FedDiv: Collaborative Noise Filtering for Federated Learning with Noisy Labels (Supplementary Document (abbr. Supp))

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

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 1. 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 2. 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.

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. 2022), `ARFL` (Yang et al. 2022), `JointOpt` (Tanaka et al. 2018), `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

---

[1]https://pytorch.org/

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 1: Key notations of F-LNL and `FedDiv`.

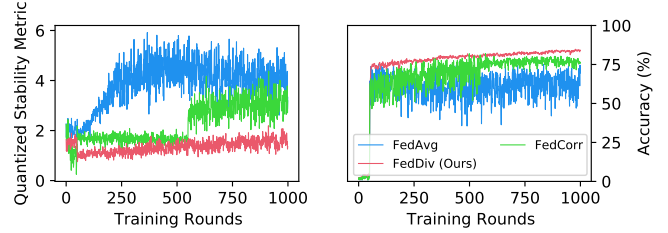| Hyper-parameter	CIFAR-10	CIFAR-100	Clothing1M
# of clients (K)	100	50	500
# of classes (C)	10	100	14
# of samples	50,000	50,000	1,000,000
Architecture	ResNet-18	ResNet-34	Pre-trained ResNet-50
Mini-batch size	10	10	16
Learning rate	0.03	0.01	0.001
$\mathcal{T}_{wu}$	5	10	2
$\mathcal{T}_{ft}$	500	450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

Table 2: Hyper-parameters on different datasets.

Figure 1: 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 $(\rho, \tau) = (6.0, 0.5)$.

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 ensure 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 train-

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 $\theta^{(\mathcal{T})}$

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

Algorithm 2: Local filter training

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

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

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 2 and Figure 3. Both experiments are conducted on CIFAR-100 with the noise setting $(\rho, \tau) = (0.4, 0.5)$ for the IID data partition. Specifically, Figure 2 shows the accuracy of label noise filtering over all 50 clients at different communication rounds, while Figure 3 provides two examples to illustrate the noise filtering performance of different noise filters on clean and noisy clients.

As depicted in both Figure 2 and Figure 3, the proposed strategy consistently produces stronger label noise filtering capabilities on the vast majority of clients than the alternative solutions. Additionally, Figure 2 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 4. Figure 4 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 5, 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

---

ing stability" to quantify the impact of data heterogeneity and noise heterogeneity (Kim et al. 2022; Yang et al. 2022) 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 1, 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.

model on CIFAR-10 and CIFAR-100 under different label noise settings for both IID and non-IID data partitions.

References

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.

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*.

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, 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.

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

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, S.; Park, H.; Byun, J.; and Kim, C. 2022. Robust federated learning with noisy labels. *IEEE Intelligent Systems*, 37(2): 35–43.
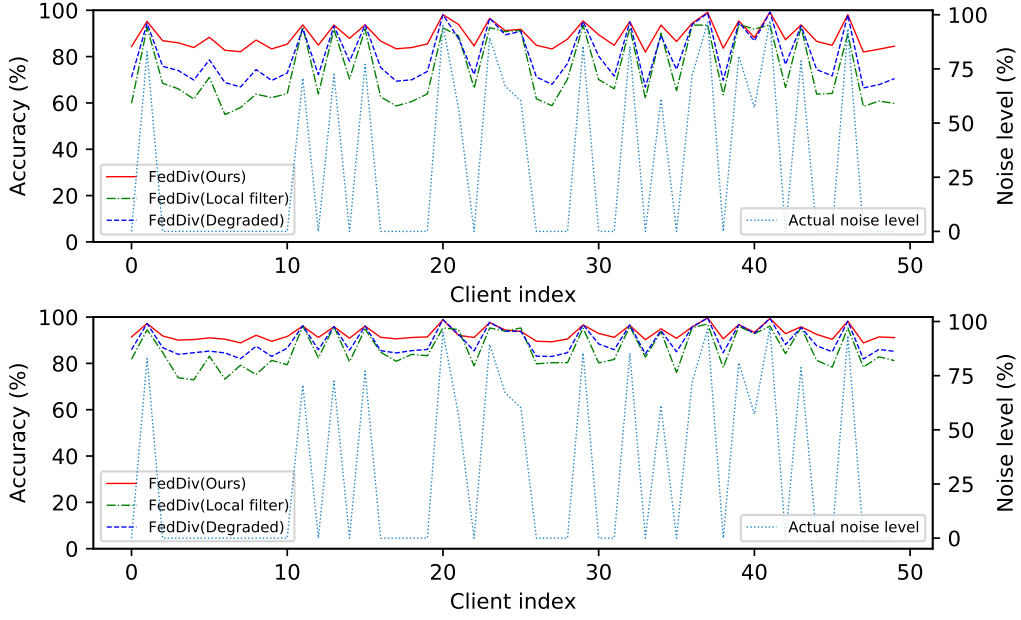
Figure 2: The accuracy of noisy label identification vs. different clients. The lightblue dotted line represents the actual noise level of each client, while the (dotted) lines in deep bright colors indicate the noise filtering performance with respect to different noise filters. The experiment is conducted 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.
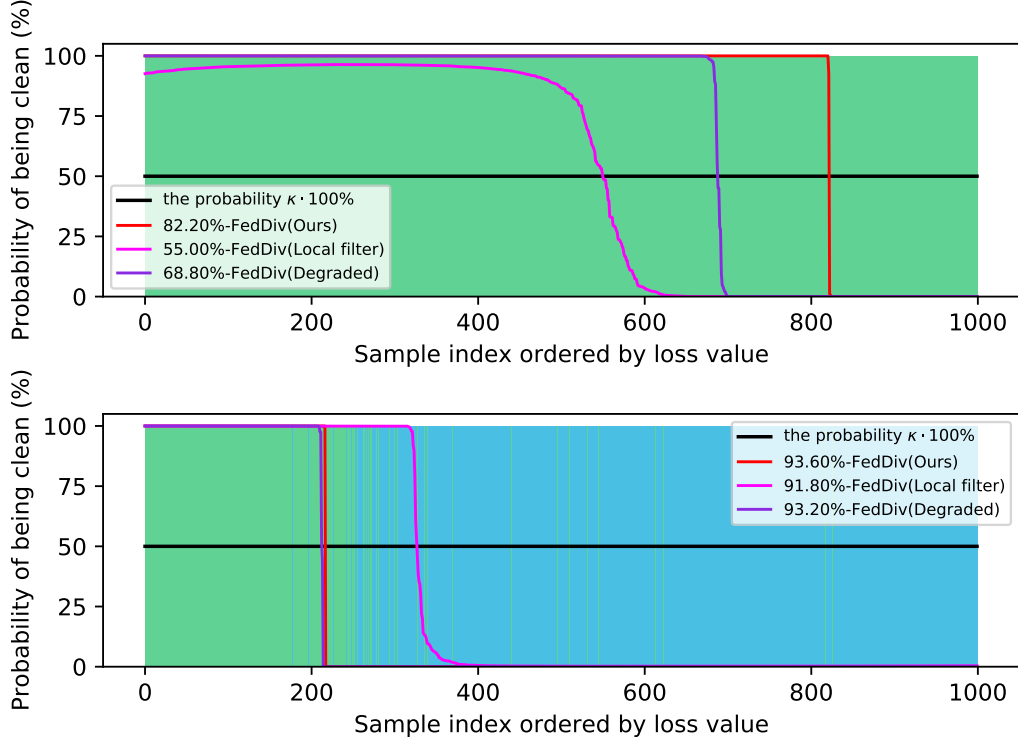
Figure 3: Two examples to illustrate the performance of three noise filters to identify label noise on a clean client (with the lowest accuracy of label noise filtering) and a noisy client. We show the probability distributions of samples being clean predicted by each filter, with the samples ranked 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.**)
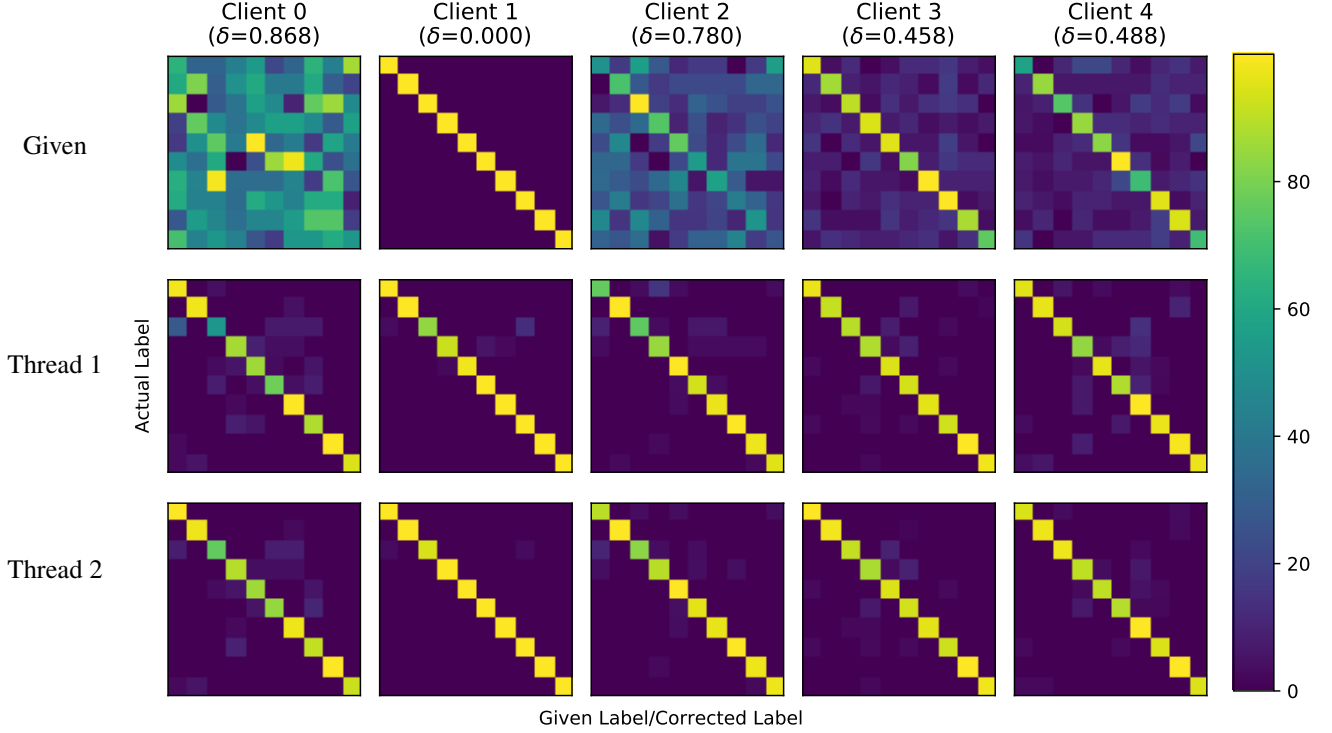
Figure 4: An evaluation of label noise filtering, noisy sample relabeling and labeled sample re-selection on five representative clients in the proposed `FedDiv`. As indicated by the heat maps, three confusion matrices for each client are associated to the actual labels v.s. the given labels before processing, 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 $(\rho, \tau) = (0.8, 0.5)$.
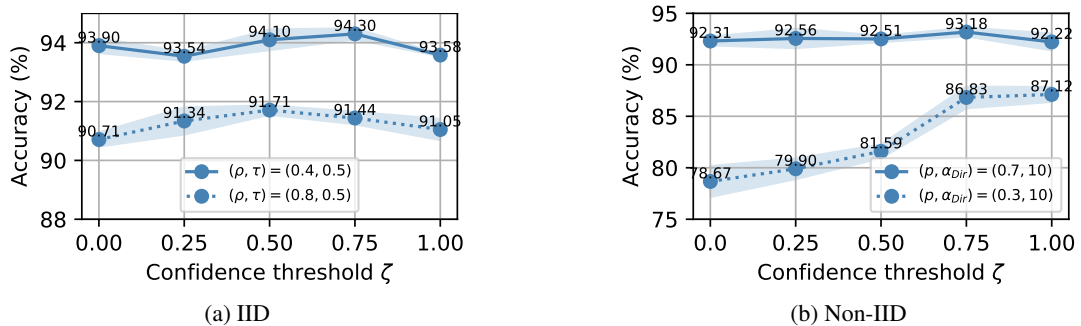

(a) IID

(b) Non-IID

Figure 5: 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 $(\rho, \tau) = (0.6, 0.5)$.