Recovering Labels from Local Updates in Federated Learning

Huancheng Chen [1] **Haris Vikalo** [1]

Abstract

Gradient inversion (GI) attacks present a threat to the privacy of clients in federated learning (FL) by aiming to enable reconstruction of the clients' data from communicated model updates. A number of such techniques attempts to accelerate data recovery by first reconstructing labels of the samples used in local training. However, existing label extraction methods make strong assumptions that typically do not hold in realistic FL settings. In this paper we present a novel label recovery scheme, *Recovering Labels from Local Updates* (RLU), which provides near-perfect accuracy when attacking untrained (most vulnerable) models. More significantly, RLU achieves high performance even in realistic real-world settings where the clients in an FL system run multiple local epochs, train on heterogeneous data, and deploy various optimizers to minimize different objective functions. Specifically, RLU estimates labels by solving a least-square problem that emerges from the analysis of the correlation between labels of the data points used in a training round and the resulting update of the output layer. The experimental results on several datasets, architectures, and data heterogeneity scenarios demonstrate that the proposed method consistently outperforms existing baselines, and helps improve quality of the reconstructed images in GI attacks in terms of both PSNR and LPIPS.

1. Introduction

Federated learning (FL) (McMahan et al., 2017), which aims to enable collaborative training of ML models while protecting privacy of the participating clients, has attracted considerable interest in privacy-sensitive fields such as healthcare and finance (Yang et al., 2019). However, recent works have demonstrated vulnerability of FL systems to privacy attacks such as membership inference (Shokri et al., 2017), property inference (Melis et al., 2019) and gradient inversion (Zhu et al., 2019). In particular, gradient inversion methods have been shown capable of reconstructing private training data given the gradients computed during local training. The milestone work, Deep Leakage from Gradient (DLG) (Zhu et al., 2019), minimizes the difference between simulated and true gradients to extract data and the corresponding labels belonging to FL clients training a model for a classification task. However, DLG does not perform well in settings where the training batch size is large and the data resolution is high, since in such scenarios the joint optimization of data and the corresponding labels becomes challenging. A subsequent study, iDLG (Zhao et al., 2020), presents an analytical method for the recovery of ground-truth labels from the gradients by exploiting a relationship between the labels and the signs of the gradients. GradInversion (Yin et al., 2021) takes a step further by exploiting a relationship between labels and the magnitudes of gradient in the output layer to perfectly restore labels of the samples used in the considered training round, and facilitates high-resolution reconstruction of data in a batch with as many as 48 samples. Unfortunately, this method assumes no repeated labels in the batch, which is unrealistic in real-world settings. Recently, iRLG (Ma et al., 2023) attempted to address the limitations of GradInversion, enabling recovery of potentially repeated labels in a batch by utilizing gradients computed using a randomly initialized untrained model. However, the performance of iRLG drastically deteriorates as the model's accuracy improves.

The above label recovery methods either: (1) assume no repeated labels in a batch; (2) use non-negative network activation functions; or (3) perform well only on untrained models. They further assume that each client in an FL system runs only a single epoch when updating its local model, while the more practical settings with multiple epochs of local training are not considered. Moreover, these studies are limited to the scenarios where a server collects gradients and do not apply to the standard FL setting where the server collects model updates rather than the gradients.

Aiming to address the aforementioned limitations of the prior work, in this paper we propose a novel method for label recovery, Recovering Labels from Local Updates (RLU).

[1]University of Texas at Austin, Texas, USA. Correspondence to: Huancheng Chen <huanchengch@utexas.edu>, Haris Vikalo <hvikalo@ece.utexas.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

We start by analyzing correlations between local updates of the output layer and the ground-truth labels of data samples in training batches for several frequently used FL algorithms. These correlations, along with the expected value of the "erroneous confidence" (i.e., the level of confidence when making erroneous decisions) evaluated on a small auxiliary dataset, are then used to recover the labels. As local training unfolds across multiple epochs, distribution of the previously mentioned "erroneous confidence" undergoes changes which the server cannot access. Simulating an underlying dynamical model helps estimate how this distribution evolves across training epochs; the obtained estimates of intermediate distributions are then utilized to help achieve notable enhancement in the accuracy of label recovery. The effectiveness of RLU is demonstrated in extensive experiments involving several FL algorithms and various model architectures on SVHN, CIFAR10, CIFAR100 and Tiny-ImageNet datasets. In all the considered settings, RLU outperforms state-of-the-art baselines on both untrained and well-trained models. The main contributions of the paper are summarized as follows:

- We propose a general analytical method that enables the server in an FL system to recover labels of the points used by clients in local training; the proposed method applies to various FL algorithms and makes no assumptions regarding the activation function or batch label composition.
- To allow accurate label recovery in FL systems where clients train across multiple epochs, we simulate the evolution of the model through the epochs via a Monte Carlo method that updates the "erroneous confidence" characterizing the correlation between local updates and training data labels.
- To evaluate the proposed label recovery method, we conduct comprehensive benchmarking experiments across a number of FL settings where we vary the level of data heterogeneity, model architectures and local objective functions.

2. Preliminary and Related Work

Federated Learning. The classical FL algorithm, FedAvg (McMahan et al., 2017), enables K clients to collaboratively train global model without sharing their private data $\mathcal{D}_k$. In FedAvg, the server initializes training round t by broadcasting global model $\boldsymbol{\theta}^{(t)}$ to the clients. Client k updates its local model by running a gradient descent procedure on its (private) local data; the server then collects updated local models $\boldsymbol{\theta}_k^{(t)}$ from the clients and averages them to form a new global model,

$$\boldsymbol{\theta}^{(t+1)} = \sum_{k=1}^{K} p_k \boldsymbol{\theta}_k^{(t)}, \tag{1}$$

where p_k denotes the weight assigned to client k. Since the training is initialized by broadcasting global model $\boldsymbol{\theta}^{(t)}$ to the clients, local updates are computed as $\Delta\boldsymbol{\theta}_k^{(t)} = \boldsymbol{\theta}_k^{(t)} - \boldsymbol{\theta}^{(t)}$. It is worth pointing out that in FL algorithms other than FedAvg, $\Delta\boldsymbol{\theta}_k^{(t)}$ is not necessarily proportional to the gradients $\nabla\boldsymbol{\theta}_k^{(t)}$. We discuss properties of local updates in Section 3.3.

Gradients Inversion Attack. DLG (Zhu et al., 2019) is the first optimization-based method for reconstructing training data given the gradients $\nabla\boldsymbol{\theta}$ and model $\boldsymbol{\theta}$. More specifically, DLG minimizes the difference between simulated and ground-truth gradients,

$$\mathbf{x}^\star, \mathbf{y}^\star = \arg\min_{\mathbf{x}', \mathbf{y}'} \left\| \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{task}}(\boldsymbol{\theta}, \mathbf{x}', \mathbf{y}') - \nabla\boldsymbol{\theta} \right\|^2. \tag{2}$$

Follow-up studies (Geiping et al., 2020; Yin et al., 2021) introduced total variation and group consistency regularization to the objective of the gradient inversion optimization, enabling high reconstruction performance on ImageNet (Deng et al., 2009). Recently, a number of works (Jeon et al., 2021; Fang et al., 2023; Zhang et al., 2023) leveraged pre-trained generative models to improve the gradient inversion attack and achieve state-of-the-art performance with batch size set to 32. However, these schemes assume knowing the ground-truth labels $\mathbf{y}$ in the batch – an unrealistic assumption that significantly accelerates the search for the optimal data $\mathbf{x}^\star$.

Label Recovery Attack. To improve the performance of an attack on gradients computed using samples coming from a relatively large batch, a series of studies (Zhao et al., 2020; Yin et al., 2021; Dang et al., 2021; Wainakh et al., 2021; Geng et al., 2021; Ma et al., 2023) proposed various analytical approaches to recovering labels $\mathbf{y}$ prior to solving the optimization over $\mathbf{x}'$. However, all these methods suffer limitations that restrict their practical feasibility. GradInversion (Yin et al., 2021) assumes there are no repeated labels in a batch; RLG (Dang et al., 2021) can only recover class-wise labels but not instance-wise labels; LLG (Wainakh et al., 2021) and ZLG (Geng et al., 2021) require non-negative activation functions; iRLG (Ma et al., 2023) performs well only on randomly initialized untrained models. Moreover, these methods primarily focus on FedAvg and generally provide little if any discussion of other FL algorithms.

3. Methodology

3.1. Problem Settings

We consider multi-class classification models trained in FL settings where an honest-but-curious (HBC) server aims to recover ground-truth labels of the training samples using local updates $\Delta\boldsymbol{\theta}_k^{(t)}$ collected from clients while following a standard FL training procedure. The HBC server knows the batch size, number of local epochs and learning rate used by the clients but has no information about local data

distribution. We assume each local model is trained by minimizing the cross-entropy (CE) loss

$$\mathcal{L}_{\text{ce}} = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\log\frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}, \tag{3}$$

where $\mathcal{B}$ denotes a training batch; $(\mathbf{x}^{(i)}, y^{(i)})$ is the i-th example in the batch; N is the number of classes; and $\mathbf{q}^{(i)} = \mathbf{W}\cdot\mathbf{e}^{(i)} + \mathbf{b}$ denotes the output logits of the model given the embedding $\mathbf{e}^{(i)} \in \mathbb{R}^L$ of $\mathbf{x}^{(i)}$, where $\mathbf{W} \in \mathbb{R}^{N\times L}$ and $\mathbf{b} \in \mathbb{R}^N$ are the weights and bias of the output layer.

3.2. Label Recovery from Local Updates (RLU)

Following the definition of the CE loss in Eq. 3, contribution of $\mathbf{x}^{(i)}$ to the j-th component of the gradient of $\mathbf{b}$ can be computed as (the proof provided in Appendix A.1)

$$\nabla\mathbf{b}_j^{(i)} = \begin{cases} \dfrac{\exp(\mathbf{q}_j^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})} = \mathbf{s}_j(\mathbf{x}^{(i)}), \text{ if } j \neq y^{(i)}, \\ -\dfrac{\sum_{n\neq j}\exp(\mathbf{q}_n^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}, \text{ if } j = y^{(i)}. \end{cases} \tag{4}$$

Assuming stochastic gradient descent (SGD) optimizer, the local update of the j-th component of the output layer's bias computed by client k is

$$\Delta\mathbf{b}_j = -\frac{\eta}{|\mathcal{B}^\tau|}\sum_{\tau=1}^{m}\sum_{i=1}^{|\mathcal{B}|}\nabla\mathbf{b}_j^{(i,\tau)}, \mathcal{B}^\tau \sim \mathcal{D}_k, \tag{5}$$

where τ is the local epoch index, m is the number of epochs, η is the learning rate, and $\mathcal{D}_k$ denotes client k's data. Note that $\mathbf{s}_j(\mathbf{x}^{(i)}) \in (0,1)$ can be interpreted as the "erroneous confidence" of labeling $\mathbf{x}^{(i)}$ as class j while $j \neq y^{(i)}$ (i.e., confidence in a labeling decision that is in fact erroneous). Let $\mathcal{S}_{n,j}$ denote the expected erroneous confidence for class j given a sample with true label $n \neq j$,

$$\mathcal{S}_{n,j} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_k^{(n)}}\left[\mathbf{s}_j(\mathbf{x})\right], \forall n, j \in [N] \wedge n \neq j, \tag{6}$$

where $\mathcal{D}_k^{(n)} \subseteq \mathcal{D}_k$ collects samples with label n and $(\mathbf{x}, y)$ is a random sample from $\mathcal{D}_k^{(n)}$. These expectations are indicative of the model's training error: $\mathcal{S}_{n,j} \approx \frac{1}{N}$ in a randomly initialized untrained model making random predictions, whereas $\mathcal{S}_{n,j}$ asymptotically goes to 0 as the accuracy of the model increases.

Note that the expected erroneous confidence, $\mathcal{S}_{n,j}$ does not admit closed-form expression; to analyze it and gain needed insight, we make the following assumption.

Assumption 3.1. The output logits $\mathbf{q}^{(n)}$ of model $\boldsymbol{\theta}$ when the input is $\mathbf{x} \sim \mathcal{D}^{(n)}$ follows a multivariate normal distribution, i.e.,

$$\mathbf{q}^{(n)} \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \tag{7}$$

where the mean $\boldsymbol{\mu}_n$ and covariance $\boldsymbol{\Sigma}_n$ depend on the accuracy of the model.

For an untrained deep model whose parameters are initialized from a zero-mean uniform distribution, expected values of the output logits $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_N$ are approximately $\mathbf{0}$. As the accuracy of the model improves during the training process, $\boldsymbol{\mu}_{j,j}$ converges to a positive value while $\boldsymbol{\mu}_{n,j}(n \neq j)$ converges to a negative value, forcing $\mathcal{S}_{n,j}$ to converge to 0. The existing methods that perform well only on untrained models operate under the assumption $\boldsymbol{\mu}_n = \mathbf{0}$, which does not hold for well-trained models. Experimental results that empirically verify Assumption 3.1 are provided in Appendix B.3.

In each global round of training, the server leverages global model $\boldsymbol{\theta}^{(t)}$ to obtain estimates of the parameters in (7), $\bar{\boldsymbol{\mu}}_n$ and $\bar{\boldsymbol{\Sigma}}_n$, via a Monte Carlo method run on a small auxiliary dataset $\mathcal{A}$: the samples from $\mathcal{A}$ are processed by $\boldsymbol{\theta}^{(t)}$ and the resulting output logits are used to empirically compute the mean and variance. Given the estimates of the parameters, the server samples M data points $\mathbf{q}^{(n,i)} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}_n, \bar{\boldsymbol{\Sigma}}_n)$ to infer $\mathcal{S}_{n,j}$ as

$$\bar{\mathcal{S}}_{n,j} = \frac{1}{M}\sum_{i=1}^{M}\frac{\exp(\mathbf{q}_j^{(n,i)})}{\sum_{c=1}^{N}\exp(\mathbf{q}_c^{(n,i)})}. \tag{8}$$

3.2.1. Single Epoch Local Training

When $m = 1$, we omit superscript τ and let $\Delta\mathbf{b}_j = -\frac{\eta}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\nabla\mathbf{b}_j^{(i)}$. By taking the expectation of $\Delta\mathbf{b}_j$, we obtain (details provided in Appendix A.3)

$$\mathbb{E}\left[\Delta\mathbf{b}_j\right] = \frac{\eta}{|\mathcal{B}|}\left(N_j\sum_{n\neq j}\mathcal{S}_{j,n} - \sum_{n\neq j}N_n\mathcal{S}_{n,j}\right), \tag{9}$$

where N_j denotes the number of samples in $\mathcal{B}$ with label j. Finally, N_j is estimated by solving

$$\begin{aligned} &\min_{\mathbf{z}\in\mathbb{R}^N} \|\mathbf{A}\mathbf{z} - \mathbf{u}\|_2^2 \\ &\text{s.t.} \quad \mathbf{0} \leq \mathbf{z} \leq \mathbf{1} \wedge \|\mathbf{z}\|_1 = 1, \end{aligned} \tag{10}$$

where $\mathbf{u} = \Delta\mathbf{b}/\eta$, the diagonal entries of the coefficient matrix $\mathbf{A}$ are $a_{j,j} = \sum_{n\neq j}\mathcal{S}_{j,n}$, and the (n,j) off-diagonal entry of $\mathbf{A}$ is $a_{n,j} = -\bar{\mathcal{S}}_{n,j}$. After finding the solution $\mathbf{z}^\star$ to the above problem, we estimate N_j as $\bar{N}_j = \lfloor|\mathcal{B}|\cdot\mathbf{z}_j^\star\rceil$.

3.2.2. Multiple Epochs of Local Training

To reduce communication bandwidth, clients in FL typically update their local models over multiple epochs (with multiple batches of data), as illustrated in Eq. 5. Assume fixed learning rate η and constant batch size across all epochs

(i.e., $|\mathcal{B}^\tau| = |\mathcal{B}|$ for all τ); then the expectation of the local update $\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right]$ in global round t can be found as

$$\frac{\eta}{|\mathcal{B}|}\sum_{\tau=1}^{m}\left(N_j^{(t,\tau)}\sum_{n\neq j}\mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n\neq j}N_n^{(t,\tau)}\mathcal{S}_{n,j}^{(t,\tau)}\right), \quad (11)$$

where $N_j^{(t,\tau)}$ is the number of samples with label j in epoch τ and $\mathcal{S}_{n,j}^{(t,\tau)}$ denotes the expected erroneous confidence of local model $\boldsymbol{\theta}_k^{(t,\tau-1)}$ on class j. At the beginning of local training, $\boldsymbol{\theta}_k^{(t,0)}$ is initialized with the global model $\boldsymbol{\theta}^{(t)}$. Note that $\mathcal{S}_{n,j}^{(t,1)}$ can be estimated using global model $\boldsymbol{\theta}^{(t)}$ while $\mathcal{S}_{n,j}^{(t,m+1)}$ is readily inferred using the collected local model $\boldsymbol{\theta}_k^{(t,m)}$ via previously described Monte Carlo procedure. However, the intermediate states $\mathcal{S}_{n,j}^{(t,2)},\ldots,\mathcal{S}_{n,j}^{(t,m)}$ are unknown since the server does not have access to local models $\boldsymbol{\theta}_k^{(t,1)},\ldots,\boldsymbol{\theta}_k^{(t,m-1)}$. Since the evolution of $\mathcal{S}_{n,j}^{(t,\tau)}$ is non-linear, trivial interpolation between $\mathcal{S}_{j,n}^{(t,1)}$ and $\mathcal{S}_{j,n}^{(t,m+1)}$ may be highly inaccurate. To this end, we propose a practical method that relies on dynamics of the model parameter updates to approximate $\mathcal{S}_{j,n}^{(t,\tau)}$ in the intermediate epochs. Suppose we know $\mathcal{S}_{j,n}^{(t,\tau)}$ and the numbers of samples with different labels $N_j^{(t,\tau)}$, $j \in [N]$. Then

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t,\tau)}\right] = \frac{\eta}{|\mathcal{B}|}\left(N_j^{(t,\tau)}\sum_{n\neq j}\mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n\neq j}N_n^{(t,\tau)}\mathcal{S}_{n,j}^{(t,\tau)}\right). \quad (12)$$

Since the gradients of the weights in the output layer satisfy $\nabla \mathbf{W}_{j,l} = \nabla \mathbf{b}_j \cdot \bar{\mathbf{e}}_l$, where $\bar{\mathbf{e}}_l$ is the l-th component of the average embedded signal, we can estimate the change of the output logit as (the proof is provided in Appendix A.2)

$$\mathbb{E}\left[\Delta \mathbf{q}_j^{(n)}\right] = \Delta \boldsymbol{\mu}_{n,j}^{(t,\tau)} = \mathbb{E}\left[\Delta \mathbf{b}_j^{(t,\tau)}\right] \cdot \sum_{l=1}^{L}\bar{\mathbf{e}}_l^2, \quad (13)$$

and then obtain $\boldsymbol{\mu}_{n,j}^{(t,\tau+1)} = \boldsymbol{\mu}_{n,j}^{(t,\tau)} + \Delta\boldsymbol{\mu}_{n,j}^{(t,\tau)}$. Using $\boldsymbol{\mu}_{n,j}^{(t,\tau+1)}$ in the next local epoch, one can estimate $\mathcal{S}_{n,j}^{(t,\tau+1)}$ according to Eq. 8. By recursively conducting the above procedure, one can estimate all the intermediate states $\mathcal{S}_{n,j}^{(t,\tau)}$. However, $N_j^{(t,\tau)}$ and $\bar{\mathbf{e}}_l$ are not known – only the average updates of weight $\Delta \mathbf{W}_{j,l}^{(t)}$ and bias $\Delta \mathbf{b}_j^t$ are given. A closer examination of the correlation between $\nabla \mathbf{W}_{j,l}^{(t)}$ and $\nabla \mathbf{b}_j^t$ suggests estimating the average embedded signal according to

$$\bar{\mathbf{e}}_l \approx \Delta \mathbf{W}_{j,l}^{(t)} / \Delta \mathbf{b}_j^{(t)}. \quad (14)$$

To estimate the total number $\{\bar{N}_j^{(t)}\}_{j=1}^N$ of labels in m sampled batches we first set $N_j^{(t,\tau)} = \mathbf{g}_j$, where $\mathbf{g} \in \mathbb{N}^N$ denotes an arbitrarily vector (a guess) satisfying $\|\mathbf{g}\|_1 = |\mathcal{B}|$. As described earlier in this subsection, if we knew $N_j^{(t,\tau)}$ we could dynamically update $\mathcal{S}_{j,n}^{(t,\tau)}$ to arrive at $\bar{\mathcal{S}}_{j,n}^{(t,m+1)}$. Since the true $\mathcal{S}_{j,n}^{(t,m+1)}$ is known by the server (collected after the final epoch), the difference between $\bar{\mathcal{S}}_{j,n}^{(t,m+1)}$ and $\mathcal{S}_{j,n}^{(t,m+1)}$ could be used to adjust $\mathbf{g}_j$ and subsequently improve the estimate $\bar{\mathcal{S}}_{j,n}^{(t,m+1)}$. If $\bar{\mathcal{S}}_{j,n}^{(t,m+1)}$ is significantly smaller than $\mathcal{S}_{j,n}^{(t,m+1)}$, $\mathbf{g}_j$ was underestimated; otherwise, $\mathbf{g}_j$ was overestimated. Intuitively, local model tends to label input data as class j if the samples with label j are dominant in the batches sampled for training.

After a number of iterations, the difference between $\bar{\mathcal{S}}_{j,n}^{(t,m+1)}$ and $\mathcal{S}_{j,n}^{(t,m+1)}$ becomes small and $m \cdot \mathbf{g}_j$ closely approximates $N_j^{(t)}$. To accelerate the search for $\mathbf{g}$, one can introduce $\bar{\mathcal{S}}_{j,n}^{(t)} = (\mathcal{S}_{j,n}^{(t,1)} + \mathcal{S}_{j,n}^{(t,m+1)})/2$ and solve optimization (10) parameterized by $\mathcal{S}_{j,n} = \bar{\mathcal{S}}_{j,n}^{(t)}$ to obtain an initial estimate $\bar{N}_j^{(t)}$ which, in turn, is used to initialize $\mathbf{g}_j = \bar{N}_j^{(t)}/m$. In our experiments, following such an initialization RLU achieves highly accurate performance after only $T = 5$ iterations. The algorithms described in this section are formalized in Appendix C.1 and C.2.

3.3. FL schemes beyond FedAvg and SGD

The prior works on GI attacks in FL focused on FedAvg (McMahan et al., 2017), where local training pursues minimization of the CE loss, $\mathcal{L}_{\text{ce}}$. Deterioration of the performance of FedAvg observed in non-i.i.d. settings motivated a number of studies (Li et al., 2020; Karimireddy et al., 2020; Acar et al., 2021; Gao et al., 2022) that address the challenge of data heterogeneity by introducing various regularization terms to the local objective function. In particular, those methods consider objectives that are combination of the empirical risk and a regularizer, i.e.,

$$\mathcal{L}_{\text{local}} = \mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{regularizer}}. \quad (15)$$

For such objectives, the local updates of the bias collected by a server are not proportional to the gradients of $\mathcal{L}_{\text{ce}}$, adversely affecting the efficacy of the existing methods that attempt to recover labels from gradients. Furthermore, the existing label recovery methods assume that the local models are updated using SGD optimizers. When optimizers other than SGD are used, local updates are generally not proportional to the gradients of $\mathcal{L}_{\text{ce}}$. To explore such settings, we analyze the expectation of local updates in several milestone non-i.i.d. FL schemes and consider two well-known variants of SGD (Ruder, 2016), SGD with momentum (SGDm) and Nesterov accelerated gradient method (NAG). To accommodate these more general cases, we rephrase the expectation of the j-th component of the local update of bias $\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right]$

Table 1: Coefficients $\rho^{(\tau)}$ and $\mathbf{h}_j^{(\tau)}$ for different FL schemes including FedAvg (McMahan et al., 2017), Scaffold (Karimireddy et al., 2020), FedProx (Li et al., 2020), FedDyn (Acar et al., 2021) and FedDC (Gao et al., 2022). Here m denotes the number of local epochs, τ is the index of a local epoch, η is the learning rate, and $\Delta\mathbf{b}_j^{(r)}$ denotes the j-th component of the local update of bias $\mathbf{b}$ in global round r. For FedDC, $\Delta\mathbf{B}_j^{(r)}$ denotes the j-th component of the global update of bias $\mathbf{b}$ while $\mathbf{d}_k^{(t)}$ is the local drift in global round t. The server in each of the schemes can collect the corresponding variables and potentially use them in a label recovery attack.

Schemes	Optimizer	$\rho^{(\tau)}$	$\mathbf{h}_j^{(t)}$	Regularizer
FedAvg	SGD	1	$\mathbf{0}$	No
	SGDm	$\frac{1-\gamma^{m+1-\tau}}{1-\gamma}$	$\mathbf{0}$	No; γ is the momentum weight
	NAG	$\frac{1-\gamma^{m+2-\tau}}{1-\gamma}$	$\mathbf{0}$	No; γ is the momentum weight
Scaffold	SGD	1	$\eta m \sum_{r=2}^{t} \mathbf{c}^{(r)} + \sum_{r=1}^{t-1} \Delta\mathbf{b}_j^{(r)}$	No; $\mathbf{c}^{(r)}$ is the server control variate in global round r
FedProx	SGD	$(1-\lambda\eta)^{m-\tau}$	$\mathbf{0}$	$\frac{\lambda}{2}\left\|\theta_k^{(t,\tau)} - \theta^{(t)}\right\|^2$
FedDyn	SGD	$(1-\lambda\eta)^{m-\tau}$	$(1-(1-\lambda\eta)^m)\sum_{r=1}^{t-1}\Delta\mathbf{b}_j^{(r)}$	$\frac{\lambda}{2}\left\|\theta_k^{(t,\tau)} - \theta^{(t)}\right\|^2 - \left\langle \nabla\mathcal{L}_{\text{ce}}^{(t-1,m)}, \theta_k^{(t,\tau)} \right\rangle$
FedDC	SGD	$(1-\lambda\eta)^{m-\tau}$	$(1-(1-\lambda\eta)^m)\sum_{r=1}^{t-1}\Delta\mathbf{b}_j^{(r)} + \frac{1-(1-\lambda\eta)^m}{\lambda\eta m}\left(\Delta\mathbf{b}_j^{(t-1)} - \Delta\mathbf{B}_j^{(t-1)}\right)$	$\frac{\lambda}{2}\left\|\theta_k^{(t,\tau)} - (\theta^{(t)} - \mathbf{d}_k^{(t)})\right\|^2 + \frac{1}{\eta m}\left\langle \theta_k^{(t,\tau)}, \Delta\theta_k^{(t-1)} - \Delta\theta^{(t-1)} \right\rangle$

in global round t as

$$\frac{\eta}{|\mathcal{B}|}\sum_{\tau=1}^{m}\rho^{(\tau)}\left(N_j^{(t,\tau)}\sum_{n\neq j}\mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n\neq j}N_n^{(t,\tau)}\mathcal{S}_{n,j}^{(t,\tau)}\right) - \mathbf{h}_j^{(t)}, \tag{16}$$

where $\rho^{(\tau)}$ is a constant that depends on hyper-parameters used by different methods and remains constant across global rounds, and $\mathbf{h}_j^{(t)}$ is a term capturing historical training information (past local updates). We summarize the values of $\rho^{(\tau)}$ and $\mathbf{h}_j^{(t)}$ for different FL schemes in Table 1 (derivations provided in Appendix A.4-A.9). To the best of our knowledge, this is the first work that studies label recovery attacks in FL schemes beyond FedAvg. In FL schemes that utilize only the gradient information computed in the current global round (e.g., FedAvg and FedProx) $\mathbf{h}_j^t = \mathbf{0}$, while for the schemes that also rely on the past gradient information (Scaffold, FedDyn and FedDC) $\mathbf{h}_j^t \neq \mathbf{0}$. In any case, $\rho^{(\tau)}$ and $\mathbf{h}_j^{(t)}$ are known to the server and may potentially be used for label recovery attacks. In our work, we rely on the procedure for label recovery from local updates discussed in the previous section to run RLU attacks on different FL schemes where $\rho^{(\tau)}$ and $\mathbf{h}_j^t$ vary from one scheme to another according to Table 1.

4. Experiments

4.1. Setups

We evaluate the performance of RLU on a classification task using a variety of model architectures including LeNet-5 (LeCun et al., 1998), VGG-16 (Simonyan & Zisserman, 2014) and ResNet-50 (He et al., 2016), and four benchmark datasets including SVHN (Netzer et al., 2011), CIFAR10, CIFAR100 and Tiny-ImageNet (Le & Yang, 2015). Throughout these experiments we employ a number of activation functions including ReLU, Tanh, ELU (Clevert et al., 2015), SELU (Klambauer et al., 2017) and SiLU (Elfwing et al., 2018) to further evaluate robustness of our proposed method. To simulate diverse FL scenarios, we follow the strategy in (Yurochkin et al., 2019) and utilize Dirichlet distribution with a concentration parameter α, controlling the level of data heterogeneity across 10 data partitions owned by 10 clients. Unless specified otherwise, the models are trained using SGD optimizers. The auxiliary dataset $\mathcal{A}$ contains 100 samples per class. Further experimental details are provided in Appendix B.2.

4.2. Baselines and Evaluation Metrics

We compare our proposed RLU to three state-of-the-art methods: LLG (Wainakh et al., 2021), ZLG (Geng et al., 2021) and iRLG (Ma et al., 2023), all capable of recovering repeated labels in a batch. For fairness, we compare RLU to LLG+ and ZLG+; the latter two utilize the same auxiliary dataset $\mathcal{A}$ as RLU to achieve improved performance. Following the strategy of iRLG, we quantify the performance of a label recovery attack using two metrics: (1) *class-level accuracy (cAcc)*: the proportion of correctly recovered classes; (2) *instance-level accuracy (iAcc)*: the proportion of correctly recovered labels. Details of computing cAcc and iAcc are provided in Appendix B.4. We

Table 2: Comparison of class-level accuracy (cAcc) and instance-level accuracy (iAcc) of various methods in label recovery attacks on **untrained** models. The concentration parameter α controlling data partitioning is set to 0.5 in experiments on SVHN and CIFAR10, and to 0.1 on CIFAR100 and Tiny-ImageNet. The optimizer used in all experiments is SGD.

Model	Dataset	Batch Size	Activation	LLG+		ZLG+		iRLG		**RLU (ours)**	
				cAcc	iAcc	cAcc	iAcc	cAcc	iAcc	cAcc	iAcc
single local epoch $m = 1$											
LeNet-5	SVHN	32	ReLU	0.980	0.984	0.810	0.906	0.998	**1.000**	**1.000**	**1.000**
			Tanh	0.199	0.063	0.314	0.831	0.994	**1.000**	**1.000**	**1.000**
Vgg-16	CIFAR10	64	ReLU	0.982	0.981	0.907	0.982	0.961	0.979	**1.000**	**1.000**
			ELU	0.845	0.925	0.962	0.932	0.920	0.936	**0.994**	**0.996**
	CIFAR100	256	ReLU	0.960	0.932	0.888	0.943	0.981	0.992	**1.000**	**1.000**
			SELU	0.267	0.104	0.893	0.855	0.958	0.982	**1.000**	**1.000**
ResNet-50	Tiny	256	ReLU	0.998	0.938	0.880	0.666	0.992	0.995	**1.000**	**1.000**
			SiLU	0.953	0.794	0.847	0.534	0.989	**1.000**	**1.000**	**1.000**
multiple local epochs $m = 10$											
LeNet-5	SVHN	32	ReLU	0.863	0.934	0.496	0.644	0.996	0.997	**1.000**	**1.000**
			Tanh	0.218	0.164	0.551	0.699	0.976	0.996	**1.000**	**1.000**
Vgg-16	CIFAR10	64	ReLU	0.674	0.875	0.671	0.866	0.843	0.717	**0.845**	**0.946**
			ELU	0.688	0.812	0.682	0.798	0.808	0.650	**0.866**	**0.923**
	CIFAR100	256	ReLU	0.902	0.948	0.693	0.929	0.463	0.902	**0.922**	**0.981**
			SELU	0.072	0.014	0.400	0.840	0.466	0.689	**0.843**	**0.904**
ResNet-50	Tiny	256	ReLU	0.929	0.921	0.643	0.723	0.792	0.937	**0.957**	**0.976**
			SiLU	0.922	0.840	0.589	0.759	0.983	0.984	**0.991**	**0.996**

recover batch labels from the local updates computed on clients' local dataset and report the average cAcc and iAcc.

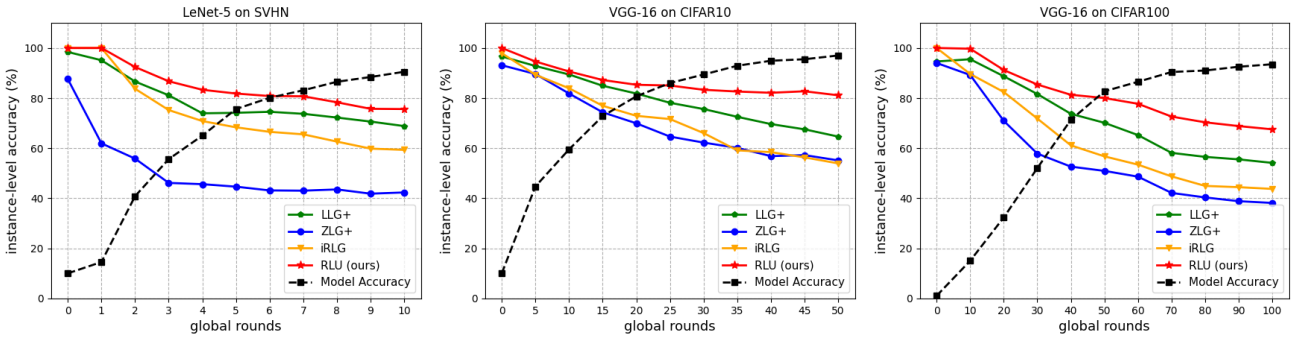
4.3. Attack on Untrained Models

A randomly initialized untrained model may be extremely vulnerable to label recovery attacks; for instance, if the server knows how the training is initialized, it does not even need an auxiliary data set to infer the parameters (e.g., $\boldsymbol{\mu}_n$) used in the attack. To compare the performance of baseline methods with that of RLU, we conduct comprehensive experiments on untrained models across various architectures, datasets, batch-sizes and activation functions. The results, reported in Table 2, demonstrate that RLU outperforms the baselines in all settings. When local training consists of a single epoch, RLU achieves near-perfect accuracy across the board in terms of both cAcc and iAcc; iRLG is a close second, outperforming other baselines. Since LLG+ assumes non-negative activation functions, its performance deteriorates significantly with Tanh and SELU. On Tiny-ImageNet, ZLG+ performs the worst among the four methods, achieving under 70% iAcc.

When the clients run $m = 10$ local epochs, performance of all methods deteriorates (as expected based on the discussion in Section 3.2.2). Nevertheless, the results in Table 2 show that RLU still outperforms the baselines, maintaining at least 84% cACC and 90% iAcc on all datasets, architectures and activation functions. While iRLG maintains solid performance on SVHN and Tiny-ImageNet, it performs significantly worse on CIFAR10 and CIFAR100 (iAcc falls below 70%, cAcc drops below 50%). The results of LLG+ follow the same pattern exhibited in single epoch settings, while ZLG+ experiences significant performance deterioration on SVHN. For consistency and a comparison with the results in Table 2, unless stated otherwise, in the remainder of this section the number of local epochs m is set to 10.

4.4. Attack on Trained Models

Since the accuracy of the global model improves as the training proceeds, comparing methods in terms of attacks on untrained model becomes no longer meaningful. As discussed in Section 3.2, distribution of output logits depends on the accuracy of the model which is parameterized by mean $\boldsymbol{\mu}_n$ and covariance $\boldsymbol{\Sigma}_n$. When the model becomes highly accurate, magnitudes of local gradients (updates) start vanishing; this, in turn, causes high sensitivity to noise and a large error in estimating the labels. All of the methods in Table 2 experience performance deterioration as the model accuracy increases. To compare their robustness, we conduct extensive experiments and evaluate performance of different attack methods at different stages of training. As shown in Fig. 1, RLU outperforms the baselines at all training stages, achieving 80% iAcc when the model accuracy reaches 80%. For the global model achieving 90% training accuracy, iAcc of iRLG reaches 75.6%, 83.3% and 72.6% on SVHN, CIFAR10 and CIFAR100, respectively. The iAcc of ZLG+ is 33.3%, 21.1% and 30.1% lower than RLU's, while iRLG's trails RLU by 16.3%, 17.3% and 23.9% on these three datasets, respectively. The iAcc performance of LLG+ is better than that of other prior methods but still falls significantly behind RLU's.



(a) SVHN (b) CIFAR10 (c) CIFAR100

Figure 1: Instance-level accuracy of different attack methods deteriorates as training progresses. Each point on the black dashed curve indicates the training accuracy of the global model in each global round.

Table 3: Experiments on CIFAR10 comparing iAcc of different methods on data partitions generated by varying α.

	Data heterogeneity α				
Schemes	0.05	0.1	0.5	1	5
LLG+	0.766	0.831	0.882	0.905	0.941
ZLG+	0.747	0.812	0.867	0.897	**0.944**
iRLG	0.449	0.630	0.715	0.766	0.820
RLU	**0.961**	**0.947**	**0.944**	**0.943**	0.931

4.5. Effect of Data Heterogeneity

Data heterogeneity is one the main challenges in real-world applications of federated learning. Prior works on label reconstruction evaluated their proposed methods on i.i.d. data, but the evaluation on non-i.i.d. data has remained largely unexplored. To this end, we benchmark the methods considered in this paper on CIFAR10 data partitions generated for varied values of $\alpha = \{0.05, 0.1, 0.5, 1, 5\}$ (smaller α corresponds to higher level of heterogeneity). We visualize the generated data partitions in Appendix B.5. Table 3 shows that iAcc of the three baselines monotonically decreases with the level of data heterogeneity. On the other hand, RLU demonstrate a great degree of robustness as it maintains high iAcc across the board; in particular, RLU achieves 93% or higher instance-level accuracy in all settings, including at the highest level of data heterogeneity ($\alpha = 0.05$).

4.6. Effect of the Size of Auxiliary Dataset $\mathcal{A}$

As previously discussed, RLU needs an auxiliary dataset $\mathcal{A}$ to estimate moments of the output logits distribution. In the benchmarking experiments presented thus far, we used 100 samples for each class in $\mathcal{A}$. To analyze the effect of the size of auxiliary dataset on the performance of RLU, we conduct 4 sets of experiments that utilize 4 auxiliary datasets with:

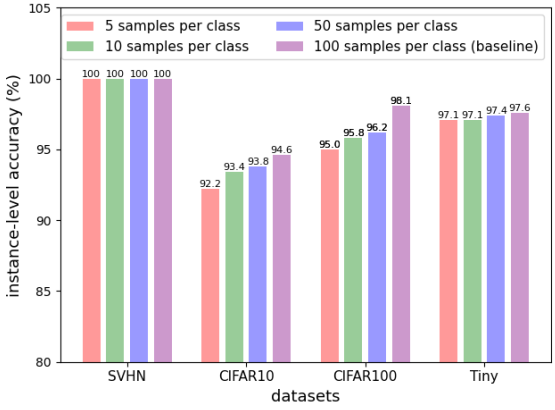


Figure 2: The iAcc of RLU utilizing auxiliary dataset $\mathcal{A}$ as the number of samples per class varies.

(1) 5 samples per class; (2) 10 samples per class; (3) 50 samples per class; and (4) 100 samples per class. As shown in Fig. 2, there appears to be no significant performance degradation due to reduction of the auxiliary data set size. When using the smallest among the auxiliary sets, on SVHN and Tiny the proposed RLU achieves performance close to the baseline. The largest performance gap is on CIFAR10 and CIFAR100, and even there the gap is only 2.4% and 3.1%, respectively. Therefore, RLU exhibits robustness with respect to the variations in the size of the auxiliary data set used by the server.

4.7. Effect of Distribution Shift in Auxiliary Dataset $\mathcal{A}$

To evaluate robustness of RLU to a domain shift in auxiliary data, we conducted experiments where we used RLU to attack models trained on CIFAR10 with auxiliary data subsampled from either CIFAR10.1 (Recht et al., 1806) or CIFAR10.2 (Lu et al., 2020) datasets. As the experimental results reported in Table 5 show, when the amount of

Table 4: Instance-level accuracy achieved by various methods in label recovery attacks on different FL schemes.

Schemes	Dataset	LLG+	ZLG+	iRLG	**RLU (ours)**	LLG+	ZLG+	iRLG	**RLU (ours)**
Hyper-parameter:		$\lambda = 0.5$				$\lambda = 5$			
FedProx	SVHN	0.917	0.656	0.982	**0.991**	0.079	0.566	0.881	**0.973**
	CIFAR10	0.747	0.795	0.691	**0.938**	0.665	0.682	0.616	**0.899**
	CIFAR100	0.805	0.791	0.594	**0.930**	0.659	0.626	0.493	**0.929**
Hyper-parameter:		$\gamma = 0.1$				$\gamma = 0.9$			
SGDm	SVHN	0.951	0.520	0.819	**0.987**	0.737	0.870	0.114	**0.916**
	CIFAR10	0.884	0.872	0.919	**0.935**	0.883	0.871	0.450	**0.928**
	CIFAR100	0.855	0.826	0.856	**0.915**	0.864	0.859	0.755	**0.882**
Hyper-parameter:		$\lambda = 0.5, t = 2$				$\lambda = 0.5, t = 6$			
FedDyn	SVHN	0.185	0.676	0.819	**0.921**	0.084	0.567	0.738	**0.863**
	CIFAR10	0.861	0.859	0.693	**0.919**	0.842	0.833	0.667	**0.889**
	CIFAR100	0.748	0.742	0.536	**0.861**	0.626	0.616	0.402	**0.861**
Hyper-parameter:		$t = 2$				$t = 6$			
Scaffold	SVHN	0.377	0.477	0.428	**0.874**	0.229	0.541	0.426	**0.814**
	CIFAR10	0.796	0.805	0.507	**0.853**	0.590	0.785	0.499	**0.817**
	CIFAR100	0.821	0.798	0.286	**0.858**	0.806	0.783	0.286	**0.860**

Table 5: iAcc of the RLU attack on models trained on CIFAR10 with the help of an auxiliary dataset $\mathcal{A}$ sampled from either CIFAR10.1 or CIFAR10.2 as the number of samples per class varies.

		Number of Samples per Class			
$\mathcal{A}$	Activation	5	10	50	100
C10.1	ReLU	0.915	0.915	0.911	0.924
C10.2	ELU	0.597	0.837	0.855	0.862
C10.1	ReLU	0.688	0.909	0.918	0.915
C10.2	ELU	0.596	0.813	0.852	0.851

auxiliary data is either 10, 50, or 100 samples per class, RLU experiences relatively small performance degradation compared to the results in Table 2. However, a noticeable deterioration occurs when only 5 samples per class are used; this is due to the difficulty of estimating parameters of the distribution that underwent a shift using a very small number of samples.

4.8. Attacks on Different FL Schemes

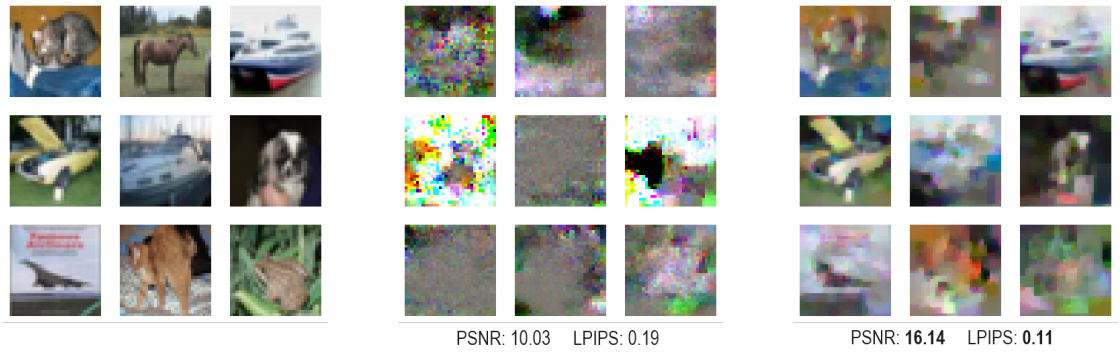
To the best of our knowledge, prior works evaluate their methods only on FedAvg. While FedAvg is indeed the oldest and perhaps the most widely used FL scheme, a number of other FL schemes has grown to prominence yet remains largely unexplored in the context of label recovery. As discussed in Section 3.3, we provide a framework to conduct label recovery attacks on several regularization-based FL schemes that may be using various optimizers. Table 4 shows the superior performance of RLU in the experiments on FedProx, SGDm, FedDyn and Scaffold. As can be seen from the table, in the experiments on FedProx and SGDm where $\lambda = 0.5$ and $\gamma = 0.1$ (which leads to $\rho^{(\tau)} \approx 1$), the three baselines achieve performance similar to that in their attacks on FedAvg. However, as $\rho^{(\tau)}$ deviates from 1 when $\lambda = 5$ and $\gamma = 0.9$, performance of the baselines severely deteriorates while RLU maintain its iAcc of at least 88%. In the experiments conducted on FedDyn and Scaffold, which rely on historical training data to update global model, RLU demonstrates capability to maintain its performance levels.

4.9. Improved Gradient Inversion Attacks with RLU

The outstanding performance of RLU on label recovery can improve the gradient inversion attacks in federated learning. In the prior work IG (Geiping et al., 2020), an HBC server performs joint optimization of the reconstructed labels $\mathbf{y}'$ and images $\mathbf{x}'$, which typically results in slow convergence and poor quality of the reconstructed images. We conduct gradients inversion attack experiments on CIFAR10, where we use RLU to estimate labels $\mathbf{y}'$ from local updates and only optimize the reconstructed images $\mathbf{x}'$ according to Eq. 2. To quantitatively characterize the quality of reconstructed images, we compute the peak signal-to-noise ratio (PSNR) and the learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018) of the reconstructed images in each batch. As illustrated in Fig. 3, images reconstructed with the help of RLU have higher PSNR and lower LPIPS, indicating smaller distance to the original images.

4.10. Defense with Differentially Private Noise

To defend against the proposed label recovery approach, clients can use differential privacy (DP) mechanism (Abadi et al., 2016) to encrypt the transmitted gradients. In particular, in such scenarios a zero-mean Gaussian noise with variance σ^2 may be injected into the gradients to provide a certified level of privacy. To investigate the impact of DP noise added to gradients, we conducted a series of experiments using RLU across three datasets. As reported in



(a) Ground Truth (b) IG (Joint Optimization) (c) Improved by RLU

Figure 3: Batch image reconstruction (batch size set to 9) on CIFAR10 compared to IG (Geiping et al., 2020). We select the best reconstructed batch for visualization and display the average metrics of the selected batches.

Table 6: we report iAcc of RLU conducted on gradients injected with DP noise in different level of privacy, characterized by variance σ^2.

		Standard Deviation σ			
Dataset	Epochs	0.05	0.1	0.2	0.5
SVHN	1	0.968	0.942	0.905	0.812
	10	0.844	0.727	0.606	0.484
CIFAR10	1	0.909	0.908	0.902	0.830
	10	0.906	0.877	0.792	0. 622
CIFAR100	1	0.999	0.977	0.933	0.831
	10	0.854	0.732	0.592	0.409

Table 6, higher variance of DP noise enhances the level of protection, leading to less accurate label recovery attacks. Noticeably, DP noise more effectively helps mitigate the RLU attacks in multi-epoch than in single-epoch settings; this is expected since the posterior search algorithm relies on the gradients which in the former case may be significantly perturbed by the DP noise.

5. Conclusion

In this paper, we studied label recovery attacks on federated learning systems in which clients send their local model updates to the server for aggregation. We developed RLU, a novel label recovery method which solves a least-square problem constructed by examining the correlation between the number of samples of each label type and local updates of the output layer. We extended the proposed framework to real-world scenarios involving well-trained models, multiple local epochs, high levels of data heterogeneity and various local objective functions, and provided theoretical analysis of RLU in different FL schemes. Comprehensive experiments on four datasets, three model architectures and six activation functions demonstrate consistently high accuracy, robustness and universality of RLU. Moreover, gradients inversion attack experiments illustrate that utilizing RLU may significantly improve quality of the reconstructed data in term of two widely-used metrics, PSNR and LPIPS. Future work will include exploring defense mechanism that may help ameliorate safety concerns caused by RLU.

Acknowledgements

This work was in part supported by the National Science Foundation under grant no. 2148224.

Impact Statement

Our paper investigates a novel method for label recovery attacks on Federated Learning (FL), with the goal of revealing security vulnerabilities in FL and no intentions to exploit these vulnerabilities to harm the community. As demonstrated in the paper, a straightforward way to limit the vulnerability to attacks by RLU is achieved by increasing the number of local epochs and injecting differentially private (DP) noise to the gradients. Nevertheless, there are many other defense mechanisms such as gradient sparsification and secure aggregation that can be utilized to defend against label recovery attacks. We are committed to exploring the defense to attacks on FL in our future work.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., and Saligrama, V. Federated learn-

ing based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Dang, T., Thakkar, O., Ramaswamy, S., Mathews, R., Chin, P., and Beaufays, F. Revealing and protecting labels in distributed training. *Advances in Neural Information Processing Systems*, 34:1727–1738, 2021.

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

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Fang, H., Chen, B., Wang, X., Wang, Z., and Xia, S.-T. Gifd: A generative gradient inversion method with feature domain optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4967–4976, 2023.

Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., and Xu, C.-Z. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10112–10121, 2022.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Geng, J., Mou, Y., Li, F., Li, Q., Beyan, O., Decker, S., and Rong, C. Towards general deep leakage in federated learning. *arXiv preprint arXiv:2110.09074*, 2021.

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

Jeon, J., Lee, K., Oh, S., Ok, J., et al. Gradient inversion with generative image prior. *Advances in neural information processing systems*, 34:29898–29908, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

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

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Lin, J. On the dirichlet distribution. *Department of Mathematics and Statistics, Queens University*, 40, 2016.

Lu, S., Nott, B., Olson, A., Todeschini, A., Vahabi, H., Carmon, Y., and Schmidt, L. Harder or different? a closer look at distribution shift in dataset reproduction. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, volume 5, pp. 15, 2020.

Ma, K., Sun, Y., Cui, J., Li, D., Guan, Z., and Liu, J. Instance-wise batch label restoration via gradients in federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

Melis, L., Song, C., De Cristofaro, E., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*, pp. 691–706. IEEE, 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do cifar-10 classifiers generalize to cifar-10? 2018. *URL https://arxiv. org/abs*, 1806.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Wainakh, A., Ventola, F., Müßig, T., Keim, J., Cordero, C. G., Zimmer, E., Grube, T., Kersting, K., and Mühlhäuser, M. User-level label leakage from gradients in federated learning. *arXiv preprint arXiv:2105.09369*, 2021.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pp. 7252–7261. PMLR, 2019.

Zhang, C., Xiaoman, Z., Sotthiwat, E., Xu, Y., Liu, P., Zhen, L., and Liu, Y. Generative gradient inversion via over-parameterized networks in federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5126–5135, 2023.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A. Proof of Analytical Results

A.1. Derivation of the gradient of the output layer's bias b

Given a batch of samples $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{B}$, the cross-entrop loss can be computed as

$$\mathcal{L}_{\text{ce}} == \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\mathcal{L}_{\text{ce}}^{(i)} = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\log\frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}, \tag{17}$$

$$\mathbf{q}_j^{(i)} = \sum_{l=1}^{L}\mathbf{W}_{j,l}\mathbf{z}_l^{(i)} + \mathbf{b}_j, \tag{18}$$

where N is the number of classes; $\mathbf{q}_j^{(i)}$ denotes the j-th component of output logits given sample $\mathbf{x}^{(i)}$; $\mathbf{W}_{j,l}$ is the (j,l)-element of weights of the output layer; $\mathbf{b}_j$ is the j-th component of bias of the output layer; $\mathbf{z}^{(i)}$ is the embedding of data $\mathbf{x}^{(i)}$; L is the dimension of embedding space. The gradient of the bias $\mathbf{b}_j$ given sample $(\mathbf{x}^{(i)}, y^{(i)})$ can be computed using the chain rule as

$$\frac{\partial\mathcal{L}_{\text{ce}}^{(i)}}{\partial\mathbf{b}_j} = \frac{\partial\mathcal{L}_{\text{ce}}^{(i)}}{\partial\mathbf{Q}}\cdot\frac{\partial\mathbf{Q}}{\partial\mathbf{q}_j^{(i)}}\cdot\frac{\partial\mathbf{q}_j^{(i)}}{\partial\mathbf{b}_j}, \tag{19}$$

where

$$\mathbf{Q} = \frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}. \tag{20}$$

To start, note that

$$\frac{\partial\mathcal{L}_{\text{ce}}^{(i)}}{\partial\mathbf{Q}} = -\frac{1}{\mathbf{Q}}, \frac{\partial\mathbf{q}_j^{(i)}}{\partial\mathbf{b}_j} = 1. \tag{21}$$

If $j = y^{(i)}$, we obtain

$$\frac{\partial\mathbf{Q}}{\partial\mathbf{q}_j^{(i)}} = \frac{\exp\left(\mathbf{q}_{y^{(i)}}^{(i)}\right)\left(\sum_{n=1}^{N}\exp\left(\mathbf{q}_n^{(i)}\right)\right) - \exp\left(\mathbf{q}_{y^{(i)}}^{(i)}\right)^2}{\left(\sum_{n=1}^{N}\exp\left(\mathbf{q}_n^{(i)}\right)\right)^2} = \frac{\sum_{n\neq y^{(i)}}\exp\left(\mathbf{q}_n^{(i)}\right)}{\sum_{n=1}^{N}\exp\left(\mathbf{q}_n^{(i)}\right)}\cdot\mathbf{Q}, \tag{22}$$

while if $j \neq y^{(i)}$,

$$\frac{\partial\mathbf{Q}}{\partial\mathbf{q}_j^{(i)}} = -\frac{\exp\left(\mathbf{q}_{y^{(i)}}^{(i)}\right)\exp\left(\mathbf{q}_i^{(i)}\right)}{\left(\sum_{n=1}^{N}\exp\left(\mathbf{q}_n^{(i)}\right)\right)^2} = -\frac{\exp\left(\mathbf{q}_i^{(i)}\right)}{\sum_{n=1}^{N}\exp\left(\mathbf{q}_n^{(i)}\right)}\cdot\mathbf{Q}. \tag{23}$$

By substituting Eq. 22 and Eq. 23 in Eq. 19, we obtain

$$\nabla\mathbf{b}_j^{(i)} = \begin{cases} \dfrac{\exp(\mathbf{q}_j^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}, \text{ if } j \neq y^{(i)}, \\ -\dfrac{\sum_{n\neq j}\exp(\mathbf{q}_n^{(i)})}{\sum_{n=1}^{N}\exp(\mathbf{q}_n^{(i)})}, \text{ if } j = y^{(i)}. \end{cases} \tag{24}$$

■

A.2. Derivation of the gradient of the output layer's weight $\mathbf{W}$

According to Eq. 17 and Eq. 18, gradients of the weight $\mathbf{W}_{j,l}$ given sample $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ can be computed as

$$\frac{\partial\mathcal{L}_{\text{ce}}^{(i)}}{\partial\mathbf{W}_{j,l}} = \frac{\partial\mathcal{L}_{\text{ce}}^{(i)}}{\partial\mathbf{Q}}\cdot\frac{\partial\mathbf{Q}}{\partial\mathbf{q}_j^{(i)}}\cdot\frac{\partial\mathbf{q}_j^{(i)}}{\partial\mathbf{W}_{j,l}}. \tag{25}$$

Since, as discussed in Section A.1, we know $\frac{\partial \mathcal{L}_{\text{ce}}^{(i)}}{\partial \mathbf{Q}}$ and $\frac{\partial \mathbf{Q}}{\partial \mathbf{q}_j^{(i)}}$, we have $\frac{\partial \mathbf{q}_j^{(i)}}{\partial \mathbf{W}_{j,l}} = \mathbf{z}_j^{(i)}$. Finally, we obtain

$$\nabla \mathbf{W}_{j,l}^{(i)} = \nabla \mathbf{b}_j^{(i)} \cdot \mathbf{z}_l^{(i)}. \tag{26}$$

■

A.3. Derivation of the expected local update of the output layer's bias b (single epoch)

Suppose mini-batch stochastic gradient descent (SGD) is used as the optimizer in FL training; the local update computed in a single epoch can be found as

$$\Delta \mathbf{b}_j = -\eta \nabla \mathbf{b}_j = -\frac{\eta}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \nabla \mathbf{b}_j^{(i)}. \tag{27}$$

Taking expectation of both sides yields

$$\begin{aligned}
\mathbb{E}\left[\Delta \mathbf{b}_j\right] &= -\frac{\eta}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \mathbb{E}\left[\nabla \mathbf{b}_j^{(i)}\right] \\
&= \frac{\eta}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left(\mathbb{I}\{j = y^{(i)}\} \mathbb{E}\left[\frac{\sum_{n \neq j} \exp(\mathbf{q}_n^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})}\right] - \mathbb{I}\{j \neq y^{(i)}\} \mathbb{E}\left[\frac{\exp(\mathbf{q}_j^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})}\right] \right) \\
&= \frac{\eta}{|\mathcal{B}|} \left(N_j \sum_{n \neq j} \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}_k^{(j)}} \left[\mathbf{s}_n(\mathbf{x})\right] - \sum_{n \neq j} N_n \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}_k^{(n)}} \left[\mathbf{s}_j(\mathbf{x})\right] \right) \\
&= \frac{\eta}{|\mathcal{B}|} \left(N_j \sum_{n \neq j} \mathcal{S}_{j,n} - \sum_{n \neq j} N_n \mathcal{S}_{n,j} \right).
\end{aligned} \tag{28}$$

Let $\mathcal{D}_k^{(-j)}$ denote the subset of local dataset $\mathcal{D}_k$ that excludes the samples with label j, and let $\mathcal{S}_j = \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}_k^{(-j)}} \left[\mathbf{s}_j(\mathbf{x})\right]$ (can be approximated by $\mathcal{S}_j \approx \frac{1}{N-1} \sum_{n \neq j} \mathcal{S}_{n,j}$). It follows that

$$\begin{aligned}
\sum_{j=1}^{N} \mathcal{S}_j &= \sum_{j=1}^{N} \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{D}_k^{(-j)}} \left[\mathbf{s}_j(\mathbf{x})\right] \\
&\approx \frac{1}{N-1} \sum_{j=1}^{N} \underbrace{\sum_{n \neq j} \frac{1}{|\mathcal{D}_k| \cdot P_n} \sum_{i=1}^{|\mathcal{D}_k|} \mathcal{I}\{y^{(i)} = n\} \frac{\exp(\mathbf{q}_j^{(i)})}{\sum_{c=1}^{N} \exp(\mathbf{q}_c^{(i)})}}_{\text{reformulate}} \\
&= \frac{1}{N-1} \sum_{j=1}^{N} \frac{1}{|\mathcal{D}_k| \cdot P_j} \sum_{i=1}^{|\mathcal{D}_k|} \mathcal{I}\{y^{(i)} = j\} \frac{\sum_{n \neq j} \exp(\mathbf{q}_j^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})} \\
&= \frac{1}{N-1} \sum_{j=1}^{N} \left(1 - \frac{1}{|\mathcal{D}_k| \cdot P_j} \sum_{i=1}^{|\mathcal{D}_k|} \mathcal{I}\{y^{(i)} = j\} \frac{\exp(\mathbf{q}_j^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})} \right) \\
&= \frac{N}{N-1} - \frac{1}{(N-1)|\mathcal{D}_k|} \sum_{j=1}^{N} \sum_{i=1}^{|\mathcal{D}_k|} \frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})} \cdot \frac{\mathcal{I}\{y^{(i)} = j\}}{P_j},
\end{aligned} \tag{29}$$

where P_j is the proportion of samples with label j in dataset $\mathcal{D}_k$. Note that the second term in Eq. 29 is positively correlated to $\mathcal{L}_{\text{ce}}$ specified in Eq. 17. Given an untrained neural network model, we have

$$\frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})} \approx \frac{1}{N}, \forall y^{(i)} \in [N], \tag{30}$$

and thus we obtain

$$\sum_{j=1}^{N} \mathcal{S}_j \approx \frac{N}{N-1} - \frac{1}{N-1} \sum_{j=1}^{N} \frac{1}{N} \frac{|\mathcal{D}_k| \cdot P_j}{|\mathcal{D}_k| \cdot P_j} = \frac{N-1}{N-1} = 1. \tag{31}$$

Assume that after T global rounds $\mathcal{L}_{ce}^{\star} = 0$; it follows that

$$\frac{\exp(\mathbf{q}_{y^{(i)}}^{(i)})}{\sum_{n=1}^{N} \exp(\mathbf{q}_n^{(i)})} = 1, \forall y^{(i)} \in [N], \tag{32}$$

and thus we obtain

$$\sum_{j=1}^{N} \mathcal{S}_j \approx \frac{N}{N-1} - \frac{1}{N-1} \sum_{j=1}^{N} 1 \cdot \frac{|\mathcal{D}_k| \cdot P_j}{|\mathcal{D}_k| \cdot P_j} = \frac{N}{N-1} - \frac{N}{N-1} = 0. \tag{33}$$

■

A.4. Derivation of the expected local update using SGD with momentum (SGDm)

In each local epoch τ, SGD with momentum (Ruder, 2016) updates the model parameters $\theta_k^{(t,\tau)}$ of client k according to

$$\mathbf{v}_k^{(t,\tau)} \leftarrow \gamma \mathbf{v}_k^{(t,\tau-1)} + \nabla \mathcal{L}_{\text{ce}}^{(t,\tau)}, \tag{34}$$

$$\theta_k^{(t,\tau)} \leftarrow \theta_k^{(t,\tau-1)} - \eta \mathbf{v}_k^{(t,\tau)}, \tag{35}$$

where η is the learning rate in global round t; $\mathbf{v}^{(t,\tau)}$ is the momentum in local epoch τ; $\gamma \in (0, 1)$ is the momentum weight. Then the local update of client k takes form

$$\Delta\theta_k^{(t)} = -\eta \sum_{\tau=1}^{m} \mathbf{v}_k^{(t,\tau)}. \tag{36}$$

We can compute $\mathbf{v}_k^{(t,\tau)}$ in Eq. 34 by mathematical induction, with $\mathbf{v}_k^{(t,1)}$ initialized as $\nabla \mathcal{L}_{\text{ce}}^{(t,1)}$. In the following, we will prove that

$$\mathbf{v}_k^{(t,\tau)} = \sum_{i=1}^{\tau} \gamma^{\tau-i} \nabla \mathcal{L}_{\text{ce}}^{(t,i)}. \tag{37}$$

Proof: According to Eq. 34,

$$\mathbf{v}_k^{(t,2)} = \gamma \mathbf{v}_k^{(t,1)} + \nabla \mathcal{L}_{\text{ce}}^{(t,2)} = \gamma \nabla \mathcal{L}_{\text{ce}}^{(t,1)} + \nabla \mathcal{L}_{\text{ce}}^{(t,2)}, \tag{38}$$

$$\mathbf{v}_k^{(t,3)} = \gamma \mathbf{v}_k^{(t,2)} + \nabla \mathcal{L}_{\text{ce}}^{(t,3)} = \gamma^2 \nabla \mathcal{L}_{\text{ce}}^{(t,1)} + \gamma \nabla \mathcal{L}_{\text{ce}}^{(t,2)} + \nabla \mathcal{L}_{\text{ce}}^{(t,3)}, \tag{39}$$

both satisfying Eq. 37. Assuming that for any $\tau \geq 2$

$$\mathbf{v}_k^{(t,\tau)} = \sum_{i=1}^{\tau} \gamma^{\tau-i} \nabla \mathcal{L}_{\text{ce}}^{(t,i)}, \tag{40}$$

we can obtain $\mathbf{v}_k^{(t,\tau+1)}$ by following Eq. 34,

$$\begin{aligned} \mathbf{v}_k^{(t,\tau+1)} &= \gamma \mathbf{v}_k^{(t,\tau)} + \nabla \mathcal{L}_{\text{ce}}^{(t,\tau+1)} \\ &= \sum_{i=1}^{\tau} \gamma^{\tau+1-i} \nabla \mathcal{L}_{\text{ce}}^{(t,i)} + \nabla \mathcal{L}_{\text{ce}}^{(t,\tau+1)} \\ &= \sum_{i=1}^{\tau+1} \gamma^{\tau+1-i} \nabla \mathcal{L}_{\text{ce}}^{(t,i)}, \end{aligned} \tag{41}$$

which proves our claim in Eq. 37. Thus we have

$$\Delta\theta_k^{(t)} = -\eta \sum_{\tau=1}^{m} \sum_{i=1}^{\tau} \gamma^{\tau-i} \nabla \mathcal{L}_{\text{ce}}^{(t,i)}. \tag{42}$$

Similar to the discussion regarding updating bias in the output layer, we obtain

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|}\sum_{\tau=1}^{m}\sum_{i=1}^{\tau}\gamma^{\tau-i}\left(N_j^{(t,\tau)}\sum_{n\neq j}\mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n\neq j}N_n^{(t,\tau)}\mathcal{S}_{n,j}^{(t,\tau)}\right). \tag{43}$$

Reformulating Eq. 43 yields

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|}\sum_{\tau=1}^{m}\rho^{(\tau)}\left(N_j^{(t,\tau)}\sum_{n\neq j}\mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n\neq j}N_n^{(t,\tau)}\mathcal{S}_{n,j}^{(t,\tau)}\right), \tag{44}$$

where

$$\rho^{(\tau)} = \frac{1-\gamma^{m+1-\tau}}{1-\gamma}. \tag{45}$$

■

A.5. Derivation of the expected local update using Nesterov accelerated gradient method (NAG)

In each local epoch τ, Nesterov's accelerated gradient method (Ruder, 2016) updates the model parameters $\theta_k^{(t,\tau)}$ of client k according to

$$\mathbf{v}_k^{(t,\tau)} \leftarrow \gamma \mathbf{v}_k^{(t,\tau-1)} + \nabla\mathcal{L}_{\text{ce}}^{(t,\tau)} + \gamma\left(\nabla\mathcal{L}_{\text{ce}}^{(t,\tau)} - \nabla\mathcal{L}_{\text{ce}}^{(t,\tau-1)}\right), \tag{46}$$

$$\theta_k^{(t,\tau)} \leftarrow \theta_k^{(t,\tau-1)} - \eta\mathbf{v}_k^{(t,\tau)}, \tag{47}$$

where η is the learning rate in global round t; $\mathbf{v}^{(t,\tau)}$ is the momentum in local epoch τ; $\gamma \in (0,1)$ is the momentum weight. We form the local update of client k as

$$\Delta\theta_k^{(t)} = -\eta\sum_{\tau=1}^{m}\mathbf{v}_k^{(t,\tau)}. \tag{48}$$

One can compute $\mathbf{v}_k^{(t,\tau)}$ according to Eq. 46 via mathematical induction, where $\mathbf{v}_k^{(t,1)}$ is initialized as $(1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,1)}$. In the following, we will prove that

$$\mathbf{v}_k^{(t,\tau)} = \sum_{i=1}^{\tau-1}\gamma^{\tau-i+1}\nabla\mathcal{L}_{\text{ce}}^{(t,i)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,\tau)}, \tau \geq 2. \tag{49}$$

Proof: According to Eq. 46,

$$\begin{aligned}\mathbf{v}_k^{(t,2)} &= \gamma\mathbf{v}_k^{(t,1)} + \nabla\mathcal{L}_{\text{ce}}^{(t,2)} + \gamma\left(\nabla\mathcal{L}_{\text{ce}}^{(t,2)} - \nabla\mathcal{L}_{\text{ce}}^{(t,1)}\right)\\ &= \gamma^2\nabla\mathcal{L}_{\text{ce}}^{(t,1)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,2)},\end{aligned} \tag{50}$$

$$\begin{aligned}\mathbf{v}_k^{(t,3)} &= \gamma\mathbf{v}_k^{(t,2)} + \nabla\mathcal{L}_{\text{ce}}^{(t,3)} + \gamma\left(\nabla\mathcal{L}_{\text{ce}}^{(t,3)} - \nabla\mathcal{L}_{\text{ce}}^{(t,2)}\right)\\ &= \gamma^3\nabla\mathcal{L}_{\text{ce}}^{(t,1)} + \gamma^2\nabla\mathcal{L}_{\text{ce}}^{(t,2)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,3)},\end{aligned} \tag{51}$$

both satisfying Eq. 49. Assuming that for any $\tau \geq 2$

$$\mathbf{v}_k^{(t,\tau)} = \sum_{i=1}^{\tau-1}\gamma^{\tau-i+1}\nabla\mathcal{L}_{\text{ce}}^{(t,i)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,\tau)}, \tag{52}$$

we obtain $\mathbf{v}_k^{(t,\tau+1)}$ according to Eq. 46 as

$$\begin{aligned}\mathbf{v}_k^{(t,\tau+1)} &= \gamma\mathbf{v}_k^{(t,\tau)} + \nabla\mathcal{L}_{\text{ce}}^{(t,\tau+1)} + \gamma\left(\nabla\mathcal{L}_{\text{ce}}^{(t,\tau+1)} - \nabla\mathcal{L}_{\text{ce}}^{(t,\tau)}\right)\\ &= \sum_{i=1}^{\tau-1}\gamma^{\tau+1-i+1}\nabla\mathcal{L}_{\text{ce}}^{(t,i)} + \gamma^2\nabla\mathcal{L}_{\text{ce}}^{(t,\tau)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,\tau+1)}\\ &= \sum_{i=1}^{\tau}\gamma^{\tau+1-i+1}\nabla\mathcal{L}_{\text{ce}}^{(t,i)} + (1+\gamma)\nabla\mathcal{L}_{\text{ce}}^{(t,\tau+1)},\end{aligned} \tag{53}$$

which proves our claim in Eq. 49 and thus

$$\Delta\theta_k^{(t)} = -\eta \sum_{\tau=2}^{m} \left(\sum_{i=1}^{\tau-1} \gamma^{\tau-i+1} \nabla \mathcal{L}_{\text{ce}}^{(t,i)} \right) - \eta(1+\gamma) \sum_{\tau=1}^{m} \nabla \mathcal{L}_{\text{ce}}^{(t,\tau)}. \tag{54}$$

Similar to the discussion regarding the update of bias in the output layer, we obtain

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|} \sum_{\tau=1}^{m} \rho^{(\tau)} \left(N_j^{(t,\tau)} \sum_{n \neq j} \mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n \neq j} N_n^{(t,\tau)} \mathcal{S}_{n,j}^{(t,\tau)} \right), \tag{55}$$

where

$$\rho^{(\tau)} = \frac{1-\gamma^{m+2-\tau}}{1-\gamma}. \tag{56}$$

■

A.6. Derivation of the expected local update of Scaffold

The local update of Scaffold (Karimireddy et al., 2020) in local epoch τ and global round t can be found as

$$\theta_k^{(t,\tau)} \leftarrow \theta_k^{(t,\tau-1)} - \eta \left(\nabla \mathcal{L}_{\text{ce}}^{(t,\tau)} - \mathbf{c}_k^{(t)} + \mathbf{c}^{(t)} \right), \tag{57}$$

$$\mathbf{c}_k^{(t+1)} \leftarrow \mathbf{c}_k^{(t)} - \mathbf{c}^{(t)} + \frac{1}{\eta m}(\theta^{(t)} - \theta_k^{(t,m)}), \tag{58}$$

where $\mathbf{c}_k^{(t)}$ is the client control variate while $\mathbf{c}^{(t)}$ is the server control variate in global round t; $\mathbf{c}_k^{(1)}$ is initialized by $\mathbf{c}^{(1)}$; m is the number of local epochs; η is the learning rate. According to the update rule of $\mathbf{c}_k^{(t+1)}$ in Eq. 58,

$$\begin{aligned}
\mathbf{c}_k^{(t)} &= \mathbf{c}_k^{(t-1)} - \mathbf{c}^{(t-1)} - \frac{1}{\eta m} \Delta\theta_k^{(t-1)} \\
&= \mathbf{c}_k^{(t-2)} - \mathbf{c}^{(t-2)} - \mathbf{c}^{(t-1)} - \frac{1}{\eta m} \Delta\theta_k^{(t-2)} - \frac{1}{\eta m} \Delta\theta_k^{(t-1)} \\
&= \dots \\
&= \mathbf{c}_k^{(1)} - \mathbf{c}^{(1)} - \sum_{r=2}^{t-1} \mathbf{c}^{(r)} - \frac{1}{\eta m} \sum_{r=1}^{t-1} \Delta\theta_k^{(r)} \\
&= -\sum_{r=2}^{t-1} \mathbf{c}^{(r)} - \frac{1}{\eta m} \sum_{r=1}^{t-1} \Delta\theta_k^{(r)},
\end{aligned} \tag{59}$$

where $\Delta\theta_k^{(r)}$ is the local update of client k in global round r. Then the local update of client k is given by

$$\Delta\theta_k^{(t)} = -\eta \sum_{\tau=1}^{m} \nabla \mathcal{L}_{\text{ce}}^{(t,\tau)} - \eta m \left(\sum_{r=2}^{t} \mathbf{c}^{(r)} + \frac{1}{\eta m} \sum_{r=1}^{t-1} \Delta\theta_k^{(r)} \right). \tag{60}$$

Therefore, the j-th component of the update of bias in the output layer can be found as

$$\Delta \mathbf{b}_j^{(t)} = -\eta \sum_{\tau=1}^{m} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} - \eta m \left(\sum_{r=2}^{t} \mathbf{c}^{(r)} + \frac{1}{\eta m} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \right), \tag{61}$$

where $\Delta \mathbf{b}_j^{(r)}$ is the update of $\mathbf{b}_j$ in global round $r < t$ (known by the server). Similar to the previous analysis, we obtain

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|} \sum_{\tau=1}^{m} \left(N_j^{(t,\tau)} \sum_{n \neq j} \mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n \neq j} N_n^{(t,\tau)} \mathcal{S}_{n,j}^{(t,\tau)} \right) - \mathbf{h}_j^{(t)}, \tag{62}$$

where

$$\mathbf{h}_j^{(t)} = \eta m \sum_{r=2}^{t} \mathbf{c}^{(r)} + \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)}. \tag{63}$$

■

A.7. Derivation of the expected local update of FedProx

The local objective function in FedProx (Li et al., 2020) is defined as

$$\mathcal{L}_{\text{prox}} = \mathcal{L}_{\text{ce}} + \frac{\lambda}{2} \left\| \theta_k^{(t,\tau)} - \theta^{(t)} \right\|^2, \tag{64}$$

where $\theta^{(t)}$ is the global model used to initialize local training; $\theta_k^{(t,\tau)}$ is the local model of client k in local epoch τ and $\theta_k^{(t,1)}$ is initialized as $\theta^{(t)}$; λ is a hyper-parameter. Therefore, the gradient of $\mathcal{L}_{\text{prox}}$ in local epoch τ can be computed as

$$\frac{\partial \mathcal{L}_{\text{prox}}^{(t,\tau)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} + \lambda \left(\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} \right), \tag{65}$$

where $\mathbf{b}_j^{(t,\tau)} \in \theta_k^{(t,\tau)}$ is the j-th component of bias $\mathbf{b}$ in the output layer in the local epoch τ. We assume the model is trained by an SGD optimizer, leading to

$$\begin{aligned}
\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} &= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{prox}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \mathbf{b}_j^{(t,1)} \\
&= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \lambda\eta \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} \right) - \mathbf{b}_j^{(t,1)} \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} + (1-\lambda\eta) \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} \right) \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \eta(1-\lambda\eta) \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-2)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^2 \left(\mathbf{b}_j^{(t,\tau-2)} - \mathbf{b}_j^{(t,1)} \right) \\
&= \dots \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^{\tau-1} \left(\mathbf{b}_j^{(t,1)} - \mathbf{b}_j^{(t,1)} \right) \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j},
\end{aligned} \tag{66}$$

and thus we obtain

$$\frac{\partial \mathcal{L}_{\text{prox}}^{(t,1)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,1)}}{\partial \mathbf{b}_j}, \frac{\partial \mathcal{L}_{\text{prox}}^{(t,\tau)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j}, \tau \geq 2. \tag{67}$$

Taking expectation of both sides yields

$$\begin{aligned}
\mathbb{E}\left[\Delta \mathbf{b}_j^t\right] &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{prox}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right] \\
&= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right] + \lambda\eta^2 \sum_{\tau=2}^{m} \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} \right] \\
&= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right] + \lambda\eta^2 \sum_{\tau=1}^{m-1} \frac{1-(1-\lambda\eta)^{m-\tau}}{\lambda\eta} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right] \\
&= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right] + \eta \sum_{\tau=1}^{m-1} \left(1-(1-\lambda\eta)^{m-\tau}\right) \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} \right],
\end{aligned} \tag{68}$$

which is readily reformulated as

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|} \sum_{\tau=1}^{m} \rho^{(\tau)} \left(N_j^{(t,\tau)} \sum_{n \neq j} \mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n \neq j} N_n^{(t,\tau)} \mathcal{S}_{n,j}^{(t,\tau)} \right), \tag{69}$$

where

$$\rho^{(\tau)} = (1-\lambda\eta)^{m-\tau}. \tag{70}$$

■

A.8. Derivation of the expected local update of FedDyn

The local objective function in FedDyn (Acar et al., 2021) is defined as

$$\mathcal{L}_{\text{dyn}}^{(t,\tau)} = \mathcal{L}_{\text{ce}} - \left\langle \nabla \mathcal{L}_{\text{ce}}^{(t-1,m)}, \theta_k^{(t,\tau)} \right\rangle + \frac{\lambda}{2} \left\| \theta_k^{(t,\tau)} - \theta^{(t)} \right\|^2, \tag{71}$$

where $\theta^{(t)}$ is the global model used to initialize local training; $\theta_k^{(t,\tau)}$ is the local model of client k in local epoch τ and $\theta_k^{(t,1)} = \theta^{(t)}$; $\nabla \mathcal{L}_{\text{dyn}}^{(t-1,m)}$ is the gradient of $\mathcal{L}_{\text{dyn}}^{(t-1,m)}$ in the previous global round; λ is a hyper-parameter; m denotes the total number of local epochs. According to the first-order condition for local optima,

$$\nabla \mathcal{L}_{\text{ce}}^{(t,\tau)} - \nabla \mathcal{L}_{\text{ce}}^{(t-1,m)} + \lambda \left(\theta_k^{(t,\tau)} - \theta^{(t)} \right) = 0, \tag{72}$$

and thus the partial gradient of $\mathcal{L}_{\text{ce}}$ in local epoch m can be computed as

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{ce}}^{(t,m)}}{\partial \mathbf{b}_j} &= \frac{\partial \mathcal{L}_{\text{ce}}^{(t-1,m)}}{\partial \mathbf{b}_j} - \lambda \left(\mathbf{b}_j^{(t,m)} - \mathbf{b}_j^{(t,1)} \right) \\
&= \frac{\partial \mathcal{L}_{\text{ce}}^{(t-2,m)}}{\partial \mathbf{b}_j} - \lambda \left(\mathbf{b}_j^{(t-1,m)} - \mathbf{b}_j^{(t-1,1)} \right) - \lambda \left(\mathbf{b}_j^{(t,m)} - \mathbf{b}_j^{(t,1)} \right) \\
&= -\lambda \sum_{r=1}^{t} \left(\mathbf{b}_j^{(r,m)} - \mathbf{b}_j^{(r,1)} \right) \\
&= -\lambda \sum_{r=1}^{t} \Delta \mathbf{b}_j^{(r)},
\end{aligned} \tag{73}$$

where $\Delta \mathbf{b}_j^{(r)}$ is the update of $\mathbf{b}_j$ in global round $r < t$ (known by the server); η is the learning rate. Therefore, we obtain

$$\frac{\partial \mathcal{L}_{\text{dyn}}^{(t,\tau)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} + \lambda \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} + \lambda \left(\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} \right). \tag{74}$$

Considering

$$\begin{aligned}
\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} &= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{dyn}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \mathbf{b}_j^{(t,1)} \\
&= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \lambda\eta \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} \right) - \mathbf{b}_j^{(t,1)} - \lambda\eta \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} + (1-\lambda\eta) \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} \right) - \lambda\eta \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \eta(1-\lambda\eta) \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-2)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^2 \left(\mathbf{b}_j^{(t,\tau-2)} - \mathbf{b}_j^{(t,1)} \right) - \lambda\eta \sum_{i=1}^{2} (1-\lambda\eta)^{2-i} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
&= \dots \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^{\tau-1} \left(\mathbf{b}_j^{(t,1)} - \mathbf{b}_j^{(t,1)} \right) - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} - \left(1 - (1-\lambda\eta)^{\tau-1} \right) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)},
\end{aligned} \tag{75}$$

we obtain

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{dyn}}^{(t,\tau)}}{\partial \mathbf{b}_j} &= \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} + \lambda \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} - \lambda \left(1-(1-\lambda\eta)^{\tau-1}\right) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\ &= \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} + \lambda(1-\lambda\eta)^{\tau-1} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)}, \tau \geq 2.\end{aligned} \tag{76}$$

Taking expectation of both sides yields

$$\begin{aligned}\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{dyn}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] \\ &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] + \lambda\eta^2 \sum_{\tau=2}^{m} \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j}\right] - \mathbf{h}_j^{(t)} \\ &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] + \lambda\eta^2 \sum_{\tau=1}^{m-1} \frac{1-(1-\lambda\eta)^{m-\tau}}{\lambda\eta} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] - \mathbf{h}_j^{(t)} \\ &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] + \eta \sum_{\tau=1}^{m-1} \left(1-(1-\lambda\eta)^{m-\tau}\right) \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] - \mathbf{h}_j^{(t)},\end{aligned} \tag{77}$$

which is readily reformulated as

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|} \sum_{\tau=1}^{m} \rho^{(\tau)} \left(N_j^{(t,\tau)} \sum_{n \neq j} \mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n \neq j} N_n^{(t,\tau)} \mathcal{S}_{n,j}^{(t,\tau)} \right) - \mathbf{h}_j^{(t)}, \tag{78}$$

where

$$\rho^{(\tau)} = (1-\lambda\eta)^{m-\tau}, \tag{79}$$

$$\mathbf{h}_j^{(t)} = (1-(1-\lambda\eta)^m) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)}. \tag{80}$$

■

A.9. Derivation of the expected local update of FedDC

The local objective function in FedDC (Gao et al., 2022) is defined as

$$\mathcal{L}_{\text{dc}}^{(t,\tau)} = \mathcal{L}_{\text{ce}} + \frac{\lambda}{2} \left\| \theta_k^{(t,\tau)} - (\theta^{(t)} - \mathbf{d}_k^{(t)}) \right\|^2 + \frac{1}{\eta m} \left\langle \theta_k^{(t,\tau)}, \Delta\theta_k^{(t-1)} - \Delta\theta^{(t-1)} \right\rangle, \tag{81}$$

$$\mathbf{d}_k^{(t+1)} = \mathbf{d}_k^{(t)} + \theta_k^{(t,m)} - \theta^{(t)}, \tag{82}$$

where $\mathbf{d}_k^{(t)}$ denotes the local drift in global round t and is initialized as $\mathbf{d}_k^{(1)} = \mathbf{0}$; m is the number of local epochs; $\Delta\theta^{(t)}$ is the global model update in round t. According to the update rule of $\mathbf{d}_k^{(t+1)}$ in Eq. 82,

$$\begin{aligned}\mathbf{d}_k^{(t)} &= \mathbf{d}_k^{(t-1)} + \Delta\theta_k^{(t-1)} \\ &= \mathbf{d}_k^{(t-2)} + \Delta\theta_k^{(t-2)} + \Delta\theta_k^{(t-1)} \\ &= \dots \\ &= \mathbf{d}_k^{(1)} + \sum_{r=1}^{t-1} \Delta\theta_k^{(r)} \\ &= \sum_{r=1}^{t-1} \Delta\theta_k^{(r)},\end{aligned} \tag{83}$$

and thus the gradient of $\mathcal{L}_{\text{dc}}$ in local epoch τ can be computed as

$$\frac{\partial \mathcal{L}_{\text{dc}}^{(t,\tau)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} + \lambda \left(\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} + \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \right) + \frac{1}{\eta m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right), \tag{84}$$

where $\Delta \mathbf{B}_j^{(t-1)}$ denotes the global update of $\mathbf{b}_j$ in global round $t-1$. Then

$$\begin{aligned}
\mathbf{b}_j^{(t,\tau)} - \mathbf{b}_j^{(t,1)} &= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{dc}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \mathbf{b}_j^{(t,1)} \\
&= \mathbf{b}_j^{(t,\tau-1)} - \eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \lambda\eta \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} + \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \right) - \mathbf{b}_j^{(t,1)} - \frac{1}{m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right) \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} + (1-\lambda\eta) \left(\mathbf{b}_j^{(t,\tau-1)} - \mathbf{b}_j^{(t,1)} \right) - \lambda\eta \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \frac{1}{m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right) \\
&= -\eta \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-1)}}{\partial \mathbf{b}_j} - \eta(1-\lambda\eta) \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau-2)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^2 \left(\mathbf{b}_j^{(t,\tau-2)} - \mathbf{b}_j^{(t,1)} \right) \\
&\quad - \lambda\eta \sum_{i=1}^{2} (1-\lambda\eta)^{2-i} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \frac{1}{m} \sum_{i=1}^{2} (1-\lambda\eta)^{2-i} \sum_{r=1}^{t-1} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right) \\
&= \dots \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} + (1-\lambda\eta)^{\tau-1} (\mathbf{b}_j^{(t,1)} - \mathbf{b}_j^{(t,1)}) \\
&\quad - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \frac{1}{m} \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \sum_{r=1}^{t-1} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right) \\
&= -\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} - \left(1 - (1-\lambda\eta)^{\tau-1} \right) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
&\quad - \frac{1}{\lambda\eta m} \left(1 - (1-\lambda\eta)^{\tau-1} \right) \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right),
\end{aligned} \tag{85}$$

and thus we obtain

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{dc}}^{(t,\tau)}}{\partial \mathbf{b}_j} = \frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j} - \lambda\eta \sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i} \frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j} + \lambda (1-\lambda\eta)^{\tau-1} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} \\
+ \frac{(1-\lambda\eta)^{\tau-1}}{\eta m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)} \right).
\end{aligned} \tag{86}$$

Table 7: Comparison of RLU and state-of-the-art label recovery methods in FL. The "Batch" column indicates whether the method can recover labels in a batch or not; "Repeating Labels" column indicates if the method assumes no repeating labels in the batch; "Activation-agnostic" column indicates if the method can work beyond non-negative activation functions; "Multiple Epochs" column indicates if the method can recover labels from updates computed in multiple local epochs; "Trained Model" column indicates if the method preserve performance on well-trained models.

Schemes	Batch	Repeating Labels	Activation-agnostic	Multiple Epochs	Trained Model
iDLG (Zhao et al., 2020)	✗	✗	✗	✗	✔
GI (Yin et al., 2021)	✔	✗	✗	✗	✔
RLG (Dang et al., 2021)	✔	✗	✔	✔	✔
LLG (Wainakh et al., 2021)	✔	✔	✗	✗	✔
ZLG (Geng et al., 2021)	✔	✔	✗	✗	✗
iRLG (Ma et al., 2023)	✔	✔	✔	✗	✗
RLU (ours)	✔	✔	✔	✔	✔

Taking expectation of both sides yields

$$\begin{aligned}
\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] &= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{dc}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] \\
&= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] + \lambda\eta^2 \sum_{\tau=2}^{m}\sum_{i=1}^{\tau-1} (1-\lambda\eta)^{\tau-1-i}\, \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,i)}}{\partial \mathbf{b}_j}\right] \\
&\quad - \lambda\eta \sum_{\tau=1}^{m} (1-\lambda\eta)^{\tau-1} \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \sum_{\tau=1}^{m} \frac{(1-\lambda\eta)^{\tau-1}}{m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)}\right) \\
&= -\eta \sum_{\tau=1}^{m} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] + \lambda\eta^2 \sum_{\tau=1}^{m-1} \frac{1-(1-\lambda\eta)^{m-\tau}}{\lambda\eta} \mathbb{E}\left[\frac{\partial \mathcal{L}_{\text{ce}}^{(t,\tau)}}{\partial \mathbf{b}_j}\right] \\
&\quad - (1-(1-\lambda\eta)^m) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} - \frac{1-(1-\lambda\eta)^m}{\lambda\eta m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)}\right)
\end{aligned} \tag{87}$$

which is readily reformulated as

$$\mathbb{E}\left[\Delta \mathbf{b}_j^{(t)}\right] = \frac{\eta}{|\mathcal{B}|} \sum_{\tau=1}^{m} \rho^{(\tau)} \left(N_j^{(t,\tau)} \sum_{n \neq j} \mathcal{S}_{j,n}^{(t,\tau)} - \sum_{n \neq j} N_n^{(t,\tau)} \mathcal{S}_{n,j}^{(t,\tau)} \right) - \mathbf{h}_j^{(t)}, \tag{88}$$

where

$$\rho^{(\tau)} = (1-\lambda\eta)^{m-\tau}, \tag{89}$$

$$\mathbf{h}_j^{(t)} = (1-(1-\lambda\eta)^m) \sum_{r=1}^{t-1} \Delta \mathbf{b}_j^{(r)} + \frac{1-(1-\lambda\eta)^m}{\lambda\eta m} \left(\Delta \mathbf{b}_j^{(t-1)} - \Delta \mathbf{B}_j^{(t-1)}\right). \tag{90}$$

■

B. Experimental Details

B.1. Comparison with Prior Works

We qualitatively summarize capabilities of the state-of-the-art label recovery methods in Table 7. A check mark (✔) indicates that a method is suitable for a given setting while the cross mark (✗) indicates that it is not. As the table indicates, RLU is capable of operating in a variety of scenarios.

B.2. Experimental Settings

B.2.1. General Settings

We used Pytorch (Paszke et al., 2019) to implement all the described experiments. In the experiments involving SVHN, the clients used LeNet (LeCun et al., 1998), a convolutional neural network with 16 convolution layers and 5×5 kernel, as the classifier. The batch size was set to 32 in all experiments on SVHN. In the experiments involving CIFAR10 and CIFAR100, the clients trained Vgg-16 (Simonyan & Zisserman, 2014) with different scales of convolution kernel as the classifier. The batch size was set to 64 and 256 in the experiments on CIFAR10 and CIFAR100, respectively. In the experiments involving Tiny-ImageNet, the clients learned a standard ResNet-50 (He et al., 2016) where the batch size was set to 256. The number of clients was set to 10 across all the experiments.

B.2.2. Settings of the Experiments in Table 2

The learning rate η for the SGD optimizer was set to 0.01 in all experiments reported in Table 2. In order to obtain high level of data heterogeneity, we set the concentration parameters α to 0.5 in the experiments on SVHN and CIFAR10; in the experiments on CIFAR100 and Tiny-ImageNet, this parameter was set to 0.1. There are two groups of experiments in Table 2; in one the number of local epochs m was set to 1, while in the other it was set to 10. For the experiments involving multiple local epochs, the number of iterations T in Alg. 2 was set to 10.

B.2.3. Settings of the Experiments in Table 3

Here we explored the effect of data heterogeneity controlled by varying the concentration parameter α from 0.05 to 5. For a fair comparison, we set the number of local epochs to $m = 10$; other settings were identical to the settings in Table 2.

B.2.4. Settings of the Experiments in Table 4, Figure 1 and Figure 2

Similar to Table 2 and Table 3, we set the number of local epochs m to 10 in all experiments reported in Table 4, Figure 1 and Figure 2. The concentration parameter α was set to 0.5 on SVHN/CIFAR10 and to 0.1 on CIFAR100. The learning rate η was set to 0.01 in experiments on SVHN and to 0.05 in experiments on CIFAR10 and CIFAR100.

B.2.5. Settings of the Experiments in Gradients Inversion Attack

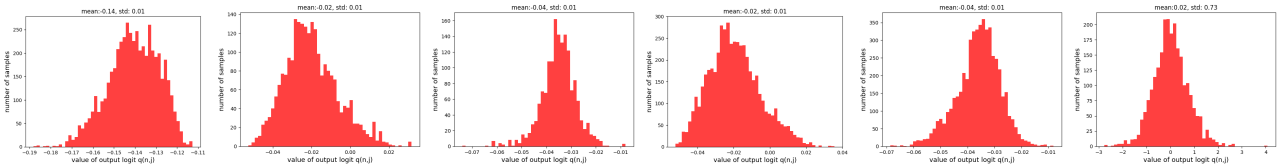
We conducted gradients inversion attack on the training set of CIFAR10 with an untrained ResNet32 model. We follow the strategy in IG (Geiping et al., 2020), using cosine similarity as the distance between ground-truth gradients and the estimated gradients computed using reconstructed images and labels. The batch size was set to 9 and the number of iterations for optimizing the reconstructed images and labels was set to 24000. We used Adam (Kingma & Ba, 2014) optimizer and set the learning rate to 0.1. We also added the total variation regularization (Yin et al., 2021) to the objective function to create more realistic images with a weight scalar 0.2.

B.3. Empirical Validation of Assumption 3.1

To verify Assumption 3.1, we perform inference on the training set of SVHN and CIFAR10 using the global model across different training rounds. We collect the j-th component of the output logits for the samples with label n, and organize them into 100 bins to generate histograms. In each histogram, x-axis indicates the values of the j-th component of the output logits while the values on y-axis indicate the corresponding number of samples. As shown in Figures 4 and 7, means $\boldsymbol{\mu}_{n,j}$ of the randomly initialized global model are close to 0 regardless of their sign. As the training accuracy of the global model improves, the means $\boldsymbol{\mu}_{j,j}$ increase to larger positive values while $\boldsymbol{\mu}_{n,j}(n \neq j)$ decreases to negative values, as shown in Figures 4-9.

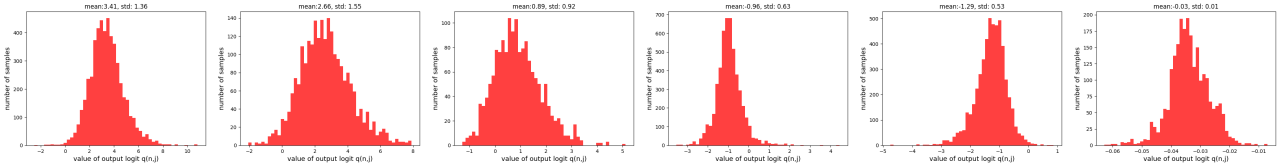
B.4. Evaluation Metrics

We follow the strategy in iRLG (Ma et al., 2023), evaluating all methods in terms of class-level accuracy (cAcc) and instance-level accuracy (iAcc). Specifically, cAcc indicates the proportion of correctly recovered classes while iAcc indicates the proportion of correctly recovered labels. Suppose $\mathbf{y}_c$ denotes the ground-truth classes that appear in the batches



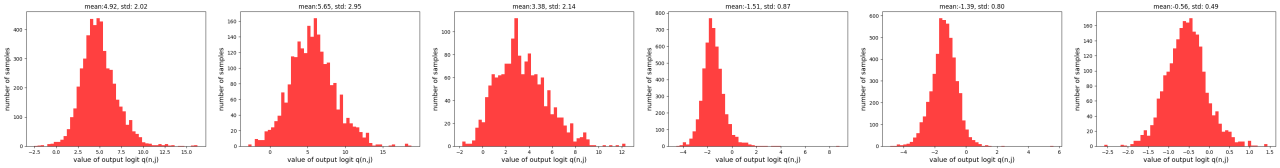
(a) $\boldsymbol{\mu}_{1,1}^{(0)} = -0.14$ (b) $\boldsymbol{\mu}_{5,5}^{(0)} = -0.02$ (c) $\boldsymbol{\mu}_{9,9}^{(0)} = -0.04$ (d) $\boldsymbol{\mu}_{1,5}^{(0)} = -0.02$ (e) $\boldsymbol{\mu}_{1,9}^{(0)} = -0.04$ (f) $\boldsymbol{\mu}_{5,9}^{(0)} = 0.02$

Figure 4: A selection of histograms that characterize distribution of the output logits in global round 0 (randomly initialized) on SVHN dataset. Specifically, each histogram illustrates the number of samples with label n as input that have a certain value of the j-th component of output logits. From left to right, the corresponding values of n and j are as follow: (a) $n = 1$, $j = 1$; (b) $n = 5$, $j = 5$; (c) $n = 9$, $j = 9$; (d) $n = 1$, $j = 5$; (e) $n = 1$, $j = 9$; (f) $n = 5$, $j = 9$.



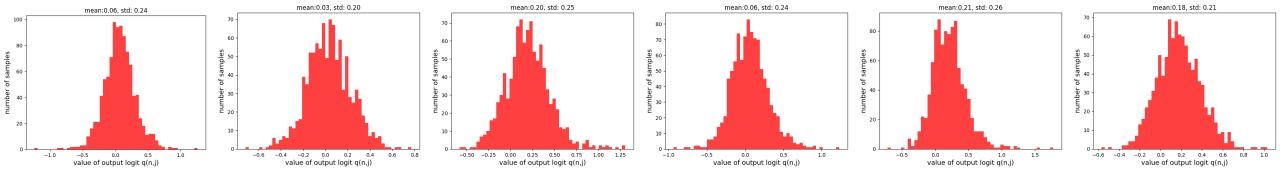
(a) $\boldsymbol{\mu}_{1,1}^{(4)} = 3.41$ (b) $\boldsymbol{\mu}_{5,5}^{(4)} = 2.66$ (c) $\boldsymbol{\mu}_{9,9}^{(4)} = 0.89$ (d) $\boldsymbol{\mu}_{1,5}^{(4)} = -0.96$ (e) $\boldsymbol{\mu}_{1,9}^{(4)} = -1.29$ (f) $\boldsymbol{\mu}_{5,9}^{(4)} = -0.03$

Figure 5: A selection of histograms that characterize distribution of the output logits in global round 4 in which the training accuracy of the global model is 68% on SVHN dataset. The other settings are identical to Fig. 4.



(a) $\boldsymbol{\mu}_{1,1}^{(9)} = 4.92$ (b) $\boldsymbol{\mu}_{5,5}^{(9)} = 5.65$ (c) $\boldsymbol{\mu}_{9,9}^{(9)} = 3.38$ (d) $\boldsymbol{\mu}_{1,5}^{(9)} = -1.51$ (e) $\boldsymbol{\mu}_{1,9}^{(9)} = -1.39$ (f) $\boldsymbol{\mu}_{5,9}^{(9)} = -0.56$

Figure 6: A selection of histograms that characterize distribution of the output logits in the global round 9 in which the training accuracy of the global model is 83% on SVHN dataset. The other settings are identical to Fig. 4.

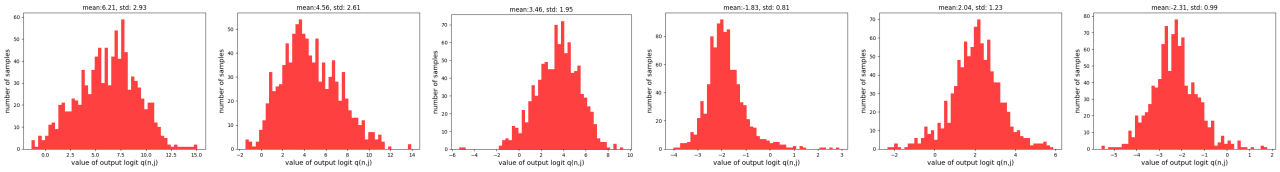


(a) $\boldsymbol{\mu}_{1,1}^{(0)} = 0.06$ (b) $\boldsymbol{\mu}_{5,5}^{(0)} = 0.03$ (c) $\boldsymbol{\mu}_{9,9}^{(0)} = 0.20$ (d) $\boldsymbol{\mu}_{1,5}^{(0)} = 0.06$ (e) $\boldsymbol{\mu}_{1,9}^{(0)} = 0.21$ (f) $\boldsymbol{\mu}_{5,9}^{(0)} = 0.18$

Figure 7: A selection of histograms that characterize distribution of the output logits in global round 0 on CIFAR10. The other settings are identical to Fig. 4.

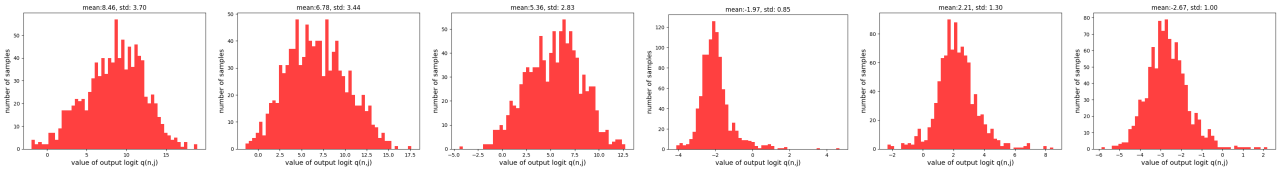
while $\hat{\mathbf{y}}_c$ denotes the estimated classes; then cAcc can be computed as

$$\mathbf{cAcc}(\hat{\mathbf{y}}_c, \mathbf{y}_c) = \frac{|\hat{\mathbf{y}}_c \ \mathbf{XNOR} \ \mathbf{y}_c|}{N}, \tag{91}$$



(a) $\boldsymbol{\mu}_{1,1}^{(20)} = 6.21$ (b) $\boldsymbol{\mu}_{5,5}^{(20)} = 4.56$ (c) $\boldsymbol{\mu}_{9,9}^{(20)} = 3.46$ (d) $\boldsymbol{\mu}_{1,5}^{(20)} = -1.83$ (e) $\boldsymbol{\mu}_{1,9}^{(20)} = 2.04$ (f) $\boldsymbol{\mu}_{5,9}^{(20)} = -2.31$

Figure 8: A part of histograms that characterize the distribution of output logits in the global round 20 in which the training accuracy of the global model is 80% on CIFAR10. The other settings are identical to Fig. 4.



(a) $\boldsymbol{\mu}_{1,1}^{(40)} = 8.46$ (b) $\boldsymbol{\mu}_{5,5}^{(40)} = 6.78$ (c) $\boldsymbol{\mu}_{9,9}^{(40)} = 5.36$ (d) $\boldsymbol{\mu}_{1,5}^{(40)} = -1.97$ (e) $\boldsymbol{\mu}_{1,9}^{(40)} = 2.21$ (f) $\boldsymbol{\mu}_{5,9}^{(40)} = -2.67$

Figure 9: A part of histograms that characterize the distribution of output logits in the global round 40 in which the training accuracy of the global model is 95% on CIFAR10. The other settings are identical to Fig. 4.

where N is the total number of classes. Similarly, suppose $\mathbf{y}_i$ denotes the ground-truth labels in the batches while $\hat{\mathbf{y}}_i$ denotes estimated labels; then iAcc can be computed as

$$\textbf{iAcc}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{|\hat{\mathbf{y}}_i \cap \mathbf{y}_i|}{m \cdot |\mathcal{B}|}, \tag{92}$$

where $|\hat{\mathbf{y}}_i| = |\mathbf{y}_i| = m \cdot |\mathcal{B}|$ and m denotes the number of local epochs.

B.5. Visualization of Data Heterogeneity

We follow the strategy in (Yurochkin et al., 2019), utilizing Dirichlet distribution to generate heterogeneous data partitions from the original datasets. Specifically, we assign different proportions $\mathbf{p}_k^{(j)}$ of samples with label j in the local dataset $\mathcal{D}_k$ to K clients according to

$$\mathbf{p}^{(j)} = \{\mathbf{p}_k^{(j)}, k \in [K]\} \sim \textbf{Dir}_K(\alpha), \tag{93}$$

where α is the concentration parameter that controls the level of heterogeneity. The number of samples with label j in client k's local dataset can be computed as

$$N_k^{(j)} = \frac{\mathbf{p}_k^{(j)}}{\sum_i^K \mathbf{p}_i^{(j)}} N^{(j)}, \tag{94}$$

where $N^{(j)}$ is the number of samples with label j in the overall training dataset. Figures 10 and 11 show the class distribution of clients' local dataset by color-coding the number of samples: the darker the color, the larger the number of samples with the corresponding label. As shown in Figures 10 and 11, clients own only 2 or 3 classes in their local dataset given $\alpha = 0.05$ while $\alpha = 5$ leads to more balanced class distribution.

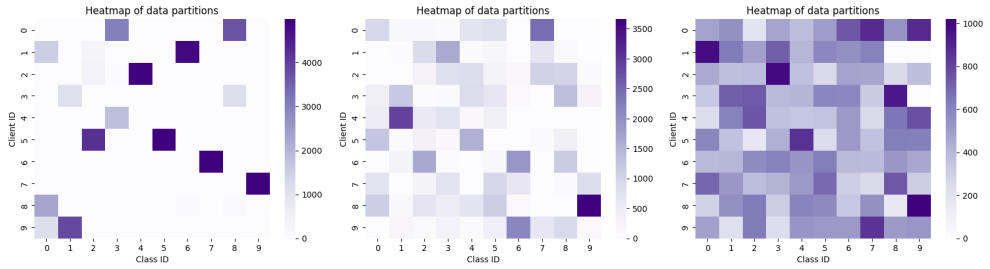
B.6. Selecting Values of α

To generate data partitions for the experiments reported in Table. 2, we attempted to create a scenario with "mild" heterogeneity – specifically, we aimed at partitions such that each client has 40-50% classes present in its local dataset. The first work (Yurochkin et al., 2019) (that we are aware of) using such a partitioning strategy sets $\alpha = 0.5$ as the default value. However, using the same value of the parameter ($\alpha = 0.5$) to partition CIFAR100 and Tiny-ImageNet would lead to different heterogeneity levels simply because these two datasets contain more classes. Formally, this is reflected in the

expression for the entropy of a Dirichlet variable $\mathbf{p}$ in Eq. 93 which, according to (Lin, 2016), is given by

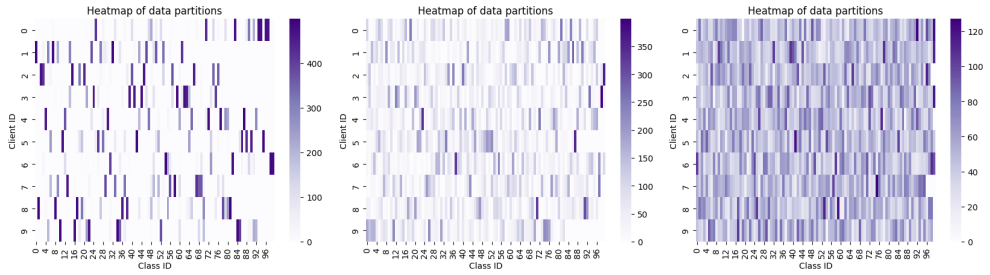
$$H(\mathbf{p}) = \ln \mathbf{B}(\alpha) + N \cdot (\alpha - 1) \cdot (\psi(N\alpha) - \psi(\alpha)) \,, \tag{95}$$

where N is the number of classes; $\mathbf{B}()$ is beta function; $\psi()$ is digamma function (strictly ascending in $\mathbb{R}^+$). Clearly, for a fixed value of $\alpha < 1$, $H(\mathbf{p})$ is smaller for a larger N; smaller entropy suggests $\mathbf{p}$ is more homogeneous. For this reason, in the experiments on CIFAR100 and Tiny-ImageNet we set $\alpha = 0.1$.



(a) $\alpha = 0.05$ (b) $\alpha = 0.5$ (c) $\alpha = 5$

Figure 10: Training data from CIFAR10 is split into 10 partitions according to a Dirichlet distribution. The concentration parameter is set as follows: (a) $\alpha = 0.05$; (b) $\alpha = 0.5$; (c) $\alpha = 5$.



(a) $\alpha = 0.05$ (b) $\alpha = 0.5$ (c) $\alpha = 5$

Figure 11: Training data from CIFAR100 is split into 10 partitions according to a Dirichlet distribution. The concentration parameter is set as follows: (a) $\alpha = 0.05$; (b) $\alpha = 0.5$; (c) $\alpha = 5$.

C. Algorithms

C.1. RLU Algorithm

In this section, we formalize the RLU algorithm as Alg.1. In the standard paradigm in federated learning, the server broadcasts the global model $\boldsymbol{\theta}^{(t)}$ to selected clients; these clients then initialize their local model $\boldsymbol{\theta}_k^{(t)}$ to start local training. After m local epochs, each client k computes the local update $\Delta\boldsymbol{\theta}_k^{(t)} = \boldsymbol{\theta}_k^{(t)} - \boldsymbol{\theta}^{(t)}$ including $\Delta\mathbf{W}_k^{(t)}$ and $\Delta\mathbf{b}_k^{(t)}$. For convenience, we omit the subscript k in $\Delta\mathbf{W}_k^{(t)}$ and $\Delta\mathbf{b}_k^{(t)}$ in Alg.1. According to the formulated problem in Section 3.1, the learning rate η in local training is known by the server. While preparing to conduct label recovery attack, the server estimates $\boldsymbol{\mu}_n^{(t)}, \Sigma_n^{(t)}, \mathcal{S}_{n,j}^{(t)}, \boldsymbol{\mu}_{n,j}^{(t+1)}, \Sigma_n^{(t+1)}$, and $\mathcal{S}_{n,j}^{(t+1)}, \forall n, j \in [N]$ via Monte Carlo method according to Eq. 8 with the global model $\boldsymbol{\theta}^{(t)}$, updated local model $\boldsymbol{\theta}_k^{(t)}$ and the auxiliary dataset $\mathcal{A}$. If $m = 1$, the server simply solves the least-squares problem described in Eq. 10. If $m > 1$, the server first solves the least-squares problem using coefficients computed as the mean of $\mathcal{S}_{n,j}^{(t)}$ and $\mathcal{S}_{n,j}^{(t+1)}$ to obtain a crude estimate. Then the server conducts posterior search (Alg. 2) to adjust the estimated $N^{(t)}$ as discussed in Section 3.2.2.

Algorithm 1 RLU

Input: Auxiliary dataset $\mathcal{A}$, the global model $\boldsymbol{\theta}^{(t)}$, local updates $\Delta\boldsymbol{\theta}_k^{(t)}$ including $\Delta\mathbf{W}^{(t)}$ and $\Delta\mathbf{b}^{(t)}$, learning rate η.

Output: The number of samples with each label $N_j^{(t)}$ in the batches sampled by client k, where $\forall j \in [N]$.

Initial: Estimate $\boldsymbol{\mu}_n^{(t)}, \boldsymbol{\Sigma}_n^{(t)}, \mathcal{S}_{n,j}^{(t)}, \boldsymbol{\mu}_{n,j}^{(t+1)}, \boldsymbol{\Sigma}_n^{(t+1)}$, and $\mathcal{S}_{n,j}^{(t+1)}, \forall n, j \in [N]$ via Monte Carlo method according to Eq.8 with the global model $\boldsymbol{\theta}^{(t)}$, updated local model $\boldsymbol{\theta}_k^{(t)} \leftarrow \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}_k^{(t)}$ and the auxiliary dataset $\mathcal{A}$; create coefficient matrices $\mathbf{A}^{(t)}, \mathbf{A}^{(t+1)} \in \mathbb{R}^{N\times N}$; initialize vector $\mathbf{u} \leftarrow \Delta\mathbf{b}^{(t)}/\eta$.

/* Compute coefficients for the least square problem */

for $n, j \in [N]$ **do**
  if $n \neq j$ **then**
    $\mathbf{A}_{n,j}^{(t)} \leftarrow -\mathcal{S}_{n,j}^{(t)}$, $\mathbf{A}_{n,j}^{(t+1)} \leftarrow -\mathcal{S}_{n,j}^{(t+1)}$
  else
    $\mathbf{A}_{j,j}^{(t)} \leftarrow \sum_{n\neq j} \mathcal{S}_{j,n}^{(t)}$, $\mathbf{A}_{j,j}^{(t+1)} \leftarrow \sum_{n\neq j} \mathcal{S}_{j,n}^{(t+1)}$
  end
end
if $m = 1$ **then**
  $\mathbf{z} \leftarrow$ **LeastSquare**$(\mathbf{A}^{(t)}, \mathbf{u})$ as described in (10)
  $\mathbf{N}^{(t)} \leftarrow \lfloor |\mathcal{B}| \cdot \mathbf{z} \rceil$
else
  $\bar{\mathbf{A}} \leftarrow (\mathbf{A}^{(t)} + \mathbf{A}^{(t+1)})/2$
  $\mathbf{z} \leftarrow$ **LeastSquare**$(\bar{\mathbf{A}}, \mathbf{u})$, $\mathbf{N}^{(t)} \leftarrow \lfloor |\mathcal{B}| \cdot \mathbf{z} \rceil$
  $\mathbf{N}^{(t)} \leftarrow$ **PosteriorSearch**$(\mathbf{N}^{(t)}, \Delta\mathbf{W}^{(t)}, \Delta\mathbf{b}^{(t)}, \boldsymbol{\mu}_n^{(t)}, \boldsymbol{\Sigma}_n^{(t)}, \boldsymbol{\mu}_n^{(t+1)}, \boldsymbol{\Sigma}_n^{(t+1)}, \eta)$
end
return $\mathbf{N}^{(t)} : \{N_j^{(t)}, \forall j \in [N]\}$

C.2. Posterior Search Algorithm

In Alg. 2, we simulated the dynamics of $\boldsymbol{\mu}_n^{(t)}$ during local training. First, the server estimates average embedded signal $\bar{\mathbf{e}}$ according to Eq. 14. Using the crude estimates of $N^{(t)}$ and $\mathcal{S}_{n,j}^{(t)}$ as inputs, the server computes $\Delta\mathbf{b}_j^{(t,\tau)}$ according to Eq. 12. Subsequently, the server computes $\Delta\boldsymbol{\mu}_{n,j}^{(t,\tau)}$ using $\bar{\mathbf{e}}$, and then estimates $\hat{\boldsymbol{\mu}}_{n,j}^{(t,\tau+1)}$. Recursively, repeating the procedure, the server finally obtains $\hat{\boldsymbol{\mu}}_{n,j}^{(t,\tau+1)}$. Based on the difference between $\hat{\boldsymbol{\mu}}_{n,j}^{(t,\tau+1)}$ and $\boldsymbol{\mu}_{n,j}^{(t,\tau+1)}$, the server can calibrate/improve estimated $N^{(t)}$.

Algorithm 2 Posterior Search

Input: Number of iterations T, $\Delta\mathbf{W}^{(t)}$, $\Delta\mathbf{b}^{(t)}$, learning rate η, estimated $\mathbf{N}^{(t)}$, $\boldsymbol{\mu}_n^{(t)}$, $\boldsymbol{\Sigma}_n^{(t)}$, $\boldsymbol{\mu}_n^{(t+1)}$ and $\boldsymbol{\Sigma}_n^{(t+1)}$ for $n, j \in [N]$.

Output: The number of samples with each label $N_j^{(t)}$ in the batches, where $\forall j \in [N]$.

Initial: The average embedding signal $\bar{\mathbf{e}} \in \mathbb{R}^L$.

/* Compute average embedding signal. */

for $l \in [L]$ **do**

  $\bar{\mathbf{e}}_l \leftarrow \Delta\mathbf{W}_{j,l}^{(t)} / \Delta\mathbf{b}_j^{(t)}$

end

for $n \in [N]$ **do**

  $\hat{\boldsymbol{\mu}}_n^{(t,1)} \leftarrow \boldsymbol{\mu}_n^{(t)}$

end

for $i \in [T]$ **do**

  $\mathbf{g} \leftarrow \lfloor \mathbf{N}^{(t)} / m \rceil$

  for $\tau \in [m]$ **do**

    /* Update the expectation */

    $\hat{\mathcal{S}}_{n,j}^{(t,\tau)} \leftarrow$ **MonteCarlo**$(\hat{\boldsymbol{\mu}}_n^{(t,\tau)}, \boldsymbol{\Sigma}_n^{(t)})$, for $n, j \in [N]$

    for $j \in [N]$ **do**

      /* Use the statics of model to estimate the updates of bias as illustrated in Eq.12. */

      $\Delta\mathbf{b}_j^{(t,\tau)} \leftarrow \frac{\eta}{|\mathcal{B}|} \left(\mathbf{g}_j \sum_{n \neq j}^{N} \hat{\mathcal{S}}_{j,n}^{(t,\tau)} - \sum_{n \neq j} \mathbf{g}_n \hat{\mathcal{S}}_{n,j}^{(t,\tau)} \right)$

    end

    for $j \in [N]$ **do**

      /* Update the intermediate means according to Eq.13. */

      $\Delta\boldsymbol{\mu}_{n,j}^{(t,\tau)} \leftarrow \Delta\mathbf{b}_j^{(t,\tau)} \cdot \sum_l^L \bar{\mathbf{e}}_l^2$, for $\forall n \in [N]$

      $\hat{\boldsymbol{\mu}}_{n,j}^{(t,\tau+1)} \leftarrow \hat{\boldsymbol{\mu}}_{n,j}^{(t,\tau)} + \Delta\boldsymbol{\mu}_{n,j}^{(t,\tau)}$, for $\forall n \in [N]$

    end

  end

  /* Compare the true statics and our proceedings estimation. */

  $\mathcal{M} \leftarrow \{j \in [N] \mid \sum_{n=1}^{N} \hat{\boldsymbol{\mu}}_{n,j}^{(T,m+1)} - \sum_{n=1}^{N} \boldsymbol{\mu}_{n,j}^{(T,m+1)} > 0\}$

  $j_{\max} \leftarrow \mathbf{argmax}_{j \in \mathcal{M}} \sum_{n=1}^{N} \hat{\boldsymbol{\mu}}_{n,j}^{(T,m+1)} - \sum_{n=1}^{N} \boldsymbol{\mu}_{n,j}^{(T,m+1)}$

  $\mathcal{I} \leftarrow \{j \in [N] \mid \sum_{n=1}^{N} \hat{\boldsymbol{\mu}}_{n,j}^{(T,m+1)} - \sum_{n=1}^{N} \boldsymbol{\mu}_{n,j}^{(T,m+1)} < 0\}$

  $j_{\min} \leftarrow \mathbf{argmin}_{j \in \mathcal{I}} \sum_{n=1}^{N} \hat{\boldsymbol{\mu}}_{n,j}^{(T,m+1)} - \sum_{n=1}^{N} \boldsymbol{\mu}_{n,j}^{(T,m+1)}$

  $N_{j_{\max}}^{(t)} \leftarrow N_{j_{\max}}^{(t)} - m$

  $N_{j_{\min}}^{(t)} \leftarrow N_{j_{\min}}^{(t)} + m$

end

return $\mathbf{N}^{(t)} : \{N_j^{(t)}, \forall j \in [N]\}$