b), by leveraging collaboratively learned knowledge across all clients, `FedDiv` demonstrates a robust capability to fit the local label noise distributions within individual clean or noisy clients. This ability enhances noise filtering performance per client, and thus reduces training instability during local training sessions, while preserving data privacy.

After label noise filtering, we remove noisy labels from the identified noisy samples on each client and relabel those samples exhibiting high prediction confidence using the pseudo-labels predicted by the global model. To further prevent local models from memorizing label noise and improve training stability, we introduce a Predictive Consistency based Sampler (PCS) for identifying credible local data for local model training. Specifically, we enforce the consistency of class labels predicted by both global and local models, and apply counterfactual reasoning (Holland 1986; Wang et al. 2022) to generate more reliable predictions for local samples.

In summary, the contributions of this paper are as follows.

- We propose `FedDiv`, a novel one-stage framework for federated learning with noisy labels. To enable stable training, `FedDiv` learns a global noise filter by distilling the complementary knowledge from all clients while performing label noise filtering locally on every client.
- We introduce a Predictive Consistency based Sampler to perform labeled sample re-selection on every client, thereby preventing local models from memorizing label noise and further improving training stability.
- Through extensive experiments conducted on CIFAR-10 (Krizhevsky 2009), CIFAR-100 (Krizhevsky 2009), and Clothing1M (Xiao et al. 2015) datasets, we demonstrate that `FedDiv` significantly outperforms state-of-the-art F-LNL methods under various label noise settings for both IID and non-IID data partitions.

Related Work

Centralized Learning with Noisy Labels (C-LNL). Diverging from conventional paradigms for centralized learning, e.g. (Li, Li, and Yu 2023a,b), which operates on training samples only with clean labels, several studies have demonstrated the effect of methods to address C-LNL in reducing model overfitting to noisy labels. For instance, `JointOpt` (Tanaka et al. 2018a) proposed a joint optimization framework to correct labels of noisy samples during training by alternating between updating model parameters and labels. As well, `DivideMix` (Li, Socher, and Hoi 2019) dynamically segregated training examples with label noise into clean and noisy subsets, incorporating auxiliary semi-supervised learning algorithms for further model training. Other strategies for handling C-LNL tasks include estimating the noise transition matrix (Cheng et al. 2022), reweighing the training data (Ren et al. 2018), designing robust loss functions (Englesson and Azizpour 2021), ensembling existing techniques (Li et al. 2022), and so on.

Given privacy constraints in decentralized applications, the server cannot directly access local samples of all clients to construct centralized noise filtering algorithms. Besides, a limited number of private data on local clients may also restrict the noise filtering capability. Hence, despite the success of existing C-LNL algorithms, they may no longer be feasible in federated settings (Xu et al. 2022).

Federated Learning with Noisy Labels. Numerous methods address challenges in federated scenarios with label noise. For instance, `FedRN` (Kim et al. 2022) detects clean samples in local clients using ensembled Gaussian Mixture Models (GMMs) trained to fit loss function values of local data assessed by multiple reliable neighboring client models. `RoFL` (Yang et al. 2022b) directly optimizes using small-loss instances of clients during local training to mitigate label noise effects. Meanwhile, `FedCorr`(Xu et al. 2022) first introduces a dimensionality-based noise filter to segregate clients into clean and noisy groups using local intrinsic dimensionalities (Ma et al. 2018), and trains local noise fil-

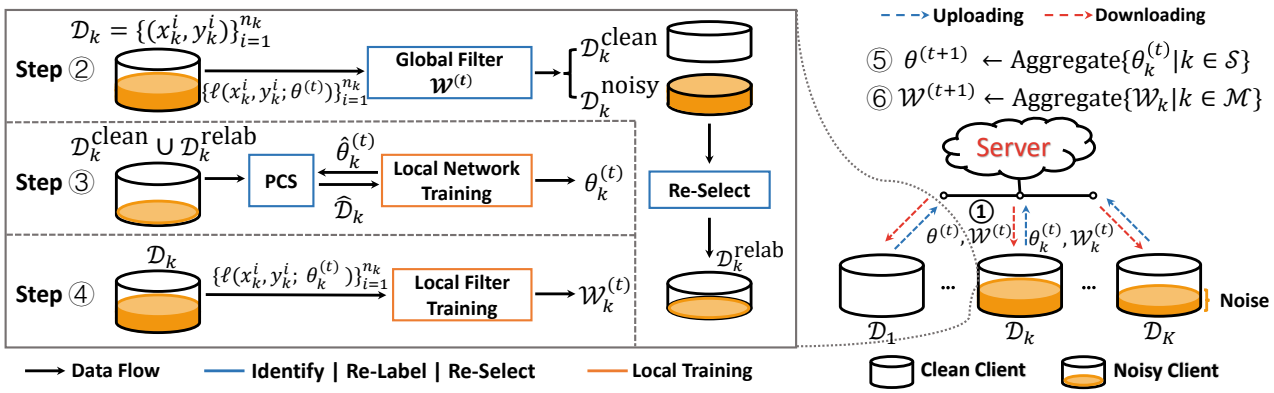


Figure 2: An overview of the training procedure proposed in FedDiv. In this work, the parameters of a local neural model and a local noise filter are simultaneously learned on each client during the local training sessions, while both types of parameters are aggregated on the server.

ters to separate clean examples from identified noisy clients based on per-sample training losses.

However, existing F-LNL algorithms concentrate on local noise filtering, utilizing each client's private data but failing to exploit collective knowledge across clients. This limitation might compromise noise filtering efficacy, resulting in incomplete label noise removal and impacting the stability of local training sessions. Conversely, FedDiv proposes distilling knowledge from all clients for federated noise filtering, enhancing label noise identification in each client's sample and improving FL model training amidst label noise.

Methodology

In this section, we introduce the proposed one-stage framework named FedDiv for federated learning with noisy labels. In detail, we first adopt the classic FL paradigm, namely FedAvg (McMahan et al. 2017), to train a neural network model. On the basis of this FL framework, we propose a global filter model called Federated Noise Filter (FNF) to perform label noise filtering and noisy sample relabeling on every client. Then, to improve the stability of local training, a Predictive Consistency based Sampler (PCS) is presented to conduct labeled sample re-selection, keeping client models from memorizing label noise.

Let us consider an FL scenario with one server and K local clients denoted by $\mathcal{M}$. Each client $k \in \mathcal{M}$ has its own private data consisting of n_k sample-label pairs $\mathcal{D}_k = \{(x_k^i, y_k^i)\}_{i=1}^{n_k}$, where x_k^i is a training sample, and y_k^i is a label index over C classes. Here, we divide local clients into two groups: clean clients (with noise level $\delta_k = 0$ where $k \in \mathcal{M}$), which only have samples with clean labels, and noisy clients (with $\delta_k > 0$), whose private data might have label noise at various levels. Also, in this work, both IID and non-IID heterogeneous data partitions are considered.

As shown in Figure 2, the training procedure of the t-th communication round performs the following steps:

Step ① : The server broadcasts the parameters of the global neural network model $\theta^{(t)}$ and the federated noise filtering model $\mathcal{W}^{(t)}$ to every client $k \in \mathcal{S}$, where $\mathcal{S} \subseteq \mathcal{M}$ is a subset of clients randomly selected with a fixed fraction $\omega = |\mathcal{S}|/K$ in this round.

Step ② : On every $k \in \mathcal{S}$, $\mathcal{W}^{(t)}$ is used to separate $\mathcal{D}_k$ into noisy and clean samples, and those noisy samples with high prediction confidence are assigned pseudo-labels predicted by $\theta^{(t)}$.

Step ③ (Local model training): Every client $k \in \mathcal{S}$ trains its local neural network using a subset of the clean and relabeled noisy samples selected using PCS to obtain its updated local parameters $\theta_k^{(t)}$. Here, we use $\hat{\theta}_k^{(t)}$ to denote the parameters of the local model being optimized during this training session.

Step ④ (Local filter training): Every client $k \in \mathcal{S}$ trains a local noise filter model with updated parameters $\mathcal{W}_k^{(t)}$ by fitting the per-sample loss function values of its private data $\mathcal{D}_k$. Such loss function values are evaluated using the logits of training samples in $\mathcal{D}_k$ predicted by $\theta_k^{(t)}$.

Step ⑤ : The server aggregates $\{\theta_k^{(t)} | k \in \mathcal{S}\}$ and then updates the global model as follows,

$$\theta^{(t+1)} \leftarrow \sum_{k \in \mathcal{S}} \frac{n_k}{\sum_{k \in \mathcal{S}} n_k} \theta_k^{(t)}. \tag{1}$$

Step ⑥ (Federated filter aggregation): The server collects $\{\mathcal{W}_k^{(t)} | k \in \mathcal{S}\}$ to update existing server-cached local filter parameters $\{\mathcal{W}_k | k \in \mathcal{S}\}$, and then aggregates all local filters $\{\mathcal{W}_k | k \in \mathcal{M}\}$ to obtain an updated federated noise filtering model $\mathcal{W}^{(t+1)}$.

This training procedure is repeated until the global model converges steadily or the pre-defined number of communication rounds $\mathcal{T}$ is reached. Each communication round involves local training sessions (Step ① - ④) on several randomly selected clients and the model aggregation phase (Step ⑤ - ⑥) on the server. The details of Step ② , Step ④ , and Step ⑥ are provided in "Federated Noise Filter", while the detailed description of Step ③ is given in "Predictive Consistency based Sampler" and "Objectives for Local Model Training".

Federated Noise Filter

Aiming at identifying label noise for the F-LNL task, we propose a Federated Noise Filter (FNF), which models the global distribution of clean and noisy samples across all clients. Motivated by (Zhu et al. 2022), this FNF model can be constructed via the federated EM algorithm. Specifically, we first conduct local filter training to obtain locally estimated GMM parameters. This goal is reached by iteratively executing the standard EM algorithm (Dempster, Laird, and Rubin 1977) per client to fit its local noise distribution. Then, we perform federated filter aggregation to aggregate local GMM parameters received from all clients to construct a federated noise filter.

Local filter training. In general, samples with label noise tend to possess higher loss function values during model training, making it feasible to use mixture models to separate noisy samples from clean ones using per-sample loss values of the training samples (Arazo et al. 2019a; Li, Socher, and Hoi 2019). Therefore, for the k-th client in the t-th communication round, a local GMM can be built to model the local distribution of clean and noisy samples by fitting the per-sample loss distribution,

$$\{\ell(x, y; \theta_k^{(t)}) | (x, y) \in \mathcal{D}_k\}, \tag{2}$$

where $\ell(x, y; \theta_k^{(t)})$ is the cross-entropy loss of a sample-label pair $(x, y) \in \mathcal{D}_k$ when the local model $\theta_k^{(t)}$ is used for prediction. Then, we denote the two-component GMM model by $\mathcal{W}_k^{(t)} = (\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\sigma}_k^{(t)}, \boldsymbol{\pi}_k^{(t)})$, where $\boldsymbol{\mu}_k^{(t)}$ and $\boldsymbol{\sigma}_k^{(t)}$ are vectors with entries $\mu_{kg}^{(t)}$ and $\sigma_{kg}^{(t)}$ denoting the mean and variance of the g-th Gaussian component, respectively. Here, we set $g = 1$ to represent the "clean" Gaussian component, i.e., the Gaussian component with a smaller mean (smaller loss), while $g = 2$ denotes the "noisy" one.

We further define a discrete variable z to represent whether a sample is clean or noisy. $\boldsymbol{\pi}_k^{(t)}$ denotes the prior distribution of z, i.e., $\pi_{kg}^{(t)} = P(z = g)$, where $\pi_{kg}^{(t)}$ should satisfy $\sum_{g=1}^{2} \pi_{kg}^{(t)} = 1$ and $0 \leq \pi_{kg}^{(t)} \leq 1$ for $g = 1, 2$. Thus, $P(\ell(x, y; \theta_k^{(t)}) | z = g)$ is modeled as a Gaussian distribution $\mathcal{N}(\ell(x, y; \theta_k^{(t)}); \mu_{kg}^{(t)}, \sigma_{kg}^{(t)})$. Then, the posterior $\gamma_{kg}(x, y; \theta_k^{(t)})$, which represents the probability of a sample x being clean ($g = 1$) or noisy ($g = 2$) given its loss value, can be computed as

$$\begin{aligned} \gamma_{kg}(x, y; \theta_k^{(t)}) &= P(z = g | x, y; \theta_k^{(t)}) \\ &= \frac{P(\ell(x, y; \theta_k^{(t)}) | z = g) P(z = g)}{\sum_{g'=1}^{2} P(\ell(x, y; \theta_k^{(t)}) | z = g') P(z = g')}. \end{aligned} \tag{3}$$

Afterwards, for each client k, leveraging its private data $\mathcal{D}_k$, updated local parameters $\theta_k^{(t)}$, and the federated filter parameters $\mathcal{W}^{(t)}$ received from the server, its optimal local filter parameters $\mathcal{W}_k^{(t)} = (\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\sigma}_k^{(t)}, \boldsymbol{\pi}_k^{(t)})$ at current round t are derived through the training of the local GMM models utilizing a standard EM algorithm (Dempster, Laird, and Rubin 1977). Noted that, $\mathcal{W}^{(t)}$ originated from the server here is used to initialize $\mathcal{W}_k^{(t)}$ for expediting convergence.

In the t-th communication round, once we have performed local filter training on the clients in $\mathcal{S}$, we upload the local filter parameters $\{\mathcal{W}_k^{(t)} | k \in \mathcal{S}\}$ to the server. Then, the server updates its cached local filter parameters corresponding to each client in $\mathcal{S}$ as follows,

$$\{\mathcal{W}_k \leftarrow \mathcal{W}_k^{(t)} | k \in \mathcal{S}\}, \tag{4}$$

where $\mathcal{W}_k$ is the server-cached version of the local noise filter on the k-th client. Note that `FedDiv` only sends three numerical matrices (i.e., $\boldsymbol{\mu}_k^{(t)}$, $\boldsymbol{\sigma}_k^{(t)}$, and $\boldsymbol{\pi}_k^{(t)}$) from each client to the server, and they merely reflect each client's local noise distributions instead of the raw input data, avoiding any risk of data privacy leakage.

Federated filter aggregation. After parameter uploading, the federated filter model can be constructed by aggregating the local filter parameters corresponding to all the clients $\{\mathcal{W}_k = (\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \boldsymbol{\pi}_k) | k \in \mathcal{M}\}$ as follows,

$$\begin{aligned} \mu_g^{(t+1)} &= \sum_{k \in \mathcal{M}} \frac{n_k}{\sum_{k \in \mathcal{M}} n_k} \mu_{kg}, \\ \sigma_g^{(t+1)} &= \sum_{k \in \mathcal{M}} \frac{n_k}{\sum_{k \in \mathcal{M}} n_k} \sigma_{kg}, \\ \pi_g^{(t+1)} &= \sum_{k \in \mathcal{M}} \frac{n_k}{\sum_{k \in \mathcal{M}} n_k} \pi_{kg}, \end{aligned} \tag{5}$$

where $\mathcal{W}^{(t+1)} = (\boldsymbol{\mu}^{(t+1)}, \boldsymbol{\sigma}^{(t+1)}, \boldsymbol{\pi}^{(t+1)})$ will be used to perform label noise filtering on the selected clients at the beginning of the $(t + 1)$-th communication round.

Label noise filtering. In the t-th communication round, once the k-th client has received the parameters of the global model $\theta^{(t)}$ and the federated filter model $\mathcal{W}^{(t)}$ from the server, the probability of a sample x from $\mathcal{D}_k$ being clean can be estimated through its posterior probability for the "clean" component as follows,

$$\mathbf{p}(\text{"clean"} | x, y; \theta_k^{(t)}) = P(z = 1 | x, y; \theta_k^{(t)}). \tag{6}$$

Afterwards, we can divide the samples of $\mathcal{D}_k$ into a clean subset $\mathcal{D}_k^{\text{clean}}$ and a noisy subset $\mathcal{D}_k^{\text{noisy}}$ by thresholding their probabilities of being clean with the threshold 0.5 as follows,

$$\begin{aligned} \mathcal{D}_k^{\text{clean}} &\leftarrow \{(x, y) \,|\mathbf{p}(\text{"clean"} | x, y; \theta_k^{(t)}) \geq 0.5, \forall\, (x, y) \in \mathcal{D}_k\}, \\ \mathcal{D}_k^{\text{noisy}} &\leftarrow \{(x,) \,|\mathbf{p}(\text{"clean"} | x, y; \theta_k^{(t)}) < 0.5, \forall\, (x, y) \in \mathcal{D}_k\}. \end{aligned} \tag{7}$$

Noisy sample relabeling. We compute the noise level of the k-th client as $\hat{\delta}_k = |\mathcal{D}_k^{\text{noisy}}| / |\mathcal{D}_k|$, while a client is considered a noisy one if $\hat{\delta}_k > 0.1$; and otherwise, a clean one. For an identified noisy client, we simply discard the given labels of noisy samples from $\mathcal{D}_k^{\text{noisy}}$ to prevent the model from memorizing label noise in further local training. In an effort to leverage these unlabeled (noisy) samples, we relabel those samples with high prediction confidence (by setting a confidence threshold ζ) by assigning predicted labels from the global model as follows,

$$\mathcal{D}_k^{\text{relab}} \leftarrow \{(x, \hat{y}) \,|\, \max(\mathbf{p}(x; \theta^{(t)})) \geq \zeta, \forall x \in \mathcal{D}_k^{\text{noisy}}\}, \tag{8}$$

where $\hat{y} = \hat{y}(x) = \arg\max \mathbf{p}(x; \theta^{(t)})$ is the pseudo-label for the sample x predicted by the global model $\theta^{(t)}$.

Method	Best Test Accuracy ± Standard Deviation							
	ρ=0.4		ρ=0.6		ρ=0.8		ρ=1.0	
	τ=0.0	τ=0.5	τ=0.0	τ=0.5	τ=0.0	τ=0.5	τ=0.0	τ=0.5
FedAvg (McMahan et al. 2017)	89.46±0.39	88.31±0.80	86.09±0.50	81.22±1.72	82.91±1.35	72.00±2.76	-	-
FedProx (Li et al. 2020)	88.54±0.33	88.20±0.63	85.80±0.41	85.25±1.02	84.17±0.77	80.59±1.49	-	-
RoFL (Yang et al. 2022b)	88.25±0.33	87.20±0.26	87.77±0.83	83.40±1.20	87.08±0.65	74.13±3.90	-	-
ARFL (Fu et al. 2019)	85.87±1.85	83.14±3.45	76.77±1.90	64.31±3.73	73.22±1.48	53.23±1.67	-	-
JointOpt (Tanaka et al. 2018a)	84.42±0.70	83.01±0.88	80.82±1.19	74.09±1.43	76.13±1.15	66.16±1.71	-	-
DivideMix (Li, Socher, and Hoi 2019)	77.35±0.20	74.40±2.69	72.67±3.39	72.83±0.30	68.66±0.51	68.04±1.38	-	-
FedCorr (Xu et al. 2022)	94.01±0.22	94.15±0.18	92.93±0.25	92.50±0.28	91.52±0.50	90.59±0.70	90.22±0.29§	69.01±3.71§
FedDiv (Ours)	**94.42±0.29**	**94.30±0.19**	**93.67±0.22**	**93.41±0.21**	**92.98±0.60**	**91.44±0.25**	**91.20±0.40**	**86.51±3.18**

Table 1: Best test accuracy (%) of FedDiv and existing SOTA methods on CIFAR-10 with IID setting at diverse noise levels. Algorithms marked with § indicate the re-implementations based on public code.

Method	Best Test Accuracy ± Standard Deviation		
	p = 0.7 α_{Dir} = 10	p = 0.7 α_{Dir} = 1	p = 0.3 α_{Dir} = 10
FedAvg (McMahan et al. 2017)	78.88±2.34	75.98±2.92	67.75±4.38
FedProx (Li et al. 2020)	83.32±0.98	80.40±0.94	73.86±2.41
RoFL (Yang et al. 2022b)	79.56±1.39	72.75±2.21	60.72±3.23
ARFL (Fu et al. 2019)	60.19±3.33	55.86±3.30	45.78±2.84
JointOpt (Tanaka et al. 2018a)	72.19±1.59	66.92±1.89	58.08±2.18
DivideMix (Li, Socher, and Hoi 2019)	65.70±0.35	61.68±0.56	56.67±1.73
FedCorr (Xu et al. 2022)	90.52±0.89	88.03±1.08	81.57±3.68
FedDiv (Ours)	**93.18±0.42**	**91.95±0.26**	**85.31±2.28**

Table 2: Best test accuracy (%) of FedDiv and existing SOTA methods on CIFAR-10 with non-IID setting at the noise level $(\rho, \tau) = (6.0, 0.5)$.

Method	Best Test Accuracy ± Standard Deviation			
	ρ=0.4 τ=0.5	ρ=0.6 τ=0.5	ρ=0.8 τ=0.5	ρ=1.0 τ=0.5
FedAvg (McMahan et al. 2017)	64.41±1.79	53.51±2.85	44.45±2.86	-
FedProx (Li et al. 2020)	65.09±1.46	57.51±2.01	51.24±1.60	-
RoFL (Yang et al. 2022b)	59.42±2.69	46.24±3.59	36.65±3.36	-
ARFL (Fu et al. 2019)	51.53±4.38	33.03±1.81	27.47±1.08	-
JointOpt (Tanaka et al. 2018a)	58.43±1.88	44.54±2.87	35.25±3.02	-
DivideMix (Li, Socher, and Hoi 2019)	43.25±1.01	40.72±1.41	38.91±1.25	-
FedCorr (Xu et al. 2022)	74.43±0.72	66.78±4.65	59.10±5.12	29.49±5.39§
FedDiv (Ours)	**74.86±0.91**	**72.37±1.12**	**65.49±2.20**	**45.95±5.36**

Table 3: Best test accuracy (%) of FedDiv and existing SOTA methods on CIFAR-100 with IID setting at diverse noise levels.

Method\(p, α_{Dir})	(0.7, 10)
FedAvg (McMahan et al. 2017)	64.75±1.75
FedProx (Li et al. 2020)	65.72±1.30
RoFL (Yang et al. 2022b)	59.31±4.14
ARFL (Fu et al. 2019)	48.03±4.39
JointOpt (Tanaka et al. 2018a)	59.84±1.99
DivideMix (Li, Socher, and Hoi 2019)	39.76±1.18
FedCorr (Xu et al. 2022)	72.73±1.02
FedDiv (Ours)	**74.47±0.34**

Table 4: Best test accuracy (%) of FedDiv and existing SOTA methods on CIFAR-100 with non-IID setting at the noise level $(\rho, \tau) = (4.0, 0)$.

Predictive Consistency based Sampler

During the t-th round on client k, upon obtaining the clean subset $\mathcal{D}k^{\text{clean}}$ and the relabeled subset $\mathcal{D}k^{\text{relab}}$, we integrate them into supervised local model training across T local epochs. However, the complete elimination of label noise among clients during noise filtering and relabeling is unattainable. On the other hand, relabeling inevitably introduces new label noise, causing instability in local model training, which further negatively affects the performance of the global model during aggregation. To tackle this, we propose a Predictive Consistency based Sampler (PCS) to reselect labeled samples for local training. Specifically, we observe enforcing the consistency of class labels respectively predicted by global and local models is a good strategy to achieve this goal. As training proceeds, the robustness of the model against label noise would be significantly increased (See below.), thus easily improving predictions' reliability of the samples having new label noise.

In addition, due to data heterogeneity in federated settings, especially for non-IID data partitions, local training samples owned by individual clients often belong to a smaller set of dominant classes. Thus, samples of dominant classes with newly introduced label noise would be better self-corrected during local model training, gradually leading to inconsistent predictions w.r.t those of the global model. However, such class-unbalanced local data would also contribute to the cause of the local model bias towards the dominant classes (Wei et al. 2021), which makes it more difficult for the local model to produce correct pseudo-labels for the samples that belong to minority classes. The proposed PCS strategy mitigates model bias to improve the reliability of class labels produced by local models. Here, we can de-bias the model predictions by improving causality via counterfactual reasoning (Holland 1986; Pearl 2009; Wang et al. 2022), and therefore, the de-biased logit of a sample x from $\mathcal{D}_k^{\text{clean}}$ or $\mathcal{D}_k^{\text{relab}}$ is induced as follows,

$$F(x) \leftarrow f(x; \hat{\theta}_k^{(t)}) - \xi \log(\hat{p}_k), \tag{9}$$

where $F(x)$ is the de-biased logit later used for generating the de-biased pseudo-label, i.e., $\tilde{y}(x) = \arg\max F(x)$, and $\xi = 0.5$ is a de-bias factor. $f(x; \hat{\theta}_k^{(t)})$ is the original logit for the sample x produced by the local model $\hat{\theta}_k^{(t)}$, currently being optimized. $\hat{p}_k$ represents the overall bias of the local model w.r.t all classes, which was previously updated according to Eq. (12) and cached on the k-th client during the last local training session.

Afterwards, PCS is used to re-select higher-quality and more reliable labeled training samples to perform local train-

RoFL (Yang et al. 2022b)	ARFL (Fu et al. 2019)	JointOpt (Tanaka et al. 2018a)	Dividemix (Li, Socher, and Hoi 2019)	FedCorr (Xu et al. 2022)	FedDiv (Ours)
70.39	70.91	71.78	68.83	72.55	**72.96±0.43**

Table 5: Best test accuracy (%) of FedDiv and existing SOTA methods on Clothing1M with non-IID setting.

ing as follows,

$$\bar{\mathcal{D}}_k \leftarrow \{(x,y)\,|\hat{y}(x)=\tilde{y}(x), \forall (x,y)\in \mathcal{D}_k^{\text{clean}} \cup \mathcal{D}_k^{\text{relab}}\}. \quad (10)$$

We update the training dataset for further optimizing the local model as follows,

$$\hat{\mathcal{D}}_k = \begin{cases} \bar{\mathcal{D}}_k, & \text{if } \hat{\delta}_k \geq 0.1, \\ \mathcal{D}_k, & \text{if } \hat{\delta}_k < 0.1. \end{cases} \quad (11)$$

Once having the optimized local model $\theta_k^{(t)}$ from a local training session, we use it to update $\hat{p}_k$ with momentum as follows,

$$\hat{p}_k^{(t)} \leftarrow m\hat{p}_k + (1-m)\frac{1}{n_k}\sum_{x\in\mathcal{D}_k} \mathbf{p}(x;\theta_k^{(t)}), \quad (12)$$

where $m = 0.2$ is a momentum coefficient. We save $\hat{p}_k^{(t)}$ on the k-th client to update existing client-cached $\hat{p}_k$.

Objectives for Local Model Training

To enhance the model's robustness against label noise, we here use MixUp regularization (Zhang et al. 2018) for local model training, further undermining the instability of training. Specifically, two sample-label pairs (x_i, y_i) and (x_j, y_j) from $\hat{\mathcal{D}}_k$ are augmented using linear interpolation, $\ddot{x} = \lambda x_i + (1-\lambda)x_j$ and $\ddot{y} = \lambda p_y(y_i) + (1-\lambda)p_y(y_j)$, where $\lambda \sim Beta(\alpha)$ is a mixup ratio, $\alpha = 1$ is a scalar to control its distribution, and $p_y(\cdot)$ is a function to generate a one-hot vector for a given label. Hence, on the k-th client in the t-th communication round, the local model $\hat{\theta}_k^{(t)}$ is trained with the cross-entropy loss applied to B augmented samples in one mini-batch as follows,

$$\mathcal{L}_{mix} = -\sum_{b=1}^{B} \ddot{y}_b \log \mathbf{p}(\ddot{x}_b; \hat{\theta}_k^{(t)}). \quad (13)$$

With the high heterogeneity of the non-IID data partitions, there could be only a limited number of categories on each client, and extensive experiments have shown that such a data partition forces a local model to predict the same class label to minimize the training loss. As in (Tanaka et al. 2018b; Arazo et al. 2019b), regularizing the average prediction of a local model over every mini-batch using a uniform prior distribution is a viable solution to overcome the above problem, i.e.,

$$\mathcal{L}_{reg} = \sum_{c=1}^{C} \hat{\mathbf{q}}^c \log\left(\frac{\hat{\mathbf{q}}^c}{\mathbf{q}^c}\right), \text{ where } \mathbf{q} = \frac{1}{B}\sum_{b=1}^{B} \mathbf{p}(\ddot{x}_b; \hat{\theta}_k^{(t)}), \quad (14)$$

where $\hat{\mathbf{q}}^c = 1/C$ denotes the prior probability of a class c. $\mathbf{q}^c$ is the c-th element of the vector $\mathbf{q}$, which refers to the predicted probability of the class c averaged over B augmented training samples in a mini-batch.

Finally, on the k-th client in the t-th communication round, the overall loss function for optimizing the local model is defined as follows,

$$\mathcal{L}_{final} = \mathcal{L}_{mix} + \eta\mathcal{L}_{reg}, \quad (15)$$

where η is a weighting factor balancing $\mathcal{L}_{mix}$ and $\mathcal{L}_{reg}$. In the experiments, we set $\eta = 1$ when the data partition is non-IID; and otherwise, $\eta = 0$.

Experiments

Experimental Setups

To be fair, we here adopt the consistent experimental setups with FedCorr (Xu et al. 2022) to assess the efficacy of our proposed approach FedDiv. Further details, e.g., data partitions, implementation, and baseline models, are provided in the supplementary document (abbr. **Supp**; see below), which can also be found at https://github.com/lijichang/FLNL-FedDiv/blob/main/supp.pdf.

Datasets and data partitions. We validate FedDiv's superiority on three classic benchmark datasets, including two synthetic datasets namely CIFAR-10 (Krizhevsky 2009) and CIFAR-100 (Krizhevsky 2009), and one real-world noisy dataset, i.e., Clothing1M (Xiao et al. 2015). Like (Xu et al. 2022), we take both IID and non-IID data partitions into account on CIFAR-10 and CIFAR-100, but only consider non-IID data partitions on Clothing1M. Under the IID data partitions, each client is randomly assigned the same number of samples with respect to each class. For non-IID data partitions, it is constructed using a Dirichlet distribution (Lin et al. 2020) with two pre-defined parameters, namely the fixed probability p and the concentration parameter α_{Dir}; See **Supp** for more details.

Label noise settings. Similar to (Xu et al. 2022), all clients are separated into a subset of clean clients with only clean samples and a subset of noisy clients potentially containing samples with label noise at varying degrees. The noise level, δ_k, for the k-th client is defined as follows:

$$\delta_k = \begin{cases} u \sim U(\tau, 1), & \text{probability of } \rho, \\ 0, & \text{probability of } 1-\rho. \end{cases} \quad (16)$$

Here, ρ signifies the probability of a client being noisy. For a noisy client with $\delta_k \neq 0$, the noise level is initially sampled at random from the uniform distribution $u \sim U(\tau, 1)$, with τ being its lower bound. Subsequently, $\delta_k \cdot 100\%$ of local examples are randomly selected as noisy samples, with their ground-truth labels replaced by all possible class labels.

Implementation. We set ω and $\mathcal{T}$ to 950, 900 and 100, and 0.1, 0.1 and 0.02 on CIFAR-10, CIFAR-100 and Clothing1M, respectively, while we also set the confidence threshold $\zeta = 0.75$ for relabeling on all datasets. Note that, to enable faster convergence, we warm up local neural network models of each client for $\mathcal{T}_{wu}$ iterations (not training rounds;

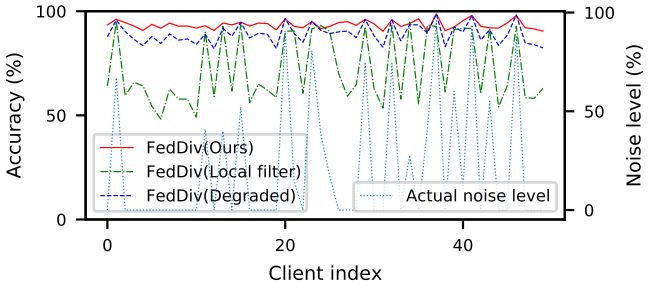


Figure 3: The accuracy of noisy label identification across different clients. The lightblue dotted line represents the actual noise level of each client, while the brightly colored dotted lines indicate the noise filtering performance w.r.t different noise filters. The experiment is conducted on CIFAR-100 with the non-IID data partition under the noise setting $(\rho, \tau) = (0.4, 0.0)$. (**Best viewed zoomed in.**)

see (Xu et al. 2022)) using the standard cross-entropy loss. Additionally, to be fair, most of our implementation details involving both local training and model aggregation are consistent with `FedCorr` (Xu et al. 2022) for each dataset under all federated settings and all label noise settings. For more implementation details, please see **Supp**.

Model variants. We build the variants of `FedDiv` to evaluate the effect of the proposed noise filter as follows.

- `FedDiv(Degraded)`: Following (Zhu et al. 2022), we here degrade the proposed federated noise filter by constructing the global noise filter using only the local filter parameters received in the current round instead of those from all clients.
- `FedDiv(Local filter)`: A local noise filter is trained for each client using its own private data to identify noisy labels within individual clients.

Comparison with State-of-the-Arts

Tables 1-5 summarize classification performance of `FedDiv` against state-of-the-art (SOTA) F-LNL methods on CIFAR-10, CIFAR-100, and Clothing1M across various noise settings for both IID and non-IID data partitions. Comparison results, based on mean accuracy and standard deviation over five trials, demonstrate `FedDiv`'s significant superiority over existing F-LNL algorithms, especially in challenging cases. For instance, in IID data partitions, Table 3 illustrates `FedDiv` outperforming `FedCorr` on CIFAR-100 by 6.39% and 16.46% in the toughest noise settings: $(\rho, \tau) = (0.8, 0.5)$ and $(\rho, \tau) = (1.0, 0.5)$. Similarly, for non-IID partitions in Table 2, `FedDiv` consistently surpasses `FedCorr` by 3.74% in the most challenging setting $(p, \alpha_{Dir}) = (0.3, 10)$ on CIFAR-10. Additionally, in Table 5, `FedDiv` exhibits a 0.41% improvement over `FedCorr` on Clothing1M, indicating its efficacy in real-world label noise distributions.

Ablation Analysis

To underscore the efficacy of `FedDiv`, we perform an ablation study to demonstrate the effect of each component.

Dataset	**CIFAR-10**		**CIFAR-100**	
Noise level (ρ, τ)	(0.8, 0.0)	(0.6, 0.5)	(0.8, 0.5)	(0.4,0.0)
Method $\backslash (p, \alpha_{Dir})$	†	(0.3, 10)	†	(0.7, 10)
FedCorr (Xu et al. 2022)	91.52±0.50	81.57±3.68	59.10±5.12	72.73±1.02
FedDiv (Ours)	92.98±0.60	85.31±2.28	65.49±2.20	74.47±0.34
FedDiv (Degraded)	92.31±0.61	83.22±2.61	62.23±2.37	73.06±0.93
FedDiv (Local filter)	91.14±0.48	81.34±3.65	60.97±1.65	71.37±0.76
FedDiv w/o Relab. & w/o PCS	91.43±0.87	82.17±3.06	61.45±3.43	73.09±1.89
FedDiv w/o PCS	92.29±0.96	82.83±2.59	62.67±2.51	73.66±0.96
FedDiv w/o PCS.debias	92.54±0.72	84.19±3.31	64.18±2.89	74.02±0.65
FedDiv w/o $\mathcal{L}_{reg}$	-	83.60±3.65	-	72.43±1.29

Table 6: Ablation study results for CIFAR-10 and CIFAR-100, in varying noise levels and both IID and non-IID data partition settings, with † indicating the IID data partitions.

More analysis results can be found in **Supp**.

Evaluation of federated noise filtering. To affirm the superiority of the proposed scheme for label noise filtering, we first compare `FedDiv` with our model variants `FedDiv(Local filter)` and `FedDiv(Degraded)`. As per Figure 3 and Table 6, the proposed noise filter exhibits a superior capacity of identifying label noise on both clean and noisy clients, leading to considerably improved classification performance in comparison to its two variants.

Evaluation of relabeling and re-selection. To assess the efficacy of the proposed strategies for noisy sample relabeling and labeled sample re-selection, we systematically remove their respective components from the `FedDiv` framework. The results depicted in Table 6 demonstrate a substantial decrease in accuracy across various noise settings for both types of data partitions. This indicates the importance of each individual component.

Conclusions

In this paper, we have presented `FedDiv` to deal with federated learning with noisy labels (F-LNL). It can effectively respond to the challenges in F-LNL tasks involving both data heterogeneity and noise heterogeneity while taking privacy concerns into account. On the basis of an FL framework, we first propose Federated Noise Filtering to separate clean samples from noisy ones on each client, thereby diminishing the instability during training. Then we perform relabeling to assign pseudo-labels to noisy samples with high predicted confidence. In addition, we introduce a Predictive Consistency based Sampler to identify credible local data for local model training, thus avoiding label noise memorization and further improving training stability. Experiments as well as comprehensive ablation analysis have revealed `FedDiv`'s superiority in handling F-LNL tasks.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (NO. 62322608), in part by the Shenzhen Science and Technology Program (NO. JCYJ20220530141211024), in part by the Open Project Program of the Key Laboratory of Artificial Intelligence for Perception and Understanding, Liaoning Province (AIPU, No. 20230003), in part by Hong Kong Research

Grants Council under Collaborative Research Fund (Project No. HKU C7004-22G).

References

Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.; and McGuinness, K. 2019a. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, 312–321. PMLR.

Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.; and McGuinness, K. 2019b. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, 312–321. PMLR.

Cheng, D.; Liu, T.; Ning, Y.; Wang, N.; Han, B.; Niu, G.; Gao, X.; and Sugiyama, M. 2022. Instance-Dependent Label-Noise Learning with Manifold-Regularized Transition Matrix Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16630–16639.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22.

Englesson, E.; and Azizpour, H. 2021. Generalized jensen-shannon divergence loss for learning with noisy labels. *Advances in Neural Information Processing Systems*, 34: 30284–30297.

Fu, S.; Xie, C.; Li, B.; and Chen, Q. 2019. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Holland, P. W. 1986. Statistics and causal inference. *Journal of the American statistical Association*, 81(396): 945–960.

Huang, D.; Li, J.; Chen, W.; Huang, J.; Chai, Z.; and Li, G. 2023. Divide and Adapt: Active Domain Adaptation via Customized Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7651–7660.

Kim, S.; Shin, W.; Jang, S.; Song, H.; and Yun, S.-Y. 2022. FedRN: Exploiting k-Reliable Neighbors Towards Robust Federated Learning. *arXiv preprint arXiv:2205.01310*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*.

Kuznetsova, A.; Rom, H.; Alldrin, N.; Uijlings, J.; Krasin, I.; Pont-Tuset, J.; Kamali, S.; Popov, S.; Malloci, M.; Kolesnikov, A.; et al. 2020. The open images dataset v4. *International Journal of Computer Vision*, 128(7): 1956–1981.

Li, J.; Li, G.; Liu, F.; and Yu, Y. 2022. Neighborhood Collective Estimation for Noisy Label Identification and Correction. In *European Conference on Computer Vision*. Springer.

Li, J.; Li, G.; Shi, Y.; and Yu, Y. 2021. Cross-Domain Adaptive Clustering for Semi-Supervised Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2505–2514.

Li, J.; Li, G.; and Yu, Y. 2023a. Adaptive Betweenness Clustering for Semi-Supervised Domain Adaptation. *IEEE Transactions on Image Processing*.

Li, J.; Li, G.; and Yu, Y. 2023b. Inter-Domain Mixup for Semi-Supervised Domain Adaptation. *Pattern Recognition*.

Li, J.; Socher, R.; and Hoi, S. C. 2019. DivideMix: Learning with Noisy Labels as Semi-supervised Learning. In *International Conference on Learning Representations*.

Li, J.; Wu, S.; Liu, C.; Yu, Z.; and Wong, H.-S. 2019a. Semi-supervised deep coupled ensemble learning with classification landmark exploration. *IEEE Transactions on Image Processing*, 29: 538–550.

Li, L.; Wang, J.; Li, J.; Ma, Q.; and Wei, J. 2019b. Relation classification via keyword-attentive sentence mechanism and synthetic stimulation loss. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9): 1392–1404.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2: 429–450.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33: 2351–2363.

Ma, X.; Wang, Y.; Houle, M. E.; Zhou, S.; Erfani, S.; Xia, S.; Wijewickrema, S.; and Bailey, J. 2018. Dimensionality-driven learning with noisy labels. In *International Conference on Machine Learning*, 3355–3364. PMLR.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Nguyen, D. C.; Pham, Q.-V.; Pathirana, P. N.; Ding, M.; Seneviratne, A.; Lin, Z.; Dobre, O.; and Hwang, W.-J. 2022. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(3): 1–37.

Pearl, J. 2009. Causal inference in statistics: An overview. *Statistics surveys*, 3: 96–146.

Ren, M.; Zeng, W.; Yang, B.; and Urtasun, R. 2018. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, 4334–4343. PMLR.

Tanaka, D.; Ikami, D.; Yamasaki, T.; and Aizawa, K. 2018a. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5552–5560.

Tanaka, D.; Ikami, D.; Yamasaki, T.; and Aizawa, K. 2018b. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5552–5560.

Tanno, R.; Saeedi, A.; Sankaranarayanan, S.; Alexander, D. C.; and Silberman, N. 2019. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11244–11253.

Wang, X.; Wu, Z.; Lian, L.; and Yu, S. X. 2022. Debiased Learning from Naturally Imbalanced Pseudo-Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14647–14657.

Wei, C.; Sohn, K.; Mellina, C.; Yuille, A.; and Yang, F. 2021. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10857–10866.

Wu, S.; Deng, G.; Li, J.; Li, R.; Yu, Z.; and Wong, H.-S. 2019a. Enhancing TripleGAN for semi-supervised conditional instance synthesis and classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10091–10100.

Wu, S.; Li, J.; Liu, C.; Yu, Z.; and Wong, H.-S. 2019b. Mutual learning of complementary networks via residual correction for improving semi-supervised classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 6500–6509.

Xiao, T.; Xia, T.; Yang, Y.; Huang, C.; and Wang, X. 2015. Learning from massive noisy labeled data for image classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2691–2699.

Xu, J.; Chen, Z.; Quek, T. Q.; and Chong, K. F. E. 2022. FedCorr: Multi-Stage Federated Learning for Label Noise Correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10184–10193.

Yang, E.; Yao, D.; Liu, T.; and Deng, C. 2022a. Mutual Quantization for Cross-Modal Search With Noisy Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7551–7560.

Yang, L.; Tan, B.; Zheng, V. W.; Chen, K.; and Yang, Q. 2020. Federated recommendation systems. In *Federated Learning*, 225–239. Springer.

Yang, S.; Park, H.; Byun, J.; and Kim, C. 2022b. Robust federated learning with noisy labels. *IEEE Intelligent Systems*, 37(2): 35–43.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.

Zheng, Z.; Zhou, Y.; Sun, Y.; Wang, Z.; Liu, B.; and Li, K. 2022. Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. *Connection Science*, 34(1): 1–28.

Zhu, C.; Xu, Z.; Chen, M.; Konečný, J.; Hard, A.; and Goldstein, T. 2022. Diurnal or Nocturnal? Federated Learning of Multi-branch Networks from Periodically Shifting Distributions. In *International Conference on Learning Representations*.

Supplementary Material (abbr. Supp)

Symbol	Definition
t	Current communication round
$\mathcal{M}$	Collection of all clients
$\mathcal{S}$	Collection of clients randomly selected at current round
k	Current index for the client selected from $\mathcal{M}$ or $\mathcal{S}$
$\mathcal{D}_k$	Given local samples for client k
$\mathcal{D}_k^{\text{clean}}$	Clean samples separated from $\mathcal{D}_k$
$\mathcal{D}_k^{\text{noisy}}$	Noisy samples separated from $\mathcal{D}_k$
$\mathcal{D}_k^{\text{relab}}$	Relabeled samples produced from $\mathcal{D}_k^{\text{noisy}}$
$\hat{\mathcal{D}}_k$	More reliable labeled samples re-selected by PCS
δ_k	Actual noise level for client k
$\hat{\delta}_k$	Estimated noise level for client k
$\theta^{(t)}$	Global network model at round t
$\hat{\theta}_k^{(t)}$	Local network model being optimized during local training for client k at round t
$\theta_k^{(t)}$	Local network model for client k at round t
$\mathcal{W}^{(t)}$	Global filter parameters for client k at round t
$\mathcal{W}_k$	Server-cached local filter parameters for client k
$\mathcal{W}_k^{(t)}$	Local filter parameters for client k obtained at round t
$\hat{p}_k$	Overall bias of the local model for client k w.r.t all classes

Table 7: Key notations of F-LNL and `FedDiv`.

In this supplementary material, we detail additional experimental setups encompassing data partitions, implementation specifics, and the baseline F-LNL methods. Furthermore, we provide further analysis of our proposed method, `FedDiv`. For a comprehensive understanding, we summarize the key notations of F-LNL and `FedDiv` in Table 7. Additionally, detailed training procedures of `FedDiv` and the specifics of local filter training are outlined in Algorithm 1 and Algorithm 2, respectively. To further enhance clarity, we present hyper-parameter summaries for each dataset in Table 8. Consistency in hyper-parameter settings is maintained across different label noise settings for both IID and non-IID data partitions on each dataset. It's worth noting that all experiments are conducted on the widely-used PyTorch platform[1] and executed on an NVIDIA GeForce GTX 2080Ti GPU with 12GB memory.

[1]https://pytorch.org/

Hyper-parameter	CIFAR-10	CIFAR-100	Clothing1M
# of clients (K)	100	50	500
# of classes (C)	10	100	14
# of samples	50,000	50,000	1,000,000
Architecture	ResNet-18	ResNet-34	Pre-trained ResNet-50
Mini-batch size	10	10	16
Learning rate	0.03	0.01	0.001
$\mathcal{T}_{wu}$	5	10	2
$\mathcal{T}_{ft}$	500	450	50
$\mathcal{T}_{ut}$	450	450	50
$\mathcal{T}$	950	900	100
T	5	5	5
ω	0.1	0.1	0.02

Table 8: Hyper-parameters on different datasets.

Additional Experimental Setups

Non-IID data partitions. We employ Dirichlet distribution (Lin et al. 2020) with the fixed probability p and the concentration parameter α_{Dir} to construct non-IID data partitions. Specifically, we begin by introducing an indicator matrix $\Phi \in \mathbb{R}^{C\times K}$, and each entry Φ_{ck} determines whether the k-th client has samples from the c-th class. For every entry, we assign a 1 or 0 sampled from the Bernoulli distribution with a fixed probability p. For the row of the matrix Φ that corresponds to the c-th class, we sample a probability vector $q_c \in \mathbb{R}^{Q_c}$ from the Dirichlet distribution with a concentration parameter $\alpha_{Dir} > 0$, where $Q_c = \sum_k \Phi_{ck}$. Then, we assign the k'-th client a $q_{ck'}$ proportion of the samples that belong to the c-th category, where k' denotes the client index with $\Phi_{ck} = 1, k = 1, \cdots, K$, and $\sum_{k'=1}^{|q_c|} q_{ck'} = 1$.

Additional Implementation Details. Similar to `FedCorr` (Xu et al. 2022), we select ResNet-18 (He et al. 2016), ResNet-34 (He et al. 2016) and Pre-trained ResNet-50 (He et al. 2016) as the network backbones for CIFAR-10, CIFAR-100 and Clothing1M, respectively. During the local model training sessions, we train each local client model over $T = 5$ local training epochs per communication round, using an SGD optimizer with a momentum of 0.5 and a mini-batch size of 10, 10, and 16 for CIFAR-10, CIFAR-100, and Clothing1M, respectively. For each optimizer, we set the learning rate as 0.03, 0.01, and 0.001 on CIFAR-10, CIFAR-100, and Clothing1M, respectively. In addition, during data pre-processing, the training samples are first normalized and then augmented by hiring random horizontal flipping and random cropping with padding of 4. For most thresholds conducted on the experiments, we set them to default as in `FedCorr` (Xu et al. 2022), e.g. $\hat{\delta}_k = 0.1$ in Eq. (11), the probability of a sample being clean/noisy = 0.50 in Eq. (7), $\xi = 0.5$ in Eq. (9), etc. Additionally, we determine ζ in Eq. (8) using a small validation set, where $\zeta = 0.70$ meets the peak of validation accuracies.

Baselines. We compare `FedDiv` with existing state-of-the-art (SOTA) F-LNL methods, including `FedAvg` (McMahan et al. 2017), `FedProx` (Li et al. 2020), `RoFL` (Yang et al.

Algorithm 1: The training procedure of `FedDiv`

Input: $\mathcal{M}$; $\{\mathcal{D}_k | k \in \mathcal{M}\}$; $\mathcal{T}$; T; ω; Initialized $\theta^{(0)}$; Initialized $\mathcal{W}^{(0)}$; Initialized $\{\mathcal{W}_k | k \in \mathcal{M}\}$; Initialized $\{\hat{p}_k | k \in \mathcal{M}\}$
Output: Global network model $\theta^{(\mathcal{T})}$

for $k \in \mathcal{M}$ **do**
  $[k] \leftarrow (\mathcal{D}_k, \hat{p}_k)$ ▷ Packaging
end for
// Model warming-up step
for $t = 1$ to $\mathcal{T}_{wu}$ **do**
  Warm up $\theta^{(t)}$ for each client $k \in \mathcal{M}$
end for
// Model training step
for $t = 1$ to $\mathcal{T}$ **do**
  $\mathcal{S} \leftarrow$ Randomly select $\omega \times 100\%$ clients from $\mathcal{M}$
  for $k \in \mathcal{S}$ **do**
    $(\mathcal{D}_k, \hat{p}_k) \leftarrow [k]$ ▷ Unpackaging
    Obtain $\mathcal{D}_k^{\text{clean}}$ and $\mathcal{D}_k^{\text{noisy}}$ via the federated noise filter model $\mathcal{W}^{(t)}$ using Eq. (7)
    Estimate $\hat{\delta}_k = |\mathcal{D}_k^{\text{noisy}}|/|\mathcal{D}_k|$
    Obtain $\mathcal{D}_k^{\text{relab}}$ using Eq. (8)
    $\hat{\theta}_k^{(t)} \leftarrow \theta^{(t)}$
    for $t' = 1$ to T **do**
      Obtain $\hat{\mathcal{D}}_k$ using Eqs. (9), (10), (11)
      Optimize $\hat{\theta}_k^{(t)}$ using Eq. (15)
    end for
    $\theta_k^{(t)} \leftarrow \hat{\theta}_k^{(t)}$
    Update $\hat{p}_k^{(t)}$ using Eq. (12)
    Update $[k] \leftarrow (\mathcal{D}_k, \hat{p}_k^{(t)})$ ▷ Repackaging
    Obtain the local filter $\mathcal{W}_k^{(t)}$ using Algorithm 2
    Upload $\theta_k^{(t)}$ and $\mathcal{W}_k^{(t)}$ to the server
  end for
  Update the network $\theta^{(t+1)}$ using Eq. (1)
  for $k \in \mathcal{S}$ **do**
    $\mathcal{W}_k \leftarrow \mathcal{W}_k^{(t)}$
  end for
  Update the global filter $\mathcal{W}^{(t+1)}$ using Eq. (5)
end for

Algorithm 2: Local filter training

Input: $\mathcal{D}_k$; $\theta_k^{(t)}$; $\mathcal{W}^{(t)}$
Output: Optimal local filter parameters $\mathcal{W}_k^{(t)}$

Initialize $\mathcal{W}_k^{(t)} = (\boldsymbol{\mu}_k^{(t)}, \boldsymbol{\sigma}_k^{(t)}, \boldsymbol{\pi}_k^{(t)})$ using $\mathcal{W}^{(t)}$
while $\mathcal{W}_k^{(t)}$ *is not converged* **do**
  // E step:
  $\gamma_{kg}(x, y; \theta_k^{(t)}) = \frac{\pi_{kg}^{(t)} \cdot \mathcal{N}(\ell(x,y;\theta_k^{(t)}); \mu_{kg}^{(t)}, \sigma_{kg}^{(t)})}{\sum_{g'=1}^{2} \pi_{kg'}^{(t)} \cdot \mathcal{N}(\ell(x,y;\theta_k^{(t)}); \mu_{kg'}^{(t)}, \sigma_{kg'}^{(t)})}$
  // M step:
  $\mu_{kg}^{(t)} = \frac{\sum_{(x,y)\in\mathcal{D}_k} \gamma_{kg}(x,y;\theta_k^{(t)}) \cdot \ell(x,y;\theta_k^{(t)})}{\sum_{(x,y)\in\mathcal{D}_k} \gamma_{kg}(x,y;\theta_k^{(t)})}$
  $\sigma_{kg}^{(t)} = \frac{\sum_{(x,y)\in\mathcal{D}_k} \gamma_{kg}(x,y;\theta_k^{(t)}) \cdot [\ell(x,y;\theta_k^{(t)}) - \mu_{kg}^{(t)}]^2}{\sum_{(x,y)\in\mathcal{D}_k} \gamma_{kg}(x,y;\theta_k^{(t)})}$
  $\pi_{kg}^{(t)} = \frac{\sum_{(x,y)\in\mathcal{D}_k} \gamma_{kg}(x,y;\theta_k^{(t)})}{n_k}$
end while

2022b), `ARFL` (Yang et al. 2022b), `JointOpt` (Tanaka et al. 2018a), `DivideMix` (Li, Socher, and Hoi 2019), `FedCorr` (Xu et al. 2022), borrowing the reported experimental results from (Xu et al. 2022). Specifically, `FedAvg` and `FedProx` are both classic FL algorithms, while `RoFL`, `ARFL` and `FedCorr` are three existing F-LNL methods. Besides, `DivideMix` and `JointOpt` are two exiting representative C-LNL algorithms, which are adapted by `FedCorr` into the FL scenarios.

How to set ω, $\mathcal{T}$ and $\mathcal{T}_{wu}$. In this work, we streamlined the multi-stage F-LNL process proposed in `FedCorr` (Xu et al. 2022) into a one-stage process, avoiding the complexity of executing multiple intricate steps across different stages as in `FedCorr`. However, for fair comparisons, we maintained an equivalent number of communication rounds as in `FedCorr`. This totals $\mathcal{T}_{wu}$, encompassing federated pre-processing from `FedCorr`'s training iterations, and $\mathcal{T}$, covering both federated fine-tuning and usual training stages involving `FedCorr`. Notably, within $\mathcal{T}_{wu}$, we solely utilize standard cross-entropy loss to warm up the local neural network models for faster convergence.

Below, we will begin by introducing the multi-stage F-LNL pipeline proposed by `FedCorr`, followed by an analysis of fraction scheduling and the construction of the training rounds of `FedDiv`.

- **`FedCorr`.** `FedCorr` comprises three FL stages: federated pre-processing, federated fine-tuning, and federated usual training. During the pre-processing stage, the FL model is initially pre-trained on all clients for $\mathcal{T}_{wu}$ *iterations* (not training round) to guarantee initial convergence of model training. At the same time, `FedCorr` evaluates the quality of each client's dataset and identifies and relabels noisy samples. After this stage, a dimensionality-based filter (Ma et al. 2018) is proposed to classify clients into clean and noisy ones. In the federated fine-tuning stage, `FedCorr` only fine-tunes the global model on relatively clean clients for $\mathcal{T}_{ft}$ rounds. At the end of this stage, `FedCorr` re-evaluates and relabels the remaining noisy clients. Finally, in the federated usual training stage, the global model is trained over $\mathcal{T}_{ut}$ rounds using `FedAvg` (McMahan et al. 2017) on all the clients, incorporating the labels corrected in the previous two training stages.
- **Fraction scheduling and communication rounds of `FedDiv`.** During FL training, a fixed fraction of clients will be selected at random to participate in local model training at the beginning of each round. Here, we set a fraction parameter ω to control the fraction scheduling, which is the same as the fine-tuning and usual training stages in `FedCorr`. However, during the pre-processing stage of `FedCorr`, every client must participate in local training exactly once in each iteration. Hence, any one client may be randomly sampled from all the clients with a probability of $\frac{1}{K} \cdot 100\%$ to participate in local model training, without replacement. As three stages of `FedCorr` have been merged into one in `FedDiv`, to en-

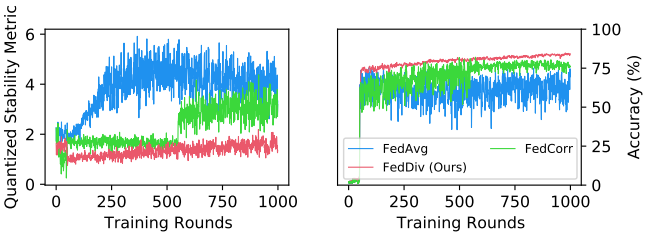


Figure 4: The evolution of quantized training stability *v.s.* test classification performance across epochs for various F-LNL algorithms. We quantitatively assess training stability in F-LNL by computing the average proximal regularization metric (Li et al. 2020; Xu et al. 2022) between the weights of local and global neural network models in the current training round. The experiments are conducted on CIFAR-10 with $(p, \alpha_{Dir}) = (0.3, 10.0)$ and $(\rho, \tau) = (6.0, 0.5)$.

sure fairness in training, we convert the training iterations of the pre-processing stage into the training rounds we used, which gives us the corresponding training rounds $\mathcal{T}'_{wu} = (\mathcal{T}_{wu} \times K)/(w \times K) = \mathcal{T}_{wu}/w$. Therefore, in our work, the total number of communication rounds in the entire training process is $\mathcal{T}'_{wu} + \mathcal{T}$, where $\mathcal{T} = \mathcal{T}_{ft} + \mathcal{T}_{ut}$.

Additional Analysis

Quantized training stability. To better grasp the motivation behind this approach, we propose using "Quantized training stability" to quantify the impact of data heterogeneity and noise heterogeneity (Kim et al. 2022; Yang et al. 2022b) on the training instability experienced during local training sessions. Technically, quantized training stability can be measured by the average proximal regularization metric between local and global model weights, denoted as $\theta_k^{(t)}$ and $\theta^{(t)}$ respectively, at round t. This is calculated by $\frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}}|\theta_k^{(t)} - \theta^{(t)}|^2$. As depicted in Figure 4, this instability results in notable discrepancies in weight divergence between local and global models, potentially hindering the performance enhancement of the aggregated model if left unaddressed. Additionally, considering the efficacy of different noise filtering strategies, our proposed federated noise filtering demonstrates superior performance in label noise identification per client, leading to decreased training instability during local training sessions and thus achieving higher classification performance of the aggregated model.

Further evaluation of federated noise filtering. To further verify the capability of our proposed label noise filtering strategy, we again compare `FedDiv` with `FedDiv(Local filter)` and `FedDiv(Degraded)` in Figure 5 and Figure 6. Both experiments are conducted on CIFAR-100 with the noise setting $(\rho, \tau) = (0.4, 0.5)$ for the IID data partition. Specifically, Figure 5 shows the accuracy of label noise filtering over all 50 clients at different communication rounds, while Figure 6 provides two examples to illustrate the noise filtering performance of different noise filters on clean and noisy clients.

As depicted in both Figure 5 and Figure 6, the proposed strategy consistently produces stronger label noise filtering capabilities on the vast majority of clients than the alternative solutions. Additionally, Figure 5 also shows that all these three noise filtering schemes significantly improve the label noise identification performance as model training proceeds, especially on clean clients—possibly because the network model offers greater classification performance—but ours continues to perform the best. These results once again highlight the feasibility of our proposed noise filtering strategy.

Further evaluation of filtering, relabeling and re-selection. To emphasize each `FedDiv` thread's effectiveness in label noise filtering, noisy sample relabeling, and labeled sample re-selection, we compare confusion matrices before processing, after label noise filtering and noisy sample relabeling (Thread 1), and after labeled sample re-selection (Thread 2) in Figure 7. Figure 7 displays heat maps of these three confusion matrices on five representative clients. On clean or noisy clients with varying noise levels, each thread gradually eliminates label noise, confirming the performance of each `FedDiv` component.

Hyper-parameter sensitivity. We analyze hyper-parameter sensitivity to the confidence threshold ζ. As shown in Figure 8, the proposed approach consistently achieves higher classification performance when ζ is set to 0.75. Therefore, $\zeta = 0.75$ is an excellent choice for setting the confidence threshold for noisy sample relabeling when training the FL model on CIFAR-10 and CIFAR-100 under different label noise settings for both IID and non-IID data partitions.

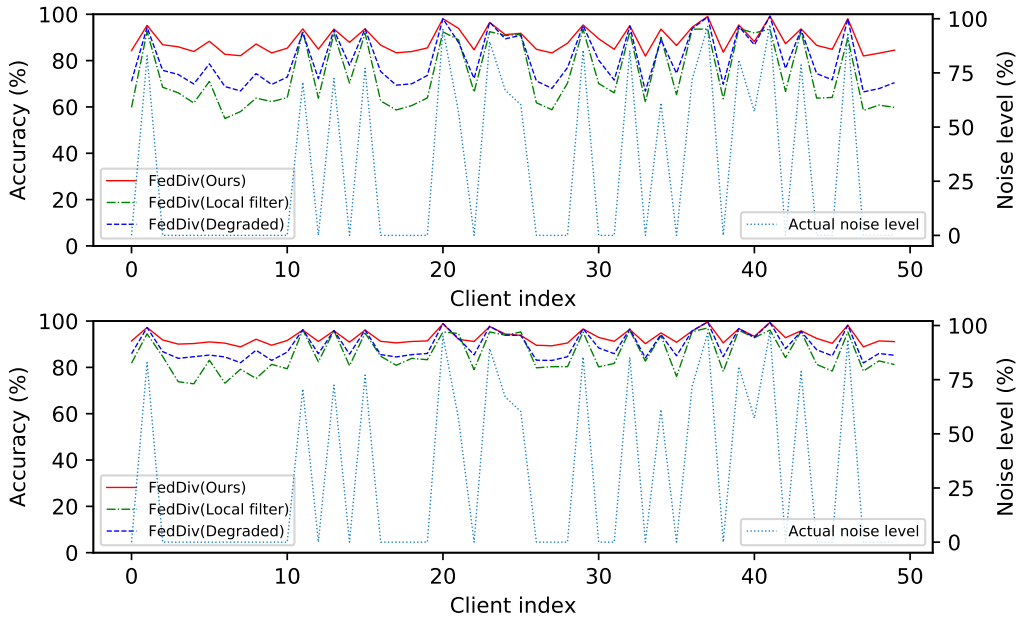


Figure 5: The accuracy of noisy label identification vs. different clients. The lightblue dotted line represents the actual noise level of each client, while the (dotted) lines in deep bright colors indicate the noise filtering performance with respect to different noise filters. The experiment is conducted on CIFAR-100 with the IID data partition under the noise setting $(\rho, \tau) = (0.4, 0.5)$. TOP: Evaluation in the 50-th communication round of the usual training stage; BOTTOM: Evaluation in the 500-th communication round of the usual training stage.

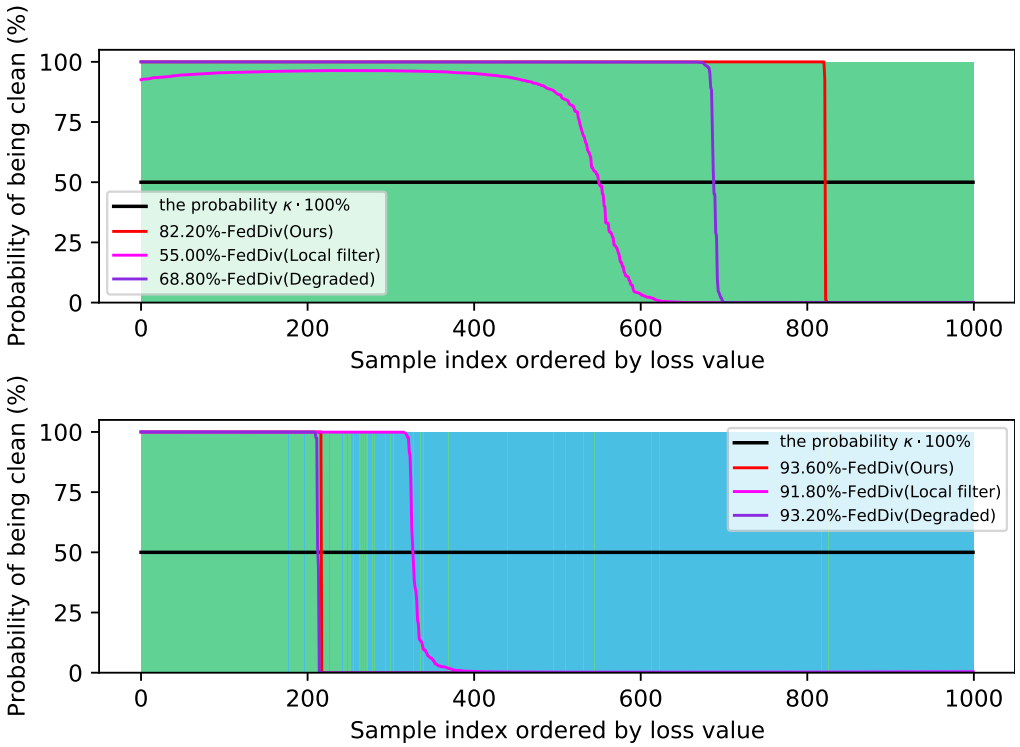


Figure 6: Two examples to illustrate the performance of three noise filters to identify label noise on a clean client (with the lowest accuracy of label noise filtering) and a noisy client. We show the probability distributions of samples being clean predicted by each filter, with the samples ranked according to the per-sample loss function values. In the legend, the percentages show the accuracy of noisy label identification with respect to each filter. In addition, as illustrated by the black line, a sample that is considered clean should have a predicted probability higher than $\kappa \cdot 100\%$. Furthermore, the green and blue bars represent, respectively, the distribution of the given **clean** and **noisy** samples. The evaluation is performed at the end of the 50-th communication round in the usual federated training stage. The experiment is conducted on CIFAR-100 with the IID data partition under the noise setting $(\rho, \tau) = (0.4, 0.5)$. TOP: Evaluation on the clean client; BOTTOM: Evaluation on the noisy client with $\delta = 0.725$. (**Best viewed zoomed in.**)

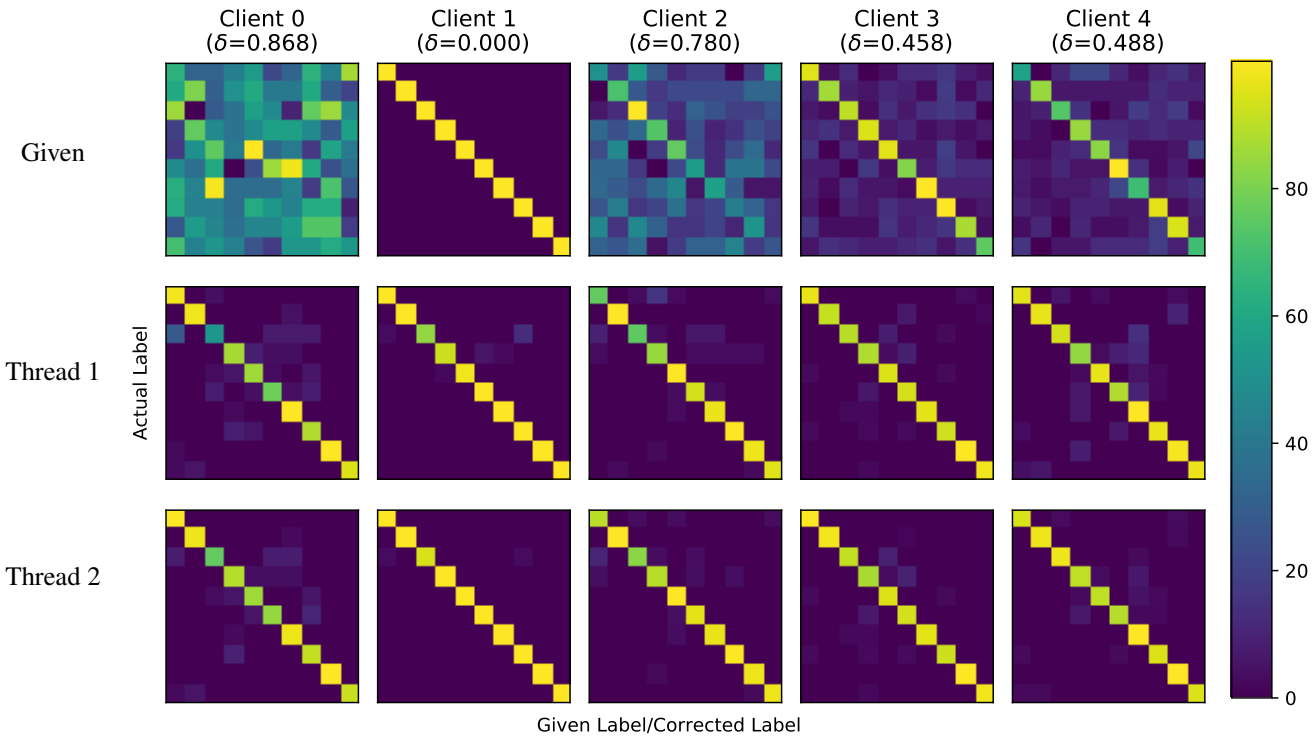
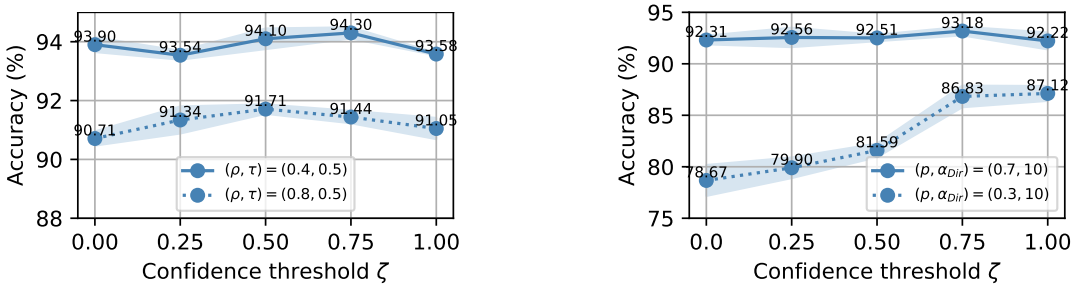


Figure 7: An evaluation of label noise filtering, noisy sample relabeling and labeled sample re-selection on five representative clients in the proposed `FedDiv`. As indicated by the heat maps, three confusion matrices for each client are associated to the actual labels v.s. the given labels before processing, the corrected labels after label noise filtering and noisy sample relabeling (named Thread 1), and the corrected labels after labeled sample re-selection (named Thread 2), respectively. Note that, in practice, noisy label relabeling and labeled sample re-selection may not necessarily be conducted on clean clients (e.g., Client 1) during local model training, in accordance with Eq. (10). The experiment is conducted on CIFAR-10 with the IID data partition under the noise setting $(\rho, \tau) = (0.8, 0.5)$.



(a) IID

(b) Non-IID

Figure 8: Sensitivity with respect to the hyper-parameter ζ. We show the test accuracy to illustrate the classification performance of the final FL model when we set ζ to 0.00, 0.25, 0.50, 0.75, and 1.00, respectively. We conduct these experiments on CIFAR-10 with both IID and non-IID data partitions. For the non-IID data partition, the noise level is set to $(\rho, \tau) = (0.6, 0.5)$.