Overcoming Noisy Labels in Federated Learning Through Local Self-Guiding

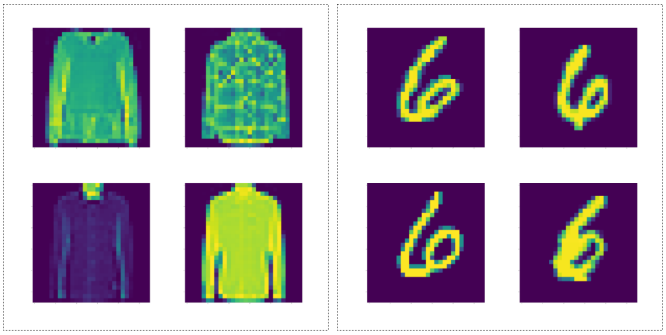
Daokuan Bai*†, Shanshan Wang*†, Wenyue Wang*†, Hua Wang§,
Chuan Zhao *†‡, Peng Yuan*†, Zhenxiang Chen *†¶
*Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan, China
†School of Information Science and Engineering, University of Jinan, Jinan, China
‡Quan Cheng Laboratory, Jinan, China
§Victoria University, Melbourne, Australia
¶Corresponding author, Email: czx.ujn@gmail.com

***Abstract*—Federated Learning (FL) is a privacy-preserving machine learning paradigm that enables clients such as Internet of Things (IoT) devices, and smartphones, to train a high-performance global model jointly. However, in real-world FL deployments, carefully human-annotated labels are expensive and time-consuming. So the presence of incorrect labels (noisy labels) in the local training data of the clients is inevitable, which will cause the performance degradation of the global model. To tackle this problem, we propose a simple but effective method Local Self-Guiding (LSG) to let clients guide themselves during training in the presence of noisy labels. Specifically, LSG keeps the model from memorizing noisy labels by enhancing the confidence of model predictions. Meanwhile, it utilizes knowledge from local historical models that haven't fit noisy patterns to extract potential ground truth labels of samples. To keep the knowledge without storing models, LSG records the exponential moving average (EMA) of model output logits at different local training epochs as self-ensemble logits on clients' devices, which will lead to negligible computation and storage overhead. Then logit-based knowledge distillation is conducted to guide the local training. Experiments on MNIST, Fashion-MNIST, CIFAR-10, ImageNet-100 with multiple noise levels, and an unbalanced noisy dataset, Clothing1M, demonstrate the resistance of LSG to noisy labels. The code of LSG is available at https://github.com/dnvbai98/LSG-Main**

***Index Terms*—Federated learning, data with noisy labels, robustness**

I. Introduction

With the rapid development of cloud computing [1], edge computing, and other technologies [2], a huge amount of end-points (smartphones, IoT devices) are put into use, and massive data are generated by them. In order to utilize knowledge in those data for better services without violating users' privacy, an attractive privacy-preserving distributed machine learning paradigm Federated Learning (FL) has received widespread attention from academia and industry. In FL [3], clients use local data to jointly train a high-performance global model in a decentralized manner, ensuring that the data never leave the devices. FL setup consists of one server and multiple clients. In each communication round, the server randomly selects a subset of clients and sends the global model to them. Clients train the model with the local data, and then send the updates to the server, which will average the updates into the new global model. However, obtaining a high-performance global model depends on high-quality data from the clients. Unfortunately, such a requirement is quite extravagant in the real world. Data used for training are very likely to be assigned incorrect labels (noisy labels) since carefully human-annotated labels are expensive and time-consuming to obtain in practice [4].



(a) True class: **Shirt** → Label: **Coat** (b) True class: **Six** → Label: **Four**

Fig 1. Examples for noisy labels in FL. (a) Images of "shirt" from the Fashion-MNIST dataset are labeled as "coat" mistakenly by clients. (b) Images of "6" from the MNIST dataset are labeled as "4" mistakenly by clients.

In FL, labels are usually independently generated by clients using various label generation methods, such as manual annotated [5] and machine-generated labels [6], so the correctness of them in clients' local data is hard to guarantee. As we can see in Fig.1, images of digit "6" can possibly be mislabeled as "4" by clients. Noisy labels will lead to degradation of model performance as deep neural network (DNN) has a huge amount of parameters to fit all training data [7]. Research [8] found that the DNN will learn simple and generalized patterns before fitting to the noisy ones in the presence of noisy labels. So the model performance will rise in the early stage of training and then start to decline, which is also called the memorization effect of DNN [8] [4]. At present, a lot of works have been proposed to alleviate the performance degradation caused by noisy labels in centralized learning (CL). However, due to the limited resources of clients' devices, most of them can not be directly applied in FL. For instance, [9] [10] [11]

require complex and computation-heavy procedures to filter out noisy labels, which are difficult to execute on resource-constrained devices. In FL, most existing methods [12] [13] [14] require a perfectly labeled auxiliary dataset to estimate the noise level of clients or select samples that are more likely to be correctly labeled for local training. There are two concerns with implementing these methods in FL. Firstly, maintaining such a perfectly labeled and task-relevant auxiliary dataset at the server is impractical in real-world FL. Secondly, simply removing samples with noisy labels will discard important information about the data distribution [15]. Recently, [16] proposes a multi-stage label correction framework to make full use of clients' local data including ones with noisy labels. However, this work highly depends on a fraction of completely clean clients and lacks flexibility, which is difficult to implement in dynamically changing scenarios of real-world FL.

Therefore, it is very important to propose a method that does not require the strong assumption mentioned above about servers and clients in FL to deal with the issue of noisy labels. Also, the valuable knowledge of clients' local data, including samples with noisy labels, should be mined as much as possible. In this work, we move our sight to clients' local training process. We aim to utilize the knowledge from the clients' local training history to obtain additional supervision. The intuition behind this is as follows: 1) DNNs will first learn generalized and simple patterns and develop a basic capability in the early stage of training. 2) Before fitting noisy labels, the model predictions are more likely to be the ground truth than the corresponding noisy label.

Based on the above intuition, we propose a method called Local Self-Guiding (LSG) to alleviate the degradation of global model performance caused by noisy labels. In LSG, for each sample in the client's local dataset, the exponential moving average (EMA) of its output logits is stored within the clients' devices during the whole training history in FL as local self-ensemble logits. Such an idea of the ensemble is also widely used in semi-supervised learning to get more stable supervision signals [17] [18].

Self-ensemble logits can be utilized to conduct logits-based knowledge distillation from models at historical epochs as guidance for current training while leading to negligible storage and computation costs. Specifically, we first enforce the model to make high-confidence predictions to keep it from memorizing noisy labels. And then we use sample-level self-ensemble logits to distill knowledge from the local model training history to extract the potential ground truth of samples with noisy labels. In summary, our contributions are mainly as follows:

- We propose a new method, named LSG against noisy labels in FL. LSG is applied in client-side local training processes, thus doesn't rely on perfectly labeled auxiliary datasets or clean clients.
- LSG keeps the model from memorizing noisy labels by enhancing the confidence of model predictions, and distills knowledge from local self-ensemble logits to extract the potential ground truth labels of samples. Therefore LSG makes full use of clients' local datasets including those with noisy labels.
- Extensive experiments on four benchmark datasets with multiple noise levels, and on an unbalanced noisy dataset, Clothing1M, compared with the state-of-the-art methods are conducted to demonstrate the effectiveness of our method.

The rest of the paper is organized as follows. The related work is presented in Section II. The preliminaries of our method is described in Section III. The proposed method Local Self-Guiding (LSG) is described in Section IV The experimental setup and results are described in Section V. Section VI concludes this study.

II. RELATED WORK

In this section, we first give an overview of previous works focusing on tackling noisy labels in CL and FL respectively. Then, we introduce the fundamental research in knowledge distillation, and focus on the important role of the multi-teacher model in maintaining the stability of knowledge distillation.

A. Learning With Noisy Label

The machine learning models highly depend on the quality of training data to get a good performance [19] [20] [21]. However, the presence of noisy labels in training data is inevitable due to annotators' skill, bias, and hardware reliability [22] [23]. DNNs are very vulnerable to noisy labels because of the huge scale of parameters. They can overfit any data with noisy labels and then face a degradation in performance. A lot of researches focus on alleviating the performance degradation caused by the noisy labels in CL. They can be roughly divided in two categories [24]: remove samples that are more likely to noisy labeled [25] [26] [27] [28] and obtaining supervision from the model outputs [29] [9] [30] [31] [10]. For the first category of methods, maintaining two networks to sample potential correctly labeled instances is a general strategy. Decouple [27] trains two deep networks simultaneously, each network selects samples to update their parameters based on a disagreement between the two networks. Co-teaching [25] also maintains two deep networks, and each of them selects a certain fraction of data with a small loss for another network to train. JoCoR [28] maintains two networks and computes the co-regularized joint loss of each training example. The instances with small losses will be selected to train both networks. The second category of methods obtains supervision from the model output information. Wang et al. [29] designs a new loss function called symmetric cross entropy (SCE). The output information can be used against noisy labels in SCE. Joint Optimization [30] proposed a joint optimization framework to correct labels during training by alternately updating network parameters and labels. SEAL [31] averages the softmax output for each sample throughout the training process and retrains the network using the averaged soft labels. Robust Curriculum Learning [10] first selects clean data dynamically based on the loss and output of historical steps, and then gradually starts learning from noisy labeled data with

pseudo-labels produced by the model. Recently, MORPH [4] proposed a self-transition learning method that automatically selects clean samples to train the network and precisely keeps expanding the training set.

Although these works are successful in a centralized fashion, many of them can not be directly applied in FL. [10] [9] [31] [30] conduct a complex and time-consuming procedure to tackle noisy labels, which can not be applied on resource-constrained devices. MORPH [4] requires the whole dataset to collect a seed of clean samples in the early stage of training. However, only a subset of clients will be selected in each round in FL.

In FL, some works have been proposed to mitigate the performance degradation caused by noisy labels in the local datasets. Most existing works [12] [13] [32] propose to utilize a correctly labeled auxiliary dataset to measure the noisy level of clients. Tuor et al. [13] proposed using a benchmark model trained on an auxiliary dataset as a sample selector. Each client then only conducts training on samples selected by the model. Focus [12] maintains a correctly labeled auxiliary dataset in FL server to measure the quality of the client's local data.

Yang et al. [32] proposed measuring the noise ratio of each client based on a clean validation dataset for client selection. The above works require the server to have an auxiliary dataset with perfectly correct labels. However, such a perfectly labeled auxiliary dataset is impractical to obtain in FL because of the annotators' skill, bias, and hardware reliability. And simply discarding noisy samples or reducing the contribution of noisy clients may lose important information about the data distribution. There are also some methods [33] [16] [24] do not require auxiliary datasets. Robust federated learning [33] proposed to exchange class-level centroids to maintain consistent decision boundaries between clients and server which do not rely on the auxiliary dataset. However, the exchanging of class-level centroids may lead to privacy leakage because centroids can be used to infer the information of the raw data. FedCorr [16] proposed a multi-stage label correction framework to handle noisy labels in FL.

However, the framework depends on perfectly clean clients, which is unrealistic in FL just like a perfectly labeled auxiliary dataset. Also, because of the dynamics of FL [34], the connection of a certain client device is not fixed and ideal. LSR [24] proposed using data augmentation to generate data pairs and minimize the discrepancy of the model output of the pairs to regularize the local training. However, utilizing supervision from a single network will inevitably introduce new noise to training.

B. Knowledge Distillation

Knowledge distillation [35] is a method that aims to compress the knowledge learned by the trained larger model (teacher) into the smaller model (student) through knowledge transfer, and is not affected by the model structure. At present, the simplest distillation strategy is distillation based on output logits. Logits are the potential information contained in the final prediction output of the neural network for the samples. To get more stable guidance from knowledge distillation, methods [36] [37] [38] have been proposed to utilize multiple teacher networks or teacher networks in different training epochs. Yang et al. [36] proposed using the label knowledge generated by the teacher model in each training epoch to guide the student model to keep stable. Wu et al. [37] proposed peer cooperative distillation by training multiple networks and transferring the logit knowledge of better teachers to peers. Such a strategy increases the stability of the model and the quality of distillation. You et al. [38] proposed a multi-teacher distillation framework, which averaged the logits of multiple teacher models and provided them to the student model for learning. The methods mentioned above have proven that multiple-teacher models will provide more stable and higher-quality guidance for student model. However, maintaining multiple networks in FL brings huge communication and storage costs. So we collect the model output logits of different training epochs instead.

III. Preliminaries

A. Federated Leaning

FL is a machine learning paradigm where multiple participants jointly train a global model. In FL, the server brings the model to data, instead of bringing data to the model, which is different from the CL [34]. Participants' local data will never leave their devices, so even privacy-sensitive data can be used for training. Such data is generated by real users' behavior, which is intrinsically different from publicly available datasets [39].

The FL setup consists of K participants and one server. For each participant, there is a local dataset with n_k samples denoted as $\mathcal{D}_k = \{(x_k^i, y_k^i)\}_{i=1}^{n_k}$. And $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$ which contains $\mathcal{C}$ classes. Here, let $\mathcal{S}$ denote the set of all K clients, and let w^t denote the weights of the global model at communication round t. At each round t, the server broadcasts the global model parameters w^t to random selected subset of client $\mathcal{S}_t$, where $\mathcal{S}_t \in \mathcal{S}$. The selected clients initialize their local model weights w_k^t using the received w^t and conduct training on their own data shard $\mathcal{D}_k$ for E epochs to reduce the Cross Entropy (CE) loss:

$$L_{CE} = CE(f(x; w_k^t), y), \tag{1}$$

where $f(x; w_t^k)$ is the output logits of local model. For instance x_i, the CE loss is calculated as:

$$\mathcal{L} = -\sum\nolimits_{c \in \mathcal{C}} y_{c,i} log(p_{c,i}), \tag{2}$$

where $y_{c,i} = 1$ if $c = c_i$ and 0 otherwise, $p_{c,i}$ is the confidence of x_i to be classified into class c. When each client finishes local training, it computes the update $w_k^t - w^t$ and sends it back to the server. The server then averages the updates to aggregate a new global model w^{t+1} and starts a new communication round.

B. Memorization Effect of DNN

Research [8] reveals the memorization effect of DNN. DNNs tend to first learn simple and generalized patterns and then gradually memorize all the patterns including irregular

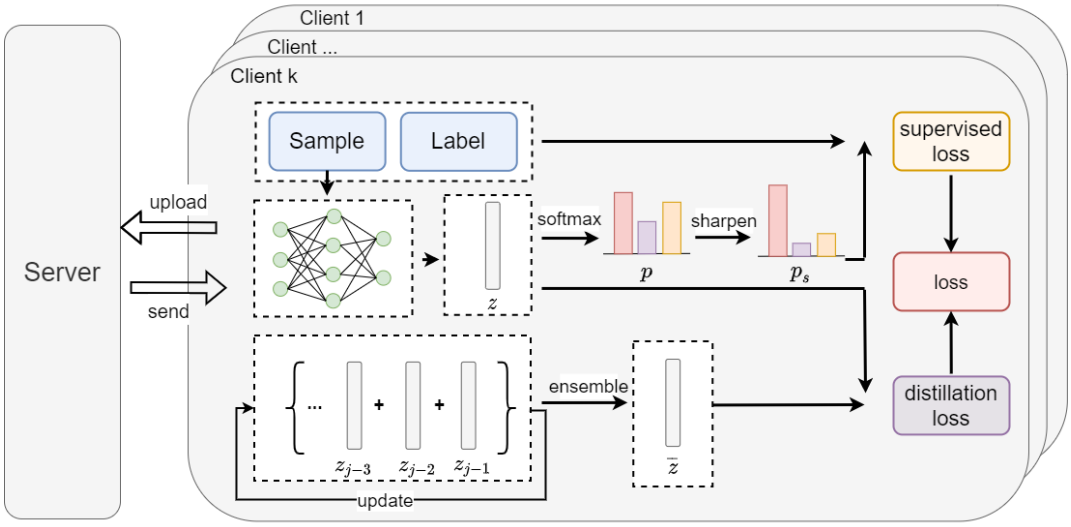


Fig 2. An overview of our LSG method. We enhance the confidence of local model predictions by sharpening them to keep the local model from memorizing noisy labels. Meanwhile, we distill knowledge from self-ensemble logits $\bar{z}$ to extract the potential ground truth labels. Note, self-ensemble logits are the EMA of local model output logits of past training epochs since FL starts.

ones such as outliers and noisy samples [4]. So in the presence of noisy labels, DNN will first learn simple and clean knowledge and develop basic discrimination ability and then gradually memorize noisy labels. The specific phenomenon is that model performance increases in the early period of training, and then starts to decrease due to the fitting of noisy labels.

In FL, such an effect also exists [24]. The global model serves as a link to learn the correct knowledge of each client's local datasets in relatively early communication rounds and then obtains a basic classification capacity. The memorization effect of DNN is an important property that our method relies on. Since the memorizing of noisy labels gradually happens after the learning of correct labels, global and local models in the historical training periods can be used to guide the subsequent local training.

C. Knowledge Distillation

Knowledge distillation [35] is a method of utilizing the knowledge of one network to guide the training of another network. The two networks can be homogeneous or heterogeneous. Generally speaking, knowledge distillation involves two roles, the teacher model and the student model, where the teacher model is pre-trained. When training the student model, the prediction of the teacher model is combined with the given label of the data for training. At this point, the loss function of the student model can be written as:

$$Loss = L_{CE} + \lambda D_{KL}(p||q). \tag{3}$$

Here, L_{CE} and D_{KL} refer to the CE loss and Kullback Leibler (KL) divergence respectively, p and q refer to the teacher and student model's prediction probability distribution respectively.

IV. METHOD

Our approach focuses on optimizing the local training process on the client side. We enforce the local model to make high-confidence predictions to keep it from memorizing noisy labels so as to keep the model clean. Then we collect self-ensemble logits,which are the EMA of historical model outputs, on the devices of clients. We distill knowledge from self-ensemble logits to extract the potential ground truth labels of samples that are noisy. The storage of self-ensemble logits is negligible so that it can be implemented in real-world FL. The overview of our method is shown in Fig.2.

Our method consists of three stages: initialization, local training, and model aggregation. We provide a detailed description of each stage in Algorithm 1.

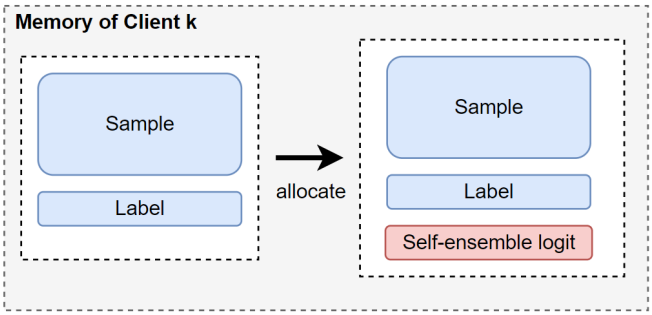


Fig 3. Initialization. Before FL starts, each client allocates corresponding memory space to initialize the self-ensemble logits, which is a $\mathcal{C}$-dimension zero-vector for each sample.

A. Initialization

Before the FL training starts, each client allocates the corresponding memory space to store self-ensemble logits

$\overline{Z}_k = \{\overline{z}_k^i\}_{i=1}^{n_k}$ on clients' device. For instance x_k^i, its corresponding initial self-ensemble logit $\overline{z}_k^i$ is a $\mathcal{C}$-dimension zero-vector, where $\mathcal{C}$ is the number of classes (see Fig.3). During the training of FL, the model outputs z_k^i will be accumulated into self-ensemble logits the model outputs $\overline{z}_k^i$ after every training epoch. Note that the self-ensemble logits are moving exponential averages of the historical model outputs, so their storage space is only positively related to the number of samples. Meanwhile, each client will initialize a scalar j_k to record the total training epochs that have been executed on clients' devices during the entire FL process, not just the local epochs in one communication round.

B. Local Training

At round t, the server sends the global model w^t to a random subset $\mathcal{S}_t$ of clients. Those clients initialize their local model w_k^t with w^t and then train on their own dataset $\mathcal{D}_k$.

Algorithm 1: Local Self-Guiding (LSG) method

Input: Global round R, Local epoch E, Numbers of clients K, Momentum α, coefficient λ.
Output: Global model w^{final}

Initialization:
for *Client 1 to K* **do**
  $j_k = 0$;
  $Z_k = [0]_{n_k \times \mathcal{C}}$;
end
For FL Server:
Initializing the global model $f_G(\cdot)$ with random parameters w^0;
for *each round 1 to R* **do**
  select k clients randomly;
  for *client 1 to k* **do**
    w_k = ClientUpdate(w^t, k);
  end
  $w^{t+1} = w^t + \sum_{k \in \mathcal{S}_t} \frac{n_k}{\sum_{k \in \mathcal{S}_t} n_k}(w_k^t - w^t)$;
end
ClientUpdate(w^t, k) :
$w_k^t \leftarrow w_t$;
for *local epoch 1 to E* **do**
  if $j_k \neq 0$ **then**
    $\overline{Z}_k = \overline{Z}_k / (1 - \alpha^{j_k})$; // *correct the startup bias*
  end
  for *each minibatch(x,y)* **do**
    $z = f(x; w_k^t)$;
    $p_s = Sharpen(softmax(z), T_s)$;
    $Loss = Loss_{sup}(p_s, y) + \lambda * loss_{dis}(\overline{z}, z, T)$;
    Update w_k^t with $Loss$;
  end
  if $j_k \neq 0$ **then**
    $\overline{Z}_k = \overline{Z}_k * (1 - \alpha^{j_k})$;
  end
  $\overline{Z}_k = \alpha \overline{Z}_k + (1 - \alpha) Z_k$;
  $j_k = j_k + 1$;
end

1) Prediction Sharpening: Considering that the given labels, which can be noisy, are the only supervision in local training, it is easy for local models to memorize incorrect labels and then accumulate noise into the self-ensemble logits. In many semi-supervised methods, a basic assumption is that the decision boundary of a classifier should not pass through high-density regions of marginal data distribution [40] [41]. Enforcing classifiers to make low-entropy (high-confidence) predictions can help to meet this assumption. Therefore, we explicitly enhance the model prediction confidence of each sample by directly sharpening the model predictions p to get sharpened prediction p_s:

$$p_s = Sharpen(p, T_s) := p^{\frac{1}{T_s}} / \sum_{c=1}^{\mathcal{C}} p_c^{\frac{1}{T_s}}, \tag{4}$$

where T_s is the temperature, $\mathcal{C}$ is the number of class. Then, p_s is used to calculate the supervised loss:

$$Loss_{sup} = CE(p_s, y), \tag{5}$$

where y is the one-hot encoding of the given labels. Sharpening operation is used in [24] to enhance the MixUp predictions for regularization. However, our purpose is to maintain the model clean to prevent accumulating errors in self-ensemble logits. The motivation of adding this operation is to enforce the model to make high-confidence predictions. Before the model starts fitting the noisy labels, the predictions will be far from it, which means argmax$(p) \neq$ argmax(y). Sharpening operation can enforce the model to make low-entropy predictions so that make the CE loss in eq.2 larger than the original one. Thus, reducing the CE loss becomes more difficult so that noisy labels will be harder to be memorized by the local model.

2) Historical Knowledge Distillation: The memorization effect of deep network [8] shows that the model will first learn generalized patterns and develop basic classification abilities. At this period, model predictions are more likely to be the ground truth. Enlightened by this effect, we regard historical models that haven't started to fit the noisy labels as teacher models and utilize them to extract the potential ground truth label. KL divergence can be seen as a measure of distance between two prediction probability distributions of the teacher model and the student model to form a distillation loss. In our method, we store the self-ensemble logits $\overline{Z}$ accumulated over past epochs on clients' devices and regard them as an ensemble of output logits obtained from different individual teacher networks. In detail, for sample x, we first compute its prediction q and self-ensemble prediction p using a softmax function with distillation temperature T:

$$p = \frac{exp(\overline{z}/T)}{\sum_c exp(\overline{z}_c/T)}, q = \frac{exp(z/T)}{\sum_c exp(z_c/T)}, \tag{6}$$

whereby z is the model outputs logits of sample x, $\sum c = \mathcal{C}$. Then we use the obtained p,q to compute a distillation loss as an additional loss term:

$$Loss_{dis} = D_{KL}(p||q), \tag{7}$$

whereby D_{KL} means KL divergence. Combined with the supervised loss, the final loss function becomes:

$$Loss = Loss_{sup} + \lambda Loss_{dis}, \tag{8}$$

where λ is the trade-off coefficient between two losses. After each training epoch, the self-ensemble logits will be updated as follows:

$$\overline{Z} = \alpha\overline{Z} + (1-\alpha)Z,\ \alpha \in [0,1), \tag{9}$$

whereby α is momentum, $Z = \{z_i\}_{i=1}^{n_k}$ is the set of model output logits for all local training samples at the current epoch. Note, before using $\overline{Z}$ to compute the distillation loss, we have to calibrate the startup bias by dividing by $(1-\alpha^j)$, where j is the total epochs the client has trained since FL starts. And it is stored as a scalar during the initialization stage. Such operation is also used in Adam [42]. When $j = 0$, we divide it by 1 instead.

We set a warm-up round t_{wu}, and the coefficient will be linearly increased from 0 to λ before reaching t_{wu}. So the given label will be considered as the prior supervision and then gradually self-ensemble logits. This also avoids the instability of model outputs in the early stage of training because the model lacks discrimination abilities.

C. Model Aggregation

For model aggregation, we follow the classical aggregation algorithm FedAvg [3]. When clients finish local training, they compute the update $w_k^t - w^t$ and send them back to the server. The server averages the updates to aggregate a new global model:

$$w^{t+1} = w^t + \sum\nolimits_{k\in\mathcal{S}_t} \frac{n_k}{n}(w_k^t - w^t), \tag{10}$$

where $n = \sum_{k\in\mathcal{S}_t} n_k$ denotes the total number of data of all selected clients.

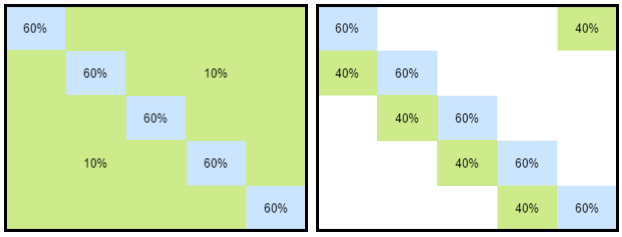
V. EXPERIMENTS

A. Experiment Setup

1) Datasets: We conduct our experiments on five datasets: MNIST [43], Fashion-MNIST [44], CIFAR-10 [45], ImageNet-100 [46], and Clothing1M [47]. The first three datasets are popular benchmark datasets and contain 10 classes. ImageNet-100 is a subset of the ImageNet [48], which contains 100 classes. The list of the one hundred categories can be found in [46]. Clothing1M is an unbalanced noisy dataset with 14 classes and contains 1 million images of clothing. It is crawled from several online shopping websites. And it contains about 39% incorrect labels [47].

Since MNIST, Fashion-MNIST, CIFAR-10, and ImageNet-100 are clean, following [24] [25], we manually flip the labels of these datasets by the noise transformation matrix to generate noisy labels. We consider two types of noise: (1) Symmetry flipping [49]; (2) Pairwise flipping [25]. Noise ratio ϵ means the proportion of noisy labels in the whole training set. For example, $\epsilon = 0.4$ means there are 40% labels of the training data incorrect. Note that under the same noise ratio ϵ, pairwise cases are much more challenging than symmetric cases [24]. As shown in Fig.4, with fixed a noise ratio of ϵ = 0.4, each class only has 20% more correctly labeled samples than noisy ones in pairwise cases while 50% in symmetric cases.

To generate noisy labels in the first four datasets, we first divide the whole datasets into $\mathcal{C}$ parts by class. Then we flip an equal number of labels in each part with a fixed random seed by a specified noise type and ratio. Finally, we mix and shuffle the $\mathcal{C}$ parts. Our experiments consider both independent identically distributed (IID) and non-IID local data distributions. For IID local data distributions, the whole dataset $\mathcal{D}$ is uniformly split at random and distributed among K clients. For non-IID local data distributions, each client only has five random classes of data, which follows the settings in [24]. Such a data distribution presents a high level of non-IID. For Clothing1M, we followed previous works [16] [24] and randomly split 1 million images into 100 partitions with the equivalent number of samples for each client. Since it is a naturally unbalanced dataset, the distribution of local data among clients is heterogeneous(Non-IID).



(a) Symmetry(ϵ = 40%) (b) Pair(ϵ = 40%)

Fig 4. Transition matrices of different noisy types (using 5 classes as an example). The diagonal blue grid represents the percentage of correctly labeled samples in each class. While green grids represent the percentage of samples labeled to other classes incorrectly

2) Implementation Details: We implement our experiment in Python using Pytorch on Nvidia GeForce RTX 3090 GPUs. For FL settings, we set 100 clients in total, at each communication round 10 clients are selected to conduct local training. We run 100 communication rounds on the first three 10-categories datasets, 400 rounds on ImageNet-100, and 40 rounds on Clothing1M. The number of local epochs is always set to 5. We use Adam optimizer for the first three 10-categories datasets and ImageNet-100. The learning rate is set to 10^{-3} and 5×10^{-4} respectively. We use SGD optimizer with a momentum of 0.9 for Clothing1M. And the learning rate is set to 10^{-2}. The batch size and weight decay are always set to 128 and 10^{-4}. We will explain the hyperparameter settings in our experiment as follows. The distillation temperature T is set to $\frac{1}{3}$ empirically. Sharpening temperature is set to $\frac{1}{2}$ following MixMatch [41]. The warm-up rounds t_{wu} (discussed in Section IV-B2) is set to 10, 20, 40, 100, and 10 for MNIST, Fashion-MNIST, CIFAR-10, ImageNet-100, and Clothing1M respectively. For faster convergence, the sharpening operation is applied after the warm-up period on ImageNet-100 and Clothing1M. The momentum α of self-ensemble logits is set to 0.9. For the first three datasets, the coefficient λ is set to 0.4 in the noisy ratio $\{0.3, 0.4, 0.5\}$ in the symmetric case and $\{0.2, 0.3\}$ in pairwise cases, and it is set to 1.0 in other extreme noise ratios. And it is set to 1.0 for ImageNet-100 and Clothing1M. If not specified, we report the average of accuracy of the last 10 communication rounds under IID settings.

Table I. Test accuracy(%) on three benchmark datasets with various noise levels.

Dataset	Method	Test Accuracy(%)								
	Noise Type	Symmetric					Pairwise			Avg.
	Noise Ratio	0.3	0.4	0.5	0.6	0.7	0.2	0.3	0.4	
MNIST	FedAvg	91.34	83.53	73.6	57.86	42.12	94.27	85.52	70.35	74.82
	Symmetric CE	99.1	98.91	98.54	97.77	95.1	99.1	98.63	94.13	97.66
	Co-teaching	98.8	98.11	97.38	95.94	93.84	98.83	97.97	94.43	96.91
	RFL	99.07	98.92	98.84	98.44	**98.4**	99.08	99.01	**98.98**	**98.84**
	FedAvg + LSR	99.05	98.88	98.71	98.39	88.89	97.89	97.72	96.98	97.06
	Ours	**99.15**	**99.98**	**98.76**	**98.51**	97.35	**99.2**	**99.06**	97.90	98.61
Fashion-MNIST	FedAvg	80.02	72.75	62.29	49.22	35.87	86.44	77.38	63.05	65.88
	Symmetric CE	88.86	85.96	80.32	60.87	49.34	89.99	84.51	68.44	76.04
	Co-teaching	89.22	88.11	86.82	84.43	81.03	90.37	87.77	83.03	86.35
	RFL	88.26	87.41	85.54	84.04	79.22	89.67	89.12	88.17	86.43
	FedAvg + LSR	90.1	89.59	88.84	87.87	84.93	90.56	90.12	88.94	88.87
	Ours	**91.31**	**90.82**	**90.04**	**88.93**	**86.19**	**91.73**	**91.20**	**89.49**	**89.96**
CIFAR-10	FedAvg	53.78	46.06	36.93	28.45	19.8	66.93	58.47	48.04	44.81
	Symmetric CE	64.8	56.4	47.45	34.11	23.97	67.56	59.48	45.91	49.96
	Co-teaching	70.23	66.84	62.54	56.25	45.28	71.44	66.41	57.21	62.03
	RFL	66.29	60.38	54.05	43.18	32.38	69.01	61.18	49.71	54.52
	FedAvg + LSR	71.1	66.92	62.6	55.56	44.12	73.17	70.18	60.04	62.96
	Ours	**74.88**	**71.43**	**67.48**	**60.7**	**48.29**	**75.33**	**72.12**	**65.38**	**66.95**

3) Baseline: We compare with five baselines to demonstrate the superiority of our method. The five baselines are FedAvg [3], Symmetric CE [29], Co-teaching [25], Robust Federated learning (RFL) [33], Local Self-Regularization (LSR) [34] respectively. Symmetric CE and Co-teaching are methods dealing with noisy labels in CL. And RFL and LSR are methods against noisy labels in FL. For both symmetric CE and Co-teaching, they are directly applied to local clients, federated average algorithm is used for local model aggregation. FedAvg means the classical FL method without any measure to deal with noisy labels using CE.

B. Experiments on 10-categories Datesets

Our experimental results on MNIST, Fashion-MNIST, and CIFAR-10 are demonstrated in Table I. For the above three datasets, we used a 9-layer CNN applied in [24] [25] [33]. For FedAvg with CE, symmetric CE, Co-teaching, and RFL, we demonstrate the results reported in [24]. The experiment results show that our method is quite robust against two kinds of noisy types with different noisy ratios in most cases. In summary, LSG achieves the best test accuracies across most noise settings on all three datasets. For MNIST, the performance of our method is better than others in most cases. When facing an extremely noisy level, the performance of RFL is mildly higher than our method. There are two reasons for the results. Firstly, for a simple task like MNIST, the centroids of classes are quite easy to find, and it can be regarded as strong supervision during training in the presence of noisy labels. Secondly, the MNIST task is relatively easy to learn, which causes the overfitting of our huge capacity network. So some model outputs with noisy patterns may be accumulated into self-ensemble logits. Note, the exchanging of the class-level centroids in RFL will lead to privacy concerns because they can be used to infer the sensitive information of the raw local data. For Fashion-MNIST and CIFAR-10, our method outperforms all other methods in all noise settings. LSG shows greater advantages on more complex tasks like the CIAFR-10. Note that the test accuracy of our method is at least 4 percentage points higher than other methods on average. It demonstrated that ensemble information produced during training history can be stable supervision in FL to tackle noisy labels.

C. Experiments on ImageNet-100

To verify LSG's scalability on large dataset with more classes, we apply our method on the ImageNet-100 benchmark to improve the model performance when facing noisy labels.

We use a ResNet-18 [50] in our experiment. Images are resized to 224×224 and normalized. The experimental results on ImageNet-100 of various noise levels are shown in Table II. It is easy to notice that the performance improvements of our method on ImageNet-100 are lower than those on 10-categories datasets. This is because ImageNet-100 is a much harder classification task, and the performance is hard to improve even if there are no noisy labels. However, our method still achieves the best test accuracies on all noise levels. For more complicated datasets that converge more slowly, most of the baselines fail to maintain the performance. From the results

of Symmetric CE and Co-teaching, we can see that methods in CL can not easily be applied in FL scenarios directly because of the smaller sizes of client datasets and the characteristic of harder-to-converge . For LSR, when faced with tasks with much more classes, which are harder to converge, utilizing data augmentation as extra supervision in training will also bring new noise to the model.

Table II. Test accuracy(%) on ImageNet-100.

Method	Sysmmetric		Pairwise
Nois Ratio	0.4	0.6	0.4
FedAvg	27.9	15.39	33.87
Symmetric CE	28.96	14.56	33.72
Co-teaching	30.7	13.73	26.8
RFL	33.55	18.04	34.13
FedAvg+LSR	27.71	11.73	28.24
Ours	**40.02**	**29.27**	**39.96**

Table III. Test accuracy(%) on non-IID Fashion-MNIST dataset with extreme noisy levels.

Method	Symmetric (ϵ=0.7)	Pairwise (ϵ=0.4)
FedAvg	32.23	57.03
Symmetric CE	45.59	66.21
Co-teaching	81.32	82.29
RFL	74.89	70.45
FedAvg+LSR	80.64	76.30
Ours	**82.56**	**85.21**

Table IV. Average (3 trails) of best test accuracy on Clothing1M. 1, 2 is quoted from [29], 3 and 4 are quoted from [51] and [52], 5 - 8 are quoted from [24]. CL and FL means centralized learning and federated learning respectively.

	Method	Settings	Test Accuracy(%)
1	Cross Entropy	CL	68.80
2	Symmetric CE	CL	71.02
3	Forward	CL	69.84
4	Generalized CE	CL	69.75
5	FedAvg	FL	68.56
6	Symmetric CE	FL	69.63
7	FedAvg+LSR	FL	69.30
8	RFL	FL	70.32
9	Ours	FL	**70.64**

D. Experiments on Non-IID Data Distribution

We conduct experiments of non-IID setting on Fashion-MNIST with an extremely noisy ratio. Each client only has five random classes. In non-IID settings, we quoted other results from [24]. The results are shown in Table III. Although our method suffers from a drop in performance compared to that under the IID settings, LSG achieves the best test accuracies among two types of extreme noise compared to other methods. This shows that the performance of the global model will affect the stability of guidance brought by self-ensemble logits. Because the global model of each round plays an important role in accumulating correct knowledge into self-ensemble logits for clients.

Table V. Test accuracy(%) for the effect of prediction sharpening.

Noisy Type	Noisy Ratio	w/o Sharpening	Ours
Symmitric	0.3	84.2	91.31
	0.4	78.81	90.82
	0.5	69.91	90.04
	0.6	59.16	88.93
	0.7	42.19	86.19
Pairwise	0.2	86.44	91.73
	0.3	77.8	91.2
	0.4	63.51	89.49

Table VI. Test accuracy(%) for various distillation terms.

Noisy Type	Noisy Ratio	Distillation Loss Term		
		w/o	KL Div	JS Div
Symmetric	0.3	91.15	91.31	**91.33**
	0.4	90.63	**90.82**	90.74
	0.5	89.83	**90.04**	90.02
	0.6	87.94	**88.93**	88.52
	0.7	81.3	**86.19**	82.71
Pairwise	0.2	91.61	**91.73**	91.68
	0.3	90.86	**91.20**	90.97
	0.4	86.15	**89.49**	87.05

E. Experiments on Unbalanced Dataset

To verify LSG for unbalanced training, experiments are conducted on Clothing1M. Clothing1M is a large-scale real-world noisy dataset containing one million images. It is a naturally unbalanced dataset, and the number of images of each class varies greatly. Similar to [16] [24] [29], we conduct the experiment using a ResNet-50 [50] pre-trained on ImageNet. The images are resized to 224×224 and normalized. We adopt random horizontal flipping during training. The experimental results are demonstrated in Table IV. Experiment with Co-teaching is not conducted since its sensitive hyperparameter $R(T)$(noise ratio of client) is hard to estimate in practice. We also compared with methods(Forward [52], Generalized CE [51]) tackling noisy labels in CL setting. The experiment results show that LSG can effectively improve the performance of Clothing1M in FL setting.

F. Ablation Study

There are two main components in LSG. We use a sharpening operation to enhance the model prediction confidence to

keep the model from memorizing noisy labels. Such operation will ensure clean knowledge is accumulated in self-ensemble logits. And then the knowledge is distilled from self-ensemble logits to extract the potential ground truth for each sample. The ablation studies are conducted on Fashion-MNIST datasets with IID data settings.

1) Sharpening Predictions: The ablation study about sharpening operation is displayed in Table V. We can see that without sharpening, there is a decline in the resistance to noisy labels of our methods, especially with extremely noisy ratios. The reason is that self-ensemble logits rely on clean model knowledge to conduct further guidance. While the model gradually fits the given noisy labels, the error is also accumulated in self-ensemble logits, which will lead to wrong direction of guidance. The experimental results prove that sharpening operation is an indispensable part because it ensures correct knowledge is accumulated in self-ensemble logits for further guidance in local training.

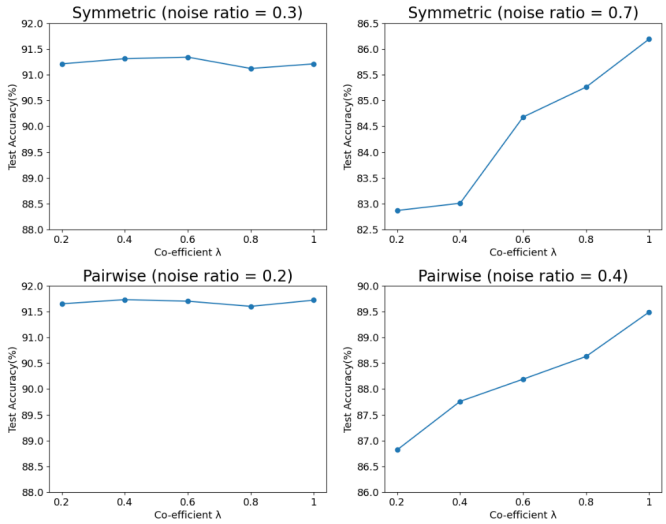


Fig 5. The performance dependency of coefficient λ in low and extreme noisy levels.

2) Distillation Loss Term: As shown in Table VI, we evaluate the effectiveness of the distillation loss term by removing and replacing it. The alternative distillation loss term is Jensen Shannon (JS) divergence used in [24], which is a kind of symmetric KL divergence to minimize the discrepancy of model output logits and self-ensemble logits. In Table VI, w/o means that we remove the distillation loss, KL Div means the KL divergence is used, and JS Div means the JS divergence is used. The results show the KL Div distillation loss term can effectively improve test accuracy in extreme noise levels, which proves the effectiveness of guidance from training history. At the same time, we found that when the noise level is relatively low, the performance improvement brought by the distillation loss is not high. We achieve 92.54% accuracy on Fashion-MNIST with no noisy labels. This illustrates that some hard mislabeled samples cannot be sufficiently learned by the supervision of the model's own knowledge. Also, we notice that JS divergence can not maintain the performance in extreme noise levels because it weakens the supervision of self-ensemble logits. We further conduct experiments about the influence of the coefficient λ. As we can see in Fig.5, when the noise level is relatively low, the performance of the global model is not very sensitive to λ. But in extreme noise levels, performance grows with coefficient because larger λ enhances the role of distillation loss term and brings stronger supervision.

VI. Conclusion

In this paper, we focus on the issue of noisy labels in FL and propose a simple but effective method LSG to deal with it. LSG keeps the model from memorizing the noisy labels and utilizes the self-ensemble logits to extract the potential ground truth. Self-ensemble logits can be used to conduct logits-based knowledge distillation from historical models as guidance for current training while leading to negligible storage and computation costs. LSG does not rely on perfectly labeled auxiliary datasets or completely clean clients, which can better meet the needs of realistic FL scenarios. Extensive experiments demonstrate the robustness of LSG method in various noise levels and types. However, LSG utilizes the knowledge of the model itself to guide subsequent training. Therefore, for some hard mislabeled samples, it is difficult to obtain reliable supervision from the model itself. Optimizing the learning of hard mislabeled samples in FL is the future direction of our work.

Acknowledgment

This work was supported by the Shandong Provincial Key RD Program of China under Grants No.2021SFGC0401, Project of Shandong Province Higher Educational Youth Innovation Science and Technology Program under Grant No.2019KJN028, Natural Science Foundation of Shandong Province under Grant No. ZR2019LZH015, Taishan Scholars Program, Shandong Provincial Natural Science Foundation under Grant No. ZR2021LZH007.

References

[1] M. E. Kabir, A. N. Mahmood, H. Wang, and A. K. Mustafa, "Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 408–417, 2015.

[2] J.-Y. Li, K.-J. Du, Z.-H. Zhan, H. Wang, and J. Zhang, "Distributed differential evolution with adaptive resource allocation," *IEEE transactions on cybernetics*, 2022.

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

[4] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Robust learning by self-transition for handling noisy labels," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1490–1500.

[5] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[7] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[8] D. Arpit, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio *et al.*, "A closer look at memorization in deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 233–242.

[9] T. Nguyen, C. Mummadi, T. Ngo, L. Beggel, and T. Brox, "Self: learning to filter noisy labels with self-ensembling," in *International Conference on Learning Representations (ICLR)*, 2020.

[10] T. Zhou, S. Wang, and J. Bilmes, "Robust curriculum learning: from clean label detection to noisy label self-correction," in *International Conference on Learning Representations*, 2020.

[11] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," *arXiv preprint arXiv:2002.07394*, 2020.

[12] Y. Chen, X. Yang, X. Qin, H. Yu, B. Chen, and Z. Shen, "Focus: Dealing with label quality disparity in federated learning," *arXiv preprint arXiv:2001.11359*, 2020.

[13] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 5020–5027.

[14] L. Nagalapatti and R. Narayanam, "Game of gradients: Mitigating irrelevant clients in federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9046–9054.

[15] E. Arazo, D. Ortego, P. Albert, N. O'Connor, and K. McGuinness, "Unsupervised label noise modeling and loss correction," in *International conference on machine learning*. PMLR, 2019, pp. 312–321.

[16] J. Xu, Z. Chen, T. Q. Quek, and K. F. E. Chong, "Fedcorr: Multi-stage federated learning for label noise correction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 184–10 193.

[17] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[18] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *arXiv preprint arXiv:1610.02242*, 2016.

[19] J. Yin, M. Tang, J. Cao, M. You, H. Wang, and M. Alazab, "Knowledge-driven cybersecurity intelligence: Software vulnerability co-exploitation behaviour discovery," *IEEE Transactions on Industrial Informatics*, pp. 1–9, 2022.

[20] G. Algan and I. Ulusoy, "Image classification with deep learning in the presence of noisy labels: A survey," *Knowledge-Based Systems*, vol. 215, p. 106771, 2021.

[21] N. Nigam, T. Dutta, and H. P. Gupta, "Impact of noisy labels in learning techniques: A survey," in *Advances in Data and Information Sciences: Proceedings of ICDIS 2019*. Springer, 2020, pp. 403–411.

[22] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, 2017, pp. 1723–1732.

[23] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," *Medical image analysis*, vol. 65, p. 101759, 2020.

[24] X. Jiang, S. Sun, Y. Wang, and M. Liu, "Towards federated learning against noisy labels via local self-regularization," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 862–873.

[25] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[26] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *International conference on machine learning*. PMLR, 2018, pp. 2304–2313.

[27] E. Malach and S. Shalev-Shwartz, "Decoupling" when to update" from" how to update"," *Advances in neural information processing systems*, vol. 30, 2017.

[28] H. Wei, L. Feng, X. Chen, and B. An, "Combating noisy labels by agreement: A joint training method with co-regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 726–13 735.

[29] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 322–330.

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

[31] P. Chen, J. Ye, G. Chen, J. Zhao, and P.-A. Heng, "Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 442–11 450.

[32] M. Yang, H. Qian, X. Wang, Y. Zhou, and H. Zhu, "Client selection for federated learning with label noise," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2193–2197, 2021.

[33] S. Yang, H. Park, J. Byun, and C. Kim, "Robust federated learning with noisy labels," *IEEE Intelligent Systems*, vol. 37, no. 2, pp. 35–43, 2022.

[34] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[35] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[36] C. Yang, L. Xie, C. Su, and A. L. Yuille, "Snapshot distillation: Teacher-student optimization in one generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2859–2868.

[37] G. Wu and S. Gong, "Peer collaborative learning for online knowledge distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 302–10 310.

[38] S. You, C. Xu, C. Xu, and D. Tao, "Learning from multiple teacher networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1285–1294.

[39] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[40] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[41] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in neural information processing systems*, vol. 32, 2019.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[45] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[46] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 2020, pp. 776–794.

[47] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang, "Learning from massive noisy labeled data for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2691–2699.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[49] B. Van Rooyen, A. Menon, and R. C. Williamson, "Learning with symmetric label noise: The importance of being unhinged," *Advances in neural information processing systems*, vol. 28, 2015.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[51] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in neural information processing systems*, vol. 31, 2018.

[52] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1944–1952.