Inferring Class-Label Distribution in Federated Learning

Raksha Ramakrishna
rakshar@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

György Dán
gyuri@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

Federated Learning (FL) has become a popular distributed learning method for training classifiers by using data that are private to individual clients. The clients′ data are typically assumed to be confidential, but their heterogeneity and potential class-imbalance adversely impact the accuracy of the trained model. The class-imbalance may not be common knowledge or may even be confidential information itself. Thus, the inference of the class-label distribution of the training data is important both from a performance and from a privacy perspective. In this paper, we study the problem of class-label distribution inference from an adversarial perspective, based on model parameter updates sent to the parameter server. Firstly, we present conditions under which exact inference is possible. We then introduce four new methods to estimate class-label distribution in the general FL setting. We evaluate the proposed inference methods on four different datasets and our results show that they significantly outperform state of the art methods.

CCS CONCEPTS

• **Security and privacy** → *Privacy-preserving protocols*; • **Computing methodologies** → **Supervised learning by classification**; Cost-sensitive learning; *Classification and regression trees*; *Multi-agent systems*;

KEYWORDS

Federated Learning, Class-imbalance, privacy leakage, class-label distribution

ACM Reference Format:
Raksha Ramakrishna and György Dán. 2022. Inferring Class-Label Distribution in Federated Learning. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security (AISec '22), November 11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3560830.3563725

1 INTRODUCTION

Federated learning (FL) is a form of distributed machine learning (ML) where during the training process, an aggregator or server collects model parameter or gradient updates from clients, computes an aggregated value and sends the updated model parameters to the clients [13]. This process is carried out until convergence, i.e., until the value of the model parameters does not change. In the end, a trained ML model is available for use-both to the server and to the clients.

In FL, the training data used to update model parameters remains local and private to each of the clients, as is common in distributed learning. However, this also means that the server may not be aware of potential class-imbalance, i.e., clients having vastly different composition of training data [2], which is prevalent in classification tasks. The fact that the server has no knowledge of the class imbalance can be approached from at least two perspectives. On the one hand, from a performance stand-point, class-imbalance is known to negatively affect the classification accuracy of the trained model, therefore detecting and mitigating it is important for training accurate models using FL. On the other hand, from the adversarial perspective, the composition of the clients' training data could be *sensitive* information that the clients do not want to reveal. Applications with class-imbalance and where the label distribution may be sensitive include fraud detection, claim prediction, default prediction, churn prediction, spam detection, anomaly detection, outlier detection, intrusion detection, and conversion prediction [11].

In both cases, the fundamental question that arises is how well the server can estimate the class-label distribution of individual clients. In this paper, we look at this problem from an adversarial perspective. The server plays the role of the adversary that tries to infer the class-label distribution of each client given the model parameter updates that the clients share with the server. We firstly discuss the case when exact inference is possible under certain conditions and then propose estimators that perform well on average in a general setting. Our numerical results show that the proposed methods of inference outperform state-of-art methods of label composition estimation [17]. We also discuss a mitigation measure that the client can utilize so that the adversary cannot infer the class-label distribution effectively. Finally, we illustrate how the knowledge of class-label distribution can be used to improve model accuracy in FL.

The rest of the paper is organized as follows. In Section 2 we review related work. In Section 3, we describe the problem setting of class-label distribution inference in a federated learning scenario and from the perspective of an adversary. Then, in Section 4, we study the attack-i.e. class-label distribution inference for a neural network classifier. Then, we analyze how the inference would change when the conditions required for exact inference are not satisfied. On this basis, in Section 5, we propose four estimators for class-label distribution inference based on different approximations. In Section 6, we provide empirical evidence of the superiority of the proposed methods of inference in comparison to state of the art. Furthermore, we also discuss a simple countermeasure to the privacy attack and demonstrate how class-label inference could be

AISec '22, November 11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9880-0/22/11.
https://doi.org/10.1145/3560830.3563725

used to improve accuracy of the trained model. Lastly, conclusions are drawn and future work is outlined.

2 RELATED WORK

Works in the literature either use class-label distribution inference to address class-imbalance in federated learning or they treat the inference as a form of property inference attack. There are many works that address class-imbalance in federated learning by either making changes to the loss function [3, 17] or by efficiently clustering or selecting clients at every iteration so that the model parameters are updated by more homogeneous data distributions [5, 6, 12, 18]. For both these approaches, class-label distribution inference is used as an intermediate step.

In [17], a new loss function called the ratio-loss is introduced in order to combat class-imbalance in federated learning. The ratio-loss function utilizes an estimate of the global data composition, i.e., class-label distribution of all the training data from the clients. In [18], an online learning framework to address class-imbalance is introduced. The framework is based on a client selection algorithm, which takes as input an estimate of the class-label distribution of all the clients. Similarily, in [12] authors assume that the class-label distribution is available to the server and this information is used to group clients by minimizing Earth mover's distance (EMD) between clients in a group and the global class-label distribution. In [5], a self-balancing scheme is designed to address class imbalance by selecting clients using a mediator. The mediator selects clients so that the KL divergence between the overall class-distributions of clients and the uniform distribution is minimized.

A different line of works treat class-label distribution inference as a property inference attack where the adversary is interested to find out private information about the clients. For example, preference profiling attacks in federated learning [21] against clients tend to reveal user preferences for a certain majority or minority class of data. The attack in [21] is carried out by training meta-classifiers [1] to extract training data information such as class-label proportion and the degree of class-imbalance using gradients received from the clients.

Additionally, the gradients from clients in federated learning are particularly vulnerable to attacks. It has been shown in [22] that it is possible to reconstruct both training data and their respective labels from the gradients alone. Other variations on the method above [8, 14, 20] also make a strong case for mitigating gradient leakage in federated learning systems. However, these methods only work when clients use small batches of training data to produce gradients. In a more realistic setting, the batch sizes are larger thereby rendering the aforementioned methods ineffective in practice.

More directly related to the methodology of class-label distribution inference presented in this paper is [16], where client-level labels are leaked in a federated learning setup by using gradients from the last layer. In [16], the sum of the gradient vectors of the weights connected to the output that corresponds to a certain class-label is used to extract label proportion information. However, the method in [16] has the drawback that as the model converges, the attack success rate decreases whereas in our proposed methods, the estimation accuracy increases as the iterations progress.

Compared to previous work, our main contributions can be summarized as follows:

- To the best of our knowledge, this paper is the first to propose an exact inference method for class-label distribution for any type of neural network classifier with a few assumptions on the initialization of parameters.
- For cases where the assumption for exact inference are not satisfied, we propose four estimators to infer class-label distribution in a federated learning system.
- We show that the proposed estimation methods outperform the state of the art label proportion estimation method in [17], and discuss the reasons for their superior performance.
- Focusing on class-label distribution inference in the adversarial setting, we show that random oversampling is a very effective countermeasure that renders the attacks ineffective.

3 PROBLEM FORMULATION

We consider a classification problem in a federated learning (FL) scenario. The task of the classifier given an input or feature $\boldsymbol{x} \in \mathbb{R}^D$ is to identify the most suitable class for the input amongst C classes. Let the output of the classifier be a vector of length C denoted by $\boldsymbol{y}$, where each entry $[\boldsymbol{y}]_c$ is the probability that the input belongs to a particular class c, $c = 1, 2, \dots C$, i.e., $\sum_c [\boldsymbol{y}]_c = 1$. Therefore, the classifier output $\boldsymbol{y}$ lies in the probability simplex Δ^{C-1}, i.e., $\boldsymbol{y} \in \Delta^{C-1}$, also called the the $C-1$ unit simplex. The classifier is modeled as a function $\boldsymbol{f} : \mathbb{R}^D \rightarrow \Delta^{C-1}$ parameterized by vector $\boldsymbol{\theta}$. Then, the output of the classifier is $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$. The classifier $\boldsymbol{f}$ described above is trained in a federated manner. The clients are indexed by $k \in \{1, 2, \dots, K\}$. The training data for client k consists of N_k training samples and is denoted by the set $\mathcal{D}_k \triangleq \{\boldsymbol{x}_i^k, \boldsymbol{e}_i^k\}_{i=1,2,\dots N_k}$, where $\boldsymbol{x}_i^k$ is the feature or input vector and $\boldsymbol{e}_i^k \in \{0, 1\}^C$ is the class-label that is a one-hot vector so that the entry corresponding to class c is 1 and the rest are zero. Let us denote by $N_{c,k}$ the number of training samples in dataset $\mathcal{D}_k$ of client k with class-label c. Then, the class-label distribution of the training data of client k is defined as

$$\boldsymbol{p}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \boldsymbol{e}_i^k \in \Delta^{C-1}, \textit{i.e.,}\ [\boldsymbol{p}_k]_c \triangleq \frac{N_{c,k}}{\sum_{c=1}^{C} N_{c,k}}, \tag{1}$$

As usual in the case of classification, the loss function for training is assumed to be the cross entropy for each client in the federated learning setup. Therefore, the optimization problem is

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \text{ where } \mathcal{L}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \mu_k \mathcal{L}_k(\boldsymbol{\theta}), \tag{2}$$

$$\text{and } \mathcal{L}_k(\boldsymbol{\theta}) = -\frac{1}{N_k} \sum_{i=1}^{N_k} \sum_{c=1}^{C} [\boldsymbol{e}_i^k]_c \log\left(\left[\boldsymbol{f}(\boldsymbol{x}_i^k; \boldsymbol{\theta})\right]_c\right) \tag{3}$$

Here $\mu_k > 0, \sum_k \mu_k = 1$. There are many approaches to solving the problem in (2) [13] [19]. Broadly, at communication round $t + 1$, each client obtains the classifier parameter $\boldsymbol{\theta}^t$ from the server, and locally updates the classifier parameter using its training data. We denote by $\boldsymbol{\theta}_k^{t+1}$ the locally updated parameter. Then, the one step update of client k in round $t + 1$ is

$$\boldsymbol{\theta}_k^{t+1} \leftarrow \boldsymbol{\theta}_k^t - \eta \nabla \mathcal{L}_k(\boldsymbol{\theta}^t). \tag{4}$$

The above local update can be repeated a number of times, called epochs E, each time on small batches of training data of size B from $\mathcal{D}_k$. The process in (4) is called stochastic gradient descent (SGD). The server aggregates the local classifier parameter estimates to form the global parameter estimate in round $t+1$ as

$$\boldsymbol{\theta}^{t+1} \leftarrow \sum_{k=1}^{K} \mu_k \boldsymbol{\theta}_k^{t+1} \tag{5}$$

The weights μ_k capture the importance of each client's update in the global parameter update. In the literature, the common choices are $\mu_k = \frac{1}{K}$ or $\mu_k = \frac{N_k}{\sum_k N_k}$.

3.1 Adversarial Model and Privacy

We consider an adversary that wants to infer the label distribution for client k, i.e., $\boldsymbol{p}_k$. The adversary has access to parameter updates sent from and to the FL server at different iterations t, that is, $\boldsymbol{\theta}^t$ and $\boldsymbol{\theta}_k^t$ $\forall t$. While this adversarial model may seem strong, the FL server is a third party in many industrial applications, which justifies the considered honest-but-curious attack model.

We quantify the (in)ability of the attacker to infer the class-label distribution by the deviation of the estimated label distribution from the true label distribution, analogous to the notion of privacy.

Definition 1 (Privacy Measure)**.** *The measure of privacy at client k is given by the mean-squared error (MSE) between the true and the estimated label distributions, $\boldsymbol{p}_k$ and $\hat{\boldsymbol{p}}_k$ respectively, i.e.*

$$MSE_k = \|\boldsymbol{p}_k - \hat{\boldsymbol{p}}_k\|_2^2 \tag{6}$$

The larger the privacy measure, the better hidden is the sensitive information $\boldsymbol{p}_k$. Hence, the adversary will try to minimize the MSE while the defender aims at maximizing it. The privacy measure is bounded since both the true and the estimated distributions lie in the probability simplex i.e. $\boldsymbol{p}_k, \hat{\boldsymbol{p}}_k \in \Delta^{C-1}$.

The goal of this paper is to study label distribution inference attacks in general deep neural network based classifiers. Nonetheless, to gain intuition, we first discuss attacks on logistic regression and then attacks on binary classifiers since they are simpler. We then extend the attacks to non-binary classifiers.

3.2 Analytic Example: Federated Logistic Regression

Logistic regression is a binary classifier where the decision boundary is dictated by a sigmoid function whose argument is linear with respect to the feature vector $\boldsymbol{x}_i^k$. The parameters of the classifier are the weight vector $\boldsymbol{w}$ and the bias b. The posterior distribution of class c given feature vector $\boldsymbol{x}_i^k$ is a sigmoid function,

$$p(c=1|\boldsymbol{x}_i^k) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i^k + b) = \frac{1}{1+\exp\left\{-\boldsymbol{w}^\top \boldsymbol{x}_i^k + b\right\}}. \tag{7}$$

Thus, the log-likelihood ratio for a binary classifier is an affine function of parameter $\boldsymbol{w}$,

$$\ln\left(p(c=1|\boldsymbol{x}_i^k)/p(c=2|\boldsymbol{x}_i^k)\right) = \boldsymbol{w}^\top \boldsymbol{x}_i^k + b. \tag{8}$$

Consequently, classification can be done using the decision rule

$$y_i^k = \begin{cases} 1, & \boldsymbol{w}^\top \boldsymbol{x}_i^k + b \geq 0,\ c=1 \\ 0, & \boldsymbol{w}^\top \boldsymbol{x}_i^k + b < 0,\ c=2 \end{cases} \tag{9}$$

The parameters $\boldsymbol{w}, b$ are learnt jointly by the clients using training data $\mathcal{D}_k \triangleq \{\boldsymbol{x}_i^k, e_i^k\}$ with $e_i^k \in \{0,1\}$ in the FL setting via gradient descent.

3.2.1 Privacy attack. We drop the subscript k denoting clients since the attack is the same for every client. We summarize the privacy attack in the following proposition.

Proposition 1. *Assume that the adversary knows the learning rate η and has access to weight updates $\boldsymbol{w}^t, b^t$ from the client. Then, if the server starts with weight vector initialized to zero, i.e, $\boldsymbol{w}^0 = \boldsymbol{0}$, and bias b^0, the adversary's attack is given by*

$$[\hat{\boldsymbol{p}}]_2 = \frac{\sum_i e_i}{N} = \frac{\Delta b^1}{\eta} + \sigma(b^0) \tag{10}$$

where $\Delta b^1 = b^1 - b^0$ is the change in bias after the first iteration. This is an exact inference, i.e., $[\boldsymbol{p}]_2 = [\hat{\boldsymbol{p}}]_2$.

We provide the proof in the Appendix. From (10), the attacker can obtain an estimate of the proportion of positive labels, i.e., $e_i = 1$ if it has access to the bias update Δb^1 and the step η. Importantly, we can extend the attack in Proposition 1 to a multi-class setting where class-labels are coordinate vectors, $\boldsymbol{e}_i \in \{0,1\}^C$. The parameters to infer in the case of multi-class regression can be written as a matrix

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{w}_1 & \boldsymbol{w}_2 & \cdots & \boldsymbol{w}_C \end{bmatrix}, \tag{11}$$

and the bias vector $\boldsymbol{b}$. The classifier outputs class c for a feature vector $\boldsymbol{x}_i$ via a *softmax* function $\boldsymbol{\sigma}\left(\mathbf{W}^\top \boldsymbol{x}_i + \boldsymbol{b}\right)$ defined as

$$\left[\boldsymbol{\sigma}\left(\mathbf{W}^\top \boldsymbol{x}_i + \boldsymbol{b}\right)\right]_c = \frac{\exp\{\boldsymbol{w}_c^\top \boldsymbol{x}_i + b_c\}}{\sum_c \exp\{\boldsymbol{w}_c^\top \boldsymbol{x}_i + b_c\}}. \tag{12}$$

The class c is then decided as the class with maximum posterior probability,

$$c^* = \arg\max_c \left[\boldsymbol{\sigma}\left(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}\right)\right]_c. \tag{13}$$

We summarize the attack in the following corollary.

Corollary 1. *Consider federated multi-class logistic regression with labels $\boldsymbol{e}_i \in \{0,1\}^C$. Assume that η is known and the server starts with weight matrix $\mathbf{W}^0 = \boldsymbol{0}$. Then the label distribution can be inferred exactly as*

$$\hat{\boldsymbol{p}} = \frac{1}{\eta}\Delta \boldsymbol{b}^1 + \boldsymbol{\sigma}(\boldsymbol{b}_0). \tag{14}$$

where $\hat{\boldsymbol{p}} = \frac{1}{N}\sum_n \boldsymbol{e}_i$ and $\Delta \boldsymbol{b} = \boldsymbol{b}^1 - \boldsymbol{b}^0$ is the weight update at the first iteration ($t=1$).

In what follows, we show how to extend these attack strategies to generic neural network classifiers.

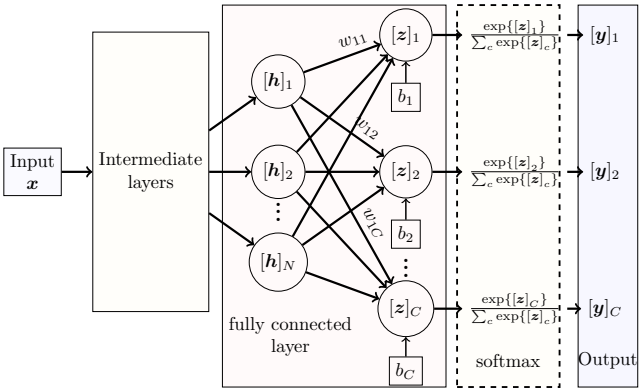


Figure 1: A generic neural network classifier with fully-connected output layer and softmax layer before the classification output. The weights and biases from the fully-connected output layer are used to estimate the class-label distribution in this paper.

4 CLASS-LABEL DISTRIBUTION INFERENCE FOR NEURAL NETWORK CLASSIFIERS

While there are several works that discuss privacy attacks on neural network architectures, existing approaches are mostly empirical, without analytical guarantees. In this section, we show that it is possible to construct attacks that are theoretically guaranteed to succeed under certain assumptions. We consider a generic neural network architecture with any type and number of intermediate layers, a fully-connected output layer followed by the softmax operation. One such example network is depicted in Fig.1. In what follows, we show that the parameters pertaining to the last (fully-connected output) layer are sufficient to infer the class-label distribution, under appropriate assumptions on the initialization of the connection weights between the fully-connected layer before the output and the output layer.

4.1 Exact Inference for Neural Network Classifiers

Let us denote by m_L the number of neurons in the fully-connected layer before the output. The number of neurons in the output layer is C, and thus the connection weight matrix between the fully-connected layer and the output layer for client k is a matrix $\mathbf{W}_k \in \mathbb{R}^{m_L \times C}$. Let the bias parameters for each of the C neurons in the output layer be $\boldsymbol{b}_k \in \mathbb{R}^C$, and denote by $\boldsymbol{h}_{i,k} \in \mathbb{R}^{m_L}$ the input to the fully-connected layer. Then the output of the fully-connected layer, $\boldsymbol{z}_i^k \in \mathbb{R}^C$ is a vector

$$\boldsymbol{z}_i^k \triangleq \mathbf{W}_k^\top \boldsymbol{h}_{i,k} + \boldsymbol{b}_k. \tag{15}$$

The softmax activation function, $\boldsymbol{\sigma} : \mathbb{R}^C \to \Delta^{C-1}$, defined as

$$[\boldsymbol{\sigma}(\boldsymbol{z})]_c = \frac{\exp\{[\boldsymbol{z}]_c\}}{\sum_c \exp\{[\boldsymbol{z}]_c\}}, \tag{16}$$

is applied to $\boldsymbol{z}_i^k$ to produce the output of the classifier,

$$\boldsymbol{y}_i^k = \boldsymbol{\sigma}\left(\boldsymbol{z}_i^k\right). \tag{17}$$

In what follows we omit the subscript k as the attack is the same for every client.

We assume that the loss function used for training is cross entropy and the learning rate is η. The class labels are one-hot vectors, $\boldsymbol{e}_i \in [0,1]^C$, there are C neurons in the output and the activation function is softmax like in (16). For such classifiers, we can formulate a privacy attack as described in the following proposition.

Proposition 2. *Consider that the loss function used for training is the cross entropy, and assume that the learning rate η is known. If the number of local epochs $E = 1$, full-batch gradient descent is undertaken, if the weight matrix of the last layer is initialized to $\mathbf{W}^0 = \mathbf{0}$ and the bias vector is initialized as $\boldsymbol{b}^0$, then the class-label distribution $\hat{\boldsymbol{p}}$ is*

$$\hat{\boldsymbol{p}} = \boldsymbol{p} = \frac{1}{N}\sum_i \boldsymbol{e}_i = \frac{1}{\eta}\Delta\boldsymbol{b} + \boldsymbol{\sigma}(\boldsymbol{b}^0) \tag{18}$$

where $\Delta\boldsymbol{b} = \boldsymbol{b}^1 - \boldsymbol{b}^0$ is the bias update at the first iteration ($t = 1$).

The proof can be found in the Appendix. Proposition 2 can be applied in the federated learning setting, if the server provides an initialization of all zeros in the last layer, $\mathbf{W}^0 = \mathbf{0}$ and demands a parameter update after $E = 1$ local epoch from every client as in (4). Then the bias update $\boldsymbol{b}^1$ from each client after a full-batch gradient descent ($B = N$) can be used to compute the label distribution $\hat{\boldsymbol{p}}$ exactly.

If `FedSGD` is used for optimization, the clients are sending gradient updates i.e. $E = 1, B = N$ naturally. This lends itself perfectly to the attack condition when the weights initialized to zero, $\mathbf{W}^0 = \mathbf{0}$. However, if `FedAvg` is used, then the conditions of attack are difficult to satisfy owing to epoch and batch size used by the clients. We discuss this next.

4.2 Non-exact Inference for Neural Network Classifiers

In this subsection, we discuss the attack when the conditions for exact inference are not satisfied. We discuss what happens to class-label distribution inference when the clients undertake a generic local update. More specifically, the general conditions are when the weights are not initialized to zero, i.e., $\mathbf{W}^0 \neq \mathbf{0}$, when the clients do not use full-batch or use stochastic gradient descent with batch size $B < N$ and when the number E of local epochs at an client is greater than 1. We use the gained insight for designing estimators based on different approximations in Section 5.

Let us consider now a generic local update that is initialized by non-zero global weights $\mathbf{W}^t$ at iteration t. Let the batch size $B < N$ and local epochs be $E > 1$. Also, let $j = 1, 2, \ldots J$ index the batch number and $\mathcal{N}_j$ be the set of data samples in batch j. Hence, the total number of mini-batches per epoch is $J \triangleq \lfloor N/B \rfloor$ times where $\lfloor . \rfloor$ is the `floor` operator. The bias update is then

$$\boldsymbol{b}_k^{t+1} = \boldsymbol{b}^t - \frac{\eta}{N_k}\sum_{\ell=1}^{E}\sum_{j=1}^{J}\sum_{i\in\mathcal{N}_j}\boldsymbol{\sigma}(\boldsymbol{z}_i^{k,j,\ell}) + \frac{\eta E}{N_k}\sum_i \boldsymbol{e}_i, \tag{19}$$

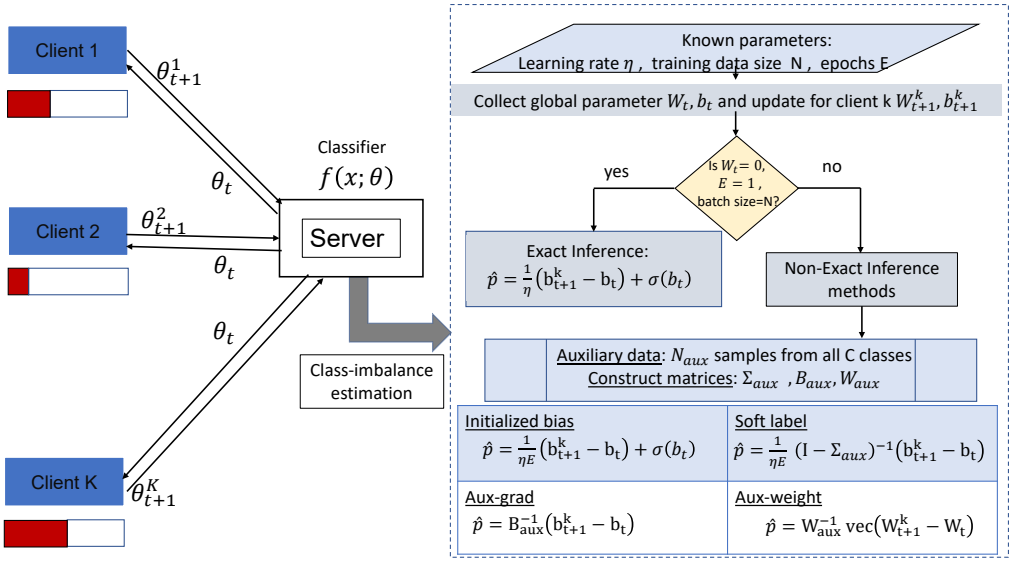


Figure 2: Block diagram of the proposed class-label distribution inference attacks.

where

$$z_i^{k,j,\ell} = (\mathbf{W}_k^{j,\ell})^\top \boldsymbol{h}_i + \boldsymbol{b}_k^{j,\ell} \tag{20}$$

$$\mathbf{W}_k^{j=1,\ell=1} \leftarrow \mathbf{W}^t,\ \mathbf{W}_k^{t+1} = \mathbf{W}_k^{j=J,\ell=E} \tag{21}$$

$$\boldsymbol{b}_k^{j=1,\ell=1} \leftarrow \boldsymbol{b}^t,\ \boldsymbol{b}_k^{t+1} = \boldsymbol{b}_k^{J,E}. \tag{22}$$

Now, the class-label distribution can be written as

$$\frac{1}{N_k}\sum_i \boldsymbol{e}_i^k = \frac{(\boldsymbol{b}_k^{t+1} - \boldsymbol{b}^t)}{\eta E} + \frac{1}{E}\sum_{\ell=1}^{E}\sum_{j=1}^{J}\sum_{i\in\mathcal{N}_j}\boldsymbol{\sigma}(z_i^{k,j,\ell}) \tag{23}$$

From (23), it is clear that the adversary has access to 1) initialized parameters (global parameters at iteration t) $\mathbf{W}^t, \boldsymbol{b}^t$ 2) updated parameters $\mathbf{W}_k^{t+1}, \boldsymbol{b}_k^{t+1}$. Furthermore, if the adversary knows the number of epochs E, then it can calculate the first term in (23). It needs to find, however, an approximation for the second term to obtain an estimator for class distribution inference. We propose four estimators in the next section.

5 ESTIMATORS FOR CLASS-LABEL DISTRIBUTION

In this section we propose four estimators for class-label distribution inference. One of them is based on Proposition 2. The other three utilize auxiliary datasets $\mathcal{D}_{\text{aux}}$ containing N_{aux} samples from each of the C classes. Auxiliary data that resembles clients' data could be obtained from the public domain or could be generated synthetically. One could also consider a case where a client is colluding with the server and offers its training data to serve as auxiliary data.

5.1 Inference using initialized bias

In this method, the second term in (23) is approximated by the softmax output of the initialized bias $\boldsymbol{b}^t$. In other words,

$$z_i^{j,\ell} = (\mathbf{W}_k^{j,\ell})^\top \boldsymbol{h}_i + \boldsymbol{b}_k^{j,\ell} \approx \boldsymbol{b}^t\ \forall i, j, \ell, k \tag{24}$$

The approximation holds with high accuracy if 1) the product $(\mathbf{W}_k^{j,\ell})^\top \boldsymbol{h}_i$ is small or close to zero and 2) if the bias is not updated much in subsequent iterations, $\boldsymbol{b}_k^{j,\ell} \approx \boldsymbol{b}^t$. The estimator for the class-label distribution of client k at global iteration t is then

$$\hat{\boldsymbol{p}}_{k,t}^{\text{init}} = \frac{(\boldsymbol{b}_k^{t+1} - \boldsymbol{b}^t)}{\eta E} + \boldsymbol{\sigma}(\boldsymbol{b}^t). \tag{25}$$

Furthermore, if it is known that the client is using all of its dataset to update at every global iteration, another more robust estimator of the class-label distribution is the average over all T observed global iterations,

$$\hat{\boldsymbol{p}}_k^{\text{init}} = \frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{p}}_{k,t}^{\text{init}}. \tag{26}$$

To further improve the estimate, the attacker could wait until the approximation holds with better accuracy, i.e., for large values of t (say T), and use $\hat{\boldsymbol{p}}_k^{\text{init}} = \hat{\boldsymbol{p}}_{k,T}^{\text{init}}$.

5.2 Inference using approximated softmax output

Using this method, we approximate the second term in (23) using an auxiliary dataset for every class c, $\mathcal{D}_{\text{aux}}^c \triangleq \{\boldsymbol{x}_i^c, \boldsymbol{e}_i^c\}_{i=1,\ldots,N_{\text{aux}}}$, which contains N_{aux} samples from class c to compute the output of the client for every epoch ℓ . At epoch ℓ, we assume that

$$\sum_{j=1}^{J}\sum_{i\in\mathcal{N}_j}\boldsymbol{\sigma}(z_i^{k,j,\ell}) \approx \sum_{c=1}^{C}[\boldsymbol{p}]_c\,\boldsymbol{\sigma}(z_{c,\text{aux}}^{\ell}), \tag{27}$$

where $\boldsymbol{\sigma}(z^{\ell}_{c,\text{aux}})$ is the average softmax output for auxiliary data belonging to class c when fed singly to update the model parameters at epoch ℓ i.e.,

$$\boldsymbol{\sigma}(z^{\ell}_{c,\text{aux}}) = \frac{1}{N_{\text{aux}}} \sum_{i=1}^{N_{\text{aux}}} \boldsymbol{\sigma}((\mathbf{W}^{\ell}_{\text{aux},c})^{\top} \boldsymbol{h}^{c}_{i} + \boldsymbol{b}^{\ell}_{\text{aux},c}) \tag{28}$$

$$\mathbf{W}^{\ell+1}_{\text{aux},c} \leftarrow \mathbf{W}^{\ell}_{\text{aux},c} - \eta \nabla_{\mathbf{W}^{\ell}_{\text{aux},c}} \mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t) \tag{29}$$

$$\mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t) = -\frac{1}{N_{\text{aux}}} \sum_{i=1}^{N_{\text{aux}}} [\boldsymbol{e}^{c}_{i}]_c \log\left(\left[f(\boldsymbol{x}^{c}_{i}; \boldsymbol{\theta}_t) \right]_c \right). \tag{30}$$

Therefore, the approximation considered is

$$\frac{1}{E} \sum_{\ell=1}^{E} \sum_{j=1}^{J} \sum_{i \in \mathcal{N}_j} \boldsymbol{\sigma}(z^{k,j,\ell}_{i}) \approx \frac{1}{E} \sum_{c=1}^{C} [\boldsymbol{p}]_c \sum_{\ell=1}^{E} \boldsymbol{\sigma}(z^{\ell}_{c,\text{aux}}). \tag{31}$$

Let $\boldsymbol{\sigma}(z_{c,\text{aux}}) \triangleq \frac{1}{E} \sum_{\ell=1}^{E} \boldsymbol{\sigma}(z^{\ell}_{c,\text{aux}})$. We can then rewrite (23) as

$$\boldsymbol{p}_k \approx (1/\eta E)(\boldsymbol{b}^{t+1}_{k} - \boldsymbol{b}^{t}) + \sum_{c=1}^{C} [\boldsymbol{p}_k]_c \boldsymbol{\sigma}(z_{c,\text{aux}}) \tag{32}$$

$$(\mathbb{I} - \Sigma_{\text{aux}})\, \boldsymbol{p}_k \approx \frac{(\boldsymbol{b}^{t+1}_{k} - \boldsymbol{b}^{t})}{\eta E}, \tag{33}$$

where $\Sigma_{\text{aux}} \triangleq \left[\boldsymbol{\sigma}(z_{1,\text{aux}}) \quad \boldsymbol{\sigma}(z_{2,\text{aux}}) \quad \dots \quad \boldsymbol{\sigma}(z_{C,\text{aux}}) \right]$. Thus, we propose the estimator

$$\hat{\boldsymbol{p}}^{\text{soft-label}}_{k,t} = \frac{1}{\eta E} \left(\mathbb{I} - \Sigma_{\text{aux}} \right)^{-1} (\boldsymbol{b}^{t+1}_{k} - \boldsymbol{b}^{t}). \tag{34}$$

Note 1. If the matrix $(\mathbb{I} - \Sigma_{\text{aux}})$ is not invertible, then one can use a low-rank approximation of the matrix obtained via singular-value decomposition.

5.3 Inference using approximated gradient

In this method, we use the same type of auxiliary dataset per class, $\mathcal{D}^{c}_{\text{aux}}$ as in the last section. Instead of approximating the softmax output, we use the auxiliary dataset to compute an approximation of the sum of gradients in local epochs for a client. We make the assumption that the sum of gradients that contribute to change in the bias are composed of proportional changes in bias when the model is updated singly with data from class c. More specifically, we approximate the bias update as

$$\frac{(\boldsymbol{b}^{t+1}_{k} - \boldsymbol{b}^{t})}{\eta E} \approx \sum_{c=1}^{C} [\boldsymbol{p}]_c \nabla_{\boldsymbol{b}_{\text{aux}}} \mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t). \tag{35}$$

Furthermore, we approximate the gradient $\nabla_{\boldsymbol{b}_{\text{aux}}} \mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t)$ using the bias updates when auxiliary data from class c is used to update the model parameters,

$$\nabla_{\boldsymbol{b}_{\text{aux}}} \mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t) \approx \frac{\boldsymbol{b}^{E}_{\text{aux},c} - \boldsymbol{b}^{t}}{\eta E}, \tag{36}$$

where $\boldsymbol{b}^{E}_{\text{aux},c}$ is the bias obtained by updating the model E epochs initialized at $\boldsymbol{\theta}_t$. The bias update for each epoch is similar to (29). This leads to the estimator

$$\hat{\boldsymbol{p}}^{\text{grad}}_{k,t} = \mathbf{B}^{-1}_{\text{aux}} (\boldsymbol{b}^{t+1}_{k} - \boldsymbol{b}^{t}), \tag{37}$$

$$\text{where } \mathbf{B}_{\text{aux}} = \left[\boldsymbol{b}^{E}_{\text{aux},1} - \boldsymbol{b}^{t} \quad \boldsymbol{b}^{E}_{\text{aux},2} - \boldsymbol{b}^{t} \quad \cdots \quad \boldsymbol{b}^{E}_{\text{aux},C} - \boldsymbol{b}^{t} \right]. \tag{38}$$

Corollary 2. *When $E = 1, B = N_k$, the weight matrix of the last layer is initialized to $\mathbf{W}^0 = \mathbf{0}$ and the bias vector is initialized as $\boldsymbol{b}^0$, then the estimators at iteration $t = 1$, $\hat{\boldsymbol{p}}^{init}_{k,1}, \hat{\boldsymbol{p}}^{soft\text{-}label}_{k,1}$ $\hat{\boldsymbol{p}}^{grad}_{k,1}$ are exact i.e. $\hat{\boldsymbol{p}}^{init}_{k,1} = \hat{\boldsymbol{p}}^{soft\text{-}label}_{k,1} = \hat{\boldsymbol{p}}^{grad}_{k,1} = \boldsymbol{p}$.*

$$\hat{\boldsymbol{p}}^{init}_{k,1} = \hat{\boldsymbol{p}}^{soft\text{-}label}_{k,1} = \hat{\boldsymbol{p}}^{grad}_{k,1} = \frac{1}{\eta}(\boldsymbol{b}^{1}_{k} - \boldsymbol{b}^{0}) + \boldsymbol{\sigma}(\boldsymbol{b}^{0}) = \boldsymbol{p} \tag{39}$$

We provide the proof of the above corollary in the Appendix.

5.4 Inference using weights from the last layer

Similar to the method described above, we make the assumption that the sum of gradients pertaining to the weight matrix in the last layer that contribute to change in the weight matrix of the last layer are composed of proportional changes in weight matrix when the model is updated singly with data from class c. Like the previously described methods, same type of auxiliary dataset per class, $\mathcal{D}^{c}_{\text{aux}}$ is used for this method. We use the following approximation,

$$\frac{1}{\eta E}(\boldsymbol{W}^{t+1}_{k} - \boldsymbol{W}^{t}) \approx \sum_{c=1}^{C} [\boldsymbol{p}]_c \nabla_{\boldsymbol{W}_{\text{aux}}} \mathcal{L}_{c,\text{aux}}(\boldsymbol{\theta}_t). \tag{40}$$

Thus, we propose the estimator

$$\hat{\boldsymbol{p}}^{\text{wt-grad}}_{k,t} = \mathbf{W}^{-1}_{\text{aux}} \text{vec}(\boldsymbol{W}^{t+1}_{k} - \boldsymbol{W}^{t}) \tag{41}$$

$$\mathbf{W}_{\text{aux}} = \left[\text{vec}(\boldsymbol{W}^{E}_{\text{aux},1} - \boldsymbol{W}^{t}) \quad \cdots \quad \text{vec}(\boldsymbol{W}^{E}_{\text{aux},C} - \boldsymbol{W}^{t}) \right]. \tag{42}$$

Note 2. The proposed estimators are not guaranteed to produce an estimate that lies in the probability simplex Δ^{C-1}. Therefore, the attacker can add a step of projecting the obtained estimate onto the probability simplex,

$$\hat{\boldsymbol{p}} \leftarrow \underset{\boldsymbol{p} \in \Delta^{C-1}}{\arg\min} \|\boldsymbol{p} - \hat{\boldsymbol{p}}\|_2^2 \tag{43}$$

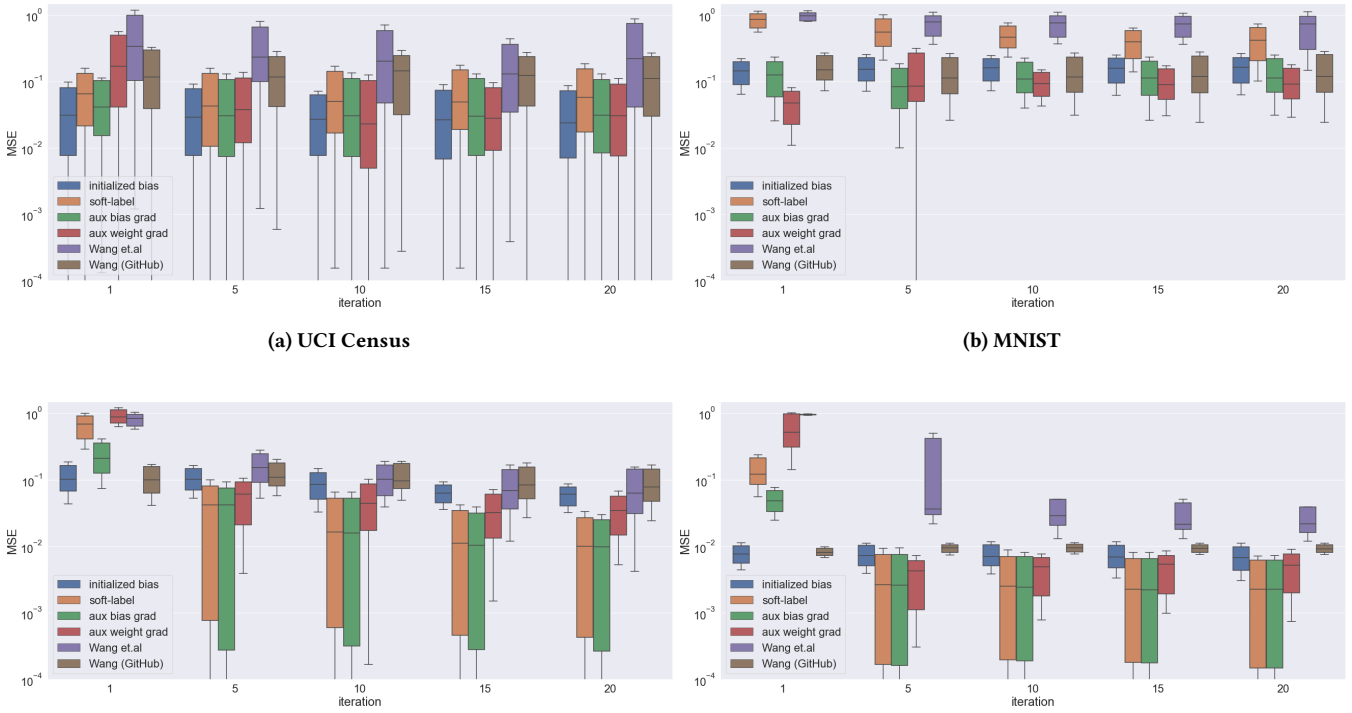
A pseudo-code to solve the problem above is provided in [7] and is employed in the numerical results.

6 NUMERICAL RESULTS

We evaluated the performance of the proposed estimators on four datasets: UCI Census Income dataset [4], MNIST [10], CIFAR-10 and CIFAR-100 [9]. An open-source implementation is also provided[1] For the UCI Census Income dataset the task is income classification (above or below 50k USD), which is a binary classification problem. The MNIST and CIFAR-10 datasets are used for multiclass $C = 10$ classification tasks where the task in MNIST is digit classification, and in CIFAR-10 it is image classification. Finally, the CIFAR-100 is a 100 class image classification task. The architectural details of the classifiers are given in the Appendix. All of them have a last softmax layer that is needed for classification. The number of clients simulated for each case is $K = 10$ and each client has $N_k \in [100, 3000]$ training samples drawn uniformly and the clients' training sets are disjoint, i.e., $\mathcal{D}_k \cap \mathcal{D}_j = \emptyset \; \forall k \neq j$.

To simulate a variety of class-label distributions, for each client k, the class-label distribution $\boldsymbol{p}_k$ is sampled uniformly from the probability simplex Δ^{C-1}. An algorithm [15] to do so involves

[1] https://github.com/raksha-ramakrishna/Inferring-Class-Label-Distribution-in-Federated-Learning



(a) UCI Census **(b) MNIST**

(c) CIFAR-10 **(d) CIFAR-100**

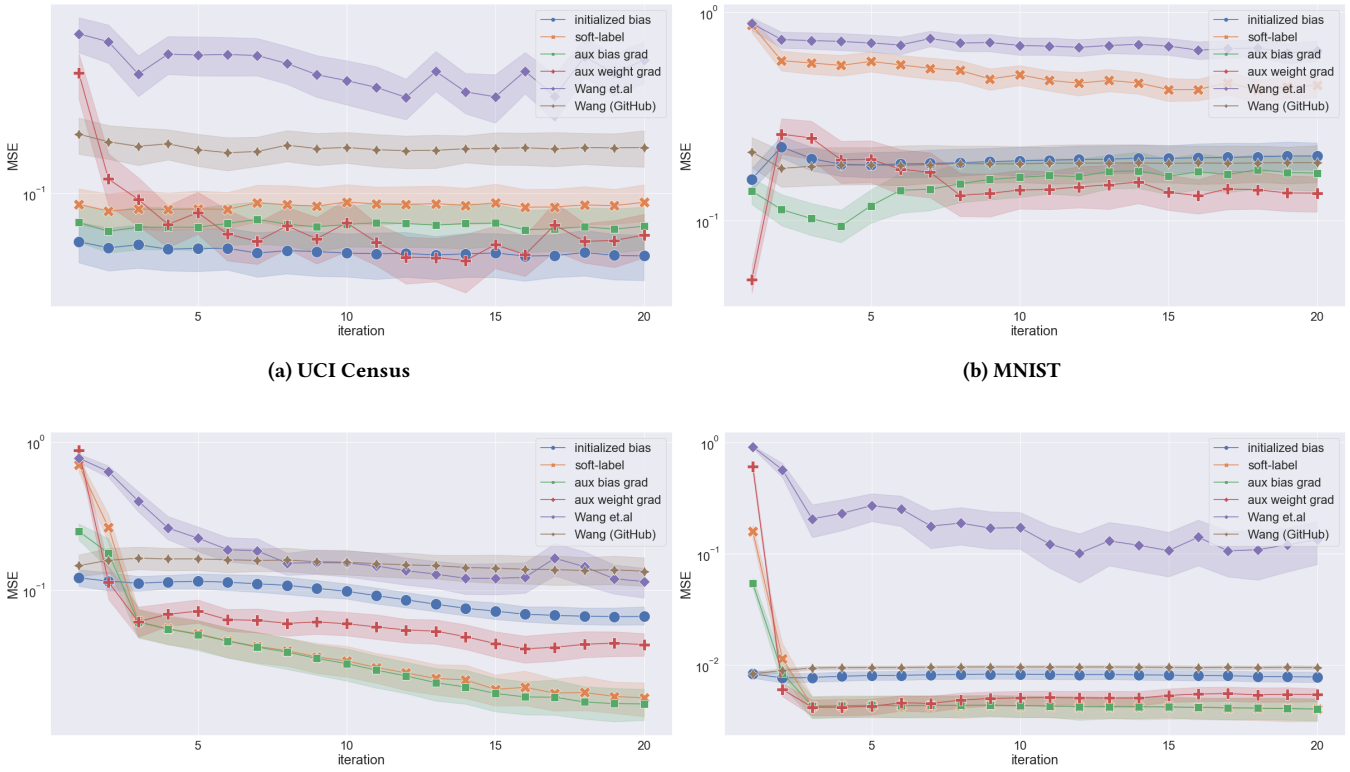
Figure 3: Boxplot of MSE of all the proposed estimators on 4 data sets. Labels 'initialized bias' refer to estimator in 5.1, 'soft label' to estimator in 5.2, 'aux bias grad' to 5.3 and 'aux weight grad' to 5.4. Labels 'Wang et.al' and 'Wang (GitHub)' refer to methods used as baselines for comparison.

sampling $C-1$ points uniformly at random from the interval $[0,1]$, sort them in ascending order by padding with 0 and 1 on either side. The consecutive difference between the points yields values $[\boldsymbol{p}_k]_c$.

The same set of clients are used for global parameter update at every iteration. The total number of global parameter update iterations is $T = 20$. For all the datasets, the number of local epochs is $E = 5$ and the batch size is $B = 256$. The learning rate is $\eta = 0.01$. The adversary can undertake the class-label inference at any global iteration $t \leq T$. We simulate 10 different FL setup by simulating different class-label distributions for $K = 10$ clients. Therefore, overall, there are 100 different test samples per global iteration. The FL training procedure is run for $T = 20$ iterations and the average mean squared error (MSE) per iteration is used as the performance metric.

Table 1: Mean MSE per iteration of proposed estimators and baselines for comparison

Dataset	Iteration	Initialized bias	Soft-label	Aux-bias grad	Aux-weight grad	Wang et.al [17]	Wang(GitHub)
UCI Census	1	0.0966	0.1061	**0.0878**	0.3118	0.5712	0.1944
	10	**0.0454**	0.0885	0.0719	0.0773	0.4129	0.1521
	20	**0.0418**	0.0869	0.0711	0.0745	0.3913	0.1494
MNIST	1	0.1579	0.8679	0.1384	**0.0520**	0.8836	0.2133
	10	0.1932	0.5007	0.1612	**0.1401**	0.6923	0.1871
	20	0.2040	0.4461	0.1688	**0.1353**	0.6528	0.1897
CIFAR-10	1	**0.1210**	0.7031	0.2493	0.8836	0.7802	0.1462
	10	0.0980	0.0331	**0.0318**	0.0591	0.1526	0.1541
	20	0.0662	0.0186	**0.0169**	0.0427	0.1141	0.1338
CIFAR-100	1	**0.0083**	0.1592	0.0546	0.6073	0.9189	**0.0083**
	10	0.0083	**0.0043**	**0.0043**	0.0051	0.1719	0.0096
	20	0.0078	**0.0040**	**0.0040**	0.0055	0.1342	0.0095



(a) UCI Census **(b) MNIST**

(c) CIFAR-10 **(d) CIFAR-100**

Figure 4: Average MSE of all estimators with respect to global iteration t.

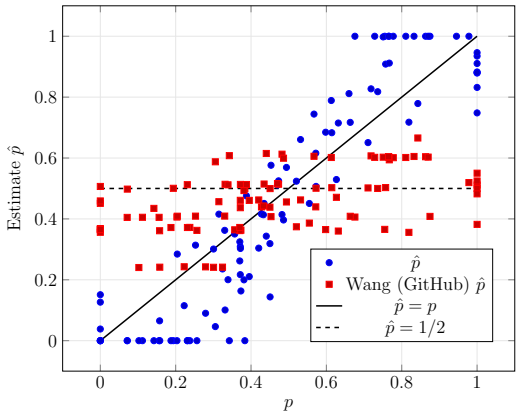


Figure 5: Actual and estimated class-label proportion for UCI Census income dataset at $t = 10$. Note that the proposed method (best for iteration $t = 10$) provides more accurate estimates compared to Wang (GitHub) which provides $\hat{p} = 1/C$ irrespective of the actual proportion.

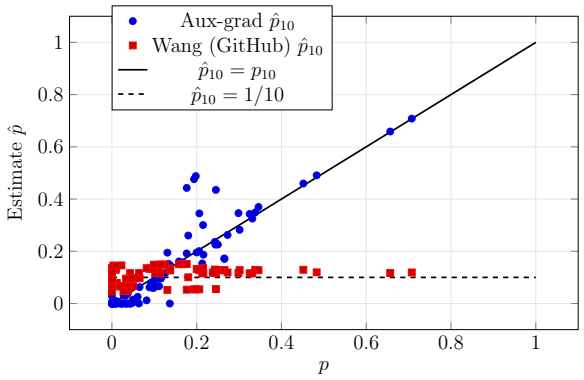
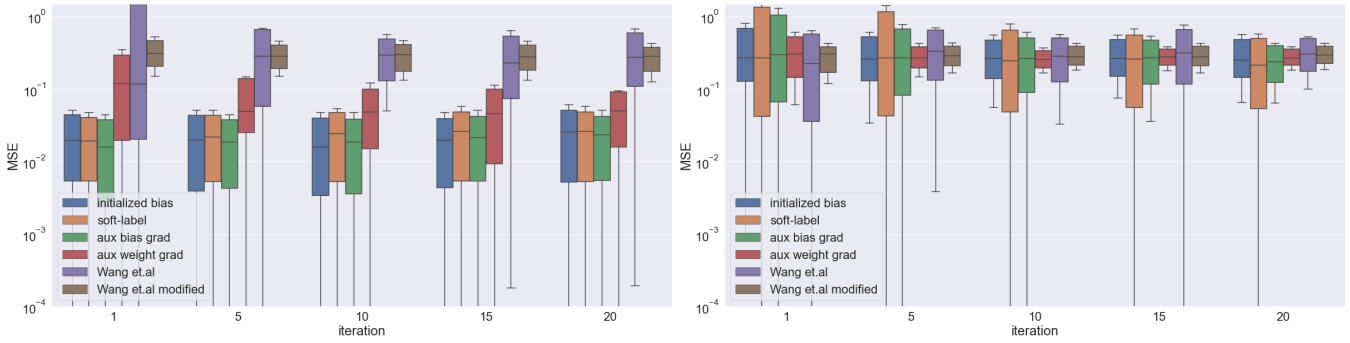


Figure 6: Actual and estimated class-label proportion for class 10 for CIFAR-10 dataset at $t = 10$. Note that the proposed method (best for iteration $t = 10$) provides more accurate estimates compared to Wang (GitHub) which provides $[\hat{p}]_{10} = 1/C$ irrespective of the actual proportion.

As baselines for comparison we use two methods. First, the method proposed in [17] for estimating the proportion of training data in each class for federated learning. Second, we consider the open source implementation[2] of the method described in [17],

[2]https://github.com/balanced-fl/Addressing-Class-Imbalance-FL



(a) Imbalanced datasets **(b) Balanced datasets via random oversampling**

Figure 7: Performance of the class-label estimation methods for highly imbalanced datasets before and after random oversampling is used as a countermeasure. US Census Income dataset is used here.

which includes an undocumented threshold step after ([17] eqn. (7)). We refer to the open-source implementation as "Wang (GitHub)".

6.1 Estimation performance

We start with comparing the performance of the proposed estimators to the baselines. Fig 3 shows the boxplot of the MSE for each of the estimators at global iterations $t \in \{1, 5, 10, 15, 20\}$. The figure shows that the median of the distribution of the MSE decreases with the global iterations, while the variance of the MSE is higher for the proposed methods that use an auxiliary dataset, which is to be expected due to the diversity of the auxiliary dataset. We can also see that the median of the distributions for the proposed methods is much lower than that for the method "Wang (GitHub)".

Fig 4 shows the average MSE (averaged over the number of test samples) as a function of the the global iteration t. The figure shows that the proposed estimators clearly outperform the scheme proposed in [17] and also perform better than "Wang (GitHub)", especially as the number of iterations t increases. Table 1 shows the average MSE for global iterations $t \in \{1, 10, 20\}$ for all the estimators and shows in bold the best estimator, i.e., the one with least average MSE. We see that it is one of our proposed methods that performs best for all datasets, outperforming [17] by an order of magnitude and significantly in the case of "Wang (GitHub)". The results from Table 1 also show that the longer the adversary waits (i.e., more global iterations), the better are the estimates of the class-label distribution obtained using the proposed estimators. This is because the approximations underlying the estimators are more accurate as the global parameters are closer to the local parameters. The method of "Wang et.al" follows a similar trend because the estimator is based on the same principle using auxiliary dataset. On the contrary, for "Wang (GitHub)", although there is a decrease in MSE with increasing global iterations, it is not significant.

To explain the good performance of the proposed estimators, note that the main intuition behind the approximations used for the estimators is that (i) the auxiliary data with class-label c would provide similar model updates as would client data with class-label c and that (ii) the overall model update by a client is proportional to the class-label distribution in the form of a linear combination of model updates given data from each class c. Note that the same approximated values computed via auxiliary data are used to infer the class-label distributions for all the clients. Hence, the difference between the model updates at different clients is assumed to stem solely from the difference between their class-label distributions. Although the estimators introduced in Section 5 in general do not have theoretical guarantees unless certain conditions hold, as in Corollary 2, they work well in practice and show that the intuition behind the proposed approximations is valid, i.e., if the auxiliary data provide the same input feature to class label mapping as the data at the clients, then the assumption about it providing similar model updates holds.

In other words, the more similar the auxiliary data and the client data are in feature space, the better the approximation. For example, if the auxiliary data for class c and client data with label class c are similar in feature space then the estimation performance is very good. With decreasing similarity, the estimators performance would also decrease.

We further examine the difference in performance between "Wang (GitHub)" and the best performing proposed method for two datasets: UCI Census Income and CIFAR-10. Fig. 5 shows the estimate of one of the classes with respect to the actual label proportion for that class after 10 global iterations. The figure shows that the estimates $\hat{p}$ obtained using "Wang (GitHub)" are generally close to $1/C = 0.5$ irrespective of the true value p. Whereas the estimates using the proposed method of initialized bias more closely match the true value. Similarly, in Fig 6 we show the estimate for class 10 after 10 global iterations. Even here, the estimate by "Wang (GitHub)" is close to $1/C = 0.1$ whereas the estimate using the proposed method of "aux-grad" lies closer to the true value. It is possible that the thresholding step in "Wang (GitHub)" results in class-label distribution estimates to be a uniform distribution over the number of classes. This makes their estimates particularly incorrect when the true distribution is highly imbalanced. Note that better estimates of class-label distribution are even more important from the perspective of correcting class-imbalance when the true distribution is skewed towards a few classes. Therefore, we conclude that not only do the proposed methods provide better estimates on average, their performance is excellent even when the distributions are highly skewed.

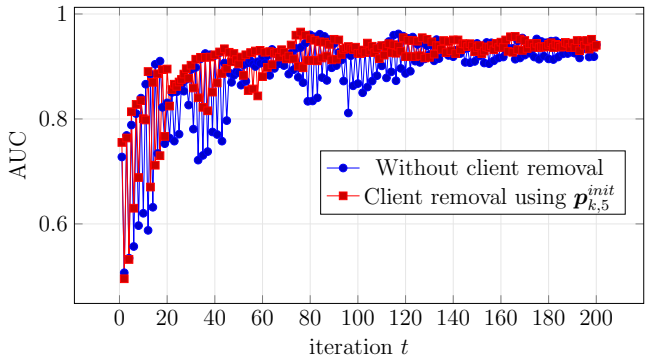


Figure 8: Area under the ROC curve (AUC) at iteration t for cases with and without client removal. US Census Income dataset used here.

6.2 The case of exact inference

When the conditions for exact inference that are outlined in Proposition 2 are satisfied, the class-label distribution estimate is exact. To verify this, we consider the same number of clients as before and same learning rate but to satisfy the conditions for exact inference, we consider one global iteration, $T = 1$, local epoch $E = 1$ and full-batch, $B = N$. Furthermore, we initialize the weight connection matrix to zero, $\boldsymbol{W}^0 = \mathbf{0}$. The mean MSE at global iteration $t = 1$ is given in Table 2. Note that among the proposed methods of inference, all except 'aux-weight grad' estimator give the correct solution barring numerical precision i.e. $\hat{\boldsymbol{p}}_{k,1}^{\text{init}} = \hat{\boldsymbol{p}}_{k,1}^{\text{soft-label}} = \hat{\boldsymbol{p}}_{k,1}^{\text{grad}} = \boldsymbol{p}$ in accordance with Corollary 2.

6.3 Random oversampling as a countermeasure

In this section, we evaluate random oversampling as a countermeasure to the proposed attacks. Random oversampling involves sampling with replacement from the class with least proportion until all the classes are equally represented in the training dataset. We perform this experiment on the UCI Census income dataset since it is easy to analyze a binary case. Before random oversampling, the class-label distributions are imbalanced, i.e., $p \in [0, 0.2] \cup [0.8, 1]$. Recall that the estimators we propose perform better than the baselines for such cases (c.f., Fig. 7a). Fig. 7b shows the results after random oversampling is used to make class-labels uniformly distributed ($p = 0.5$). The figure shows that all estimators (proposed and baselines) fail to estimate the original class-label distribution. Indeed, all estimators estimate $p = 0.5$, which is the class-label distribution after oversampling. We can thus conclude that random oversampling is indeed an effective countermeasure against all existing attacks, including the ones proposed in this paper. The results also indicate that one may need shadow datasets for making stronger attacks.

6.4 Addressing class imbalance in federated learning using class-label distribution estimate

It has been shown in prior works that when the clients in a federated learning system have imbalanced datasets, the overall convergence is slower and the accuracy is lower. To mitigate this problem, [12] proposed to group clients in a pairwise fashion based on their class-label distribution and select groups to send parameter updates at a certain global iteration. [12] showed empirically improved model accuracy as well as faster convergence. However, authors in [12] assume that the class-label distribution is known beforehand.

In a similar vein, we perform an experiment to demonstrate how one could use the estimated class-label distribution in order to improve accuracy with fewer global iterations in FL. We use the US Census Income dataset for this purpose. At a certain global iteration where the estimate of class-label distribution is fairly accurate, an additional step is undertaken by the server wherein clients with imbalanced class-label distribution estimates are removed (their contribution is not considered while updating the model parameters). We evaluate the accuracy of the global model at iteration t for a validation set of 100 samples using the AUC (area under the ROC curve) score, since the considered data set involves binary classification. As a baseline we consider that no action is taken to address class imbalance. Fig. 8 shows the AUC with respect to iteration t for the two schemes. We see improved accuracy and faster convergence to higher AUC when removing clients based on the class imbalance estimate. The improved performance is due to the accurate estimation of the class-label distribution using which we are able to effectively remove the clients with class imbalance and see an improved accuracy in the global model compared with the scheme where no action is taken. This illustrates a benevolent use case of class-label distribution inference.

7 CONCLUSION

In this paper we proposed exact and approximate methods for class-label distribution inference in federated learning systems based on parameter updates from the clients. We evaluated the efficacy of the proposed methods on four different datasets and found that our proposed methods outperform the state of the art by a large margin, and provide a rather accurate estimate of the class-label distribution during training. Our results are not only useful as a means of detecting potential class imbalance in client data sets in FL, but they also reveal potential privacy leakage in FL and provide an effective countermeasure of random oversampling. Furthermore, we also illustrate how the estimate of class-label distribution can be used to address class-imbalance in federated learning.

ACKNOWLEDGMENTS

This work was partly funded by the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications (TECoSA) at KTH and by the Swedish Foundation for Strategic Research through the CLAS project (grant RIT17-0046).

REFERENCES

[1] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks* 10, 3 (2015), 137–150.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (2002), 321–357.

[3] Yen-Hsiu Chou, Shenda Hong, Chenxi Sun, Derun Cai, Moxian Song, and Hongyan Li. 2022. GRP-FED: Addressing Client Imbalance in Federated Learning

Table 2: Mean MSE at iteration $t = 1$ when conditions for exact inference are met

Dataset	**Initialized bias**	**Soft-label**	**Aux-bias grad**	Aux-weight grad	Wang et.al [17]	Wang (GitHub)
UCI Census	9.473×10^{-12}	9.471×10^{-12}	1.07×10^{-11}	0.7446	0.1950	0.1861
MNIST	2.986×10^{-11}	2.998×10^{-11}	2.924×10^{-11}	0.0033	0.1132	0.0932
CIFAR-10	3.018×10^{-11}	3.023×10^{-11}	2.667×10^{-11}	0.0106	0.1227	0.1324
CIFAR-100	3.155×10^{-10}	3.155×10^{-10}	3×10^{-10}	9.675×10^{-4}	0.0848	0.0084

via Global-Regularized Personalization. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 451–458.

[4] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[5] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. 2020. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2020), 59–71.

[6] Moming Duan, Duo Liu, Xinyuan Ji, Yu Wu, Liang Liang, Xianzhang Chen, Yujuan Tan, and Ao Ren. 2021. Flexible Clustered Federated Learning for Client-Level Data Distribution Shift. *IEEE Transactions on Parallel and Distributed Systems* (2021).

[7] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*. 272–279.

[8] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients - How easy is it to break privacy in federated learning?. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 16937–16947. https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf

[9] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[10] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[11] Charles X Ling and Chenghui Li. 1998. Data Mining for Direct Marketing: Problems and Solutions.. In *International Conference on Knowledge Discovery Data Mining*, Vol. 98. 73–79.

[12] Jiahua Ma, Xinghua Sun, Wenchao Xia, Xijun Wang, Xiang Chen, and Hongbo Zhu. 2021. Client selection based on label quantity information for federated learning. In *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 1–6.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[14] Fan Mo, Anastasia Borovykh, Mohammad Malekzadeh, Hamed Haddadi, and Soteris Demetriou. 2020. Layer-wise characterization of latent information leakage in federated learning. *arXiv preprint arXiv:2010.08762* (2020).

[15] Noah A Smith and Roy W Tromble. 2004. Sampling uniformly from the unit simplex. (2004).

[16] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. 2022. User-Level Label Leakage from Gradients in Federated Learning. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (2022), 227–244.

[17] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. 2021. Addressing Class Imbalance in Federated Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10165–10173.

[18] Miao Yang, Ximin Wang, Hongbin Zhu, Haifeng Wang, and Hua Qian. 2021. Federated learning with class imbalance reduction. In *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2174–2178.

[19] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*. PMLR, 5650–5659.

[20] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2020. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610* (2020).

[21] Chunyi Zhou, Yansong Gao, Anmin Fu, Kai Chen, Zhiyang Dai, Zhi Zhang, Minhui Xue, and Yuqing Zhang. 2022. PPA: Preference Profiling Attack Against Federated Learning. *arXiv preprint arXiv:2202.04856* (2022).

[22] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc.

A PROOF OF PROPOSITION 1

Proof. In the binary classification problem, the negative log likelihood of the observed data with respect of parameters $\boldsymbol{\theta}^\top = \begin{bmatrix}\mathbf{w}^\top & b\end{bmatrix}$ is the *cross-entropy* loss function routinely considered for classification schemes as discussed in Section 3. This can be written as

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_i e_i \ln \sigma\left(\mathbf{w}^\top \boldsymbol{h}_i + b\right) + (1 - e_i)\ln\left(1 - \sigma\left(\mathbf{w}^\top \boldsymbol{h}_i + b\right)\right) \tag{44}$$

The weight update for b after a single iteration of full-batch gradient descent can be expressed as

$$\Delta b = b_1 - b_0 = -\eta \frac{\partial L(\boldsymbol{\theta})}{\partial b} = -\frac{\eta}{N}\sum_i \sigma\left((\mathbf{w}^0)^\top \boldsymbol{h}_i + b_0\right) - \boldsymbol{e}_i.$$

Since all weights for the last layer are initialized to zero in the proposition,

$$\mathbf{w}^0 = \mathbf{0} \implies \sigma\left((\mathbf{w}^0)^\top \boldsymbol{h}_i + b_0\right) = \sigma\left(b_0\right) \ \forall i, \tag{45}$$

thus, we get,

$$\Delta b = -\eta\sigma\left(b_0\right) + \frac{\eta}{N}\sum_i \boldsymbol{e}_i \tag{46}$$

rearranging (46) gives (10) and completes the proof. □

B PROOF OF PROPOSITION 2

Proof. The negative log-likelihood (cross-entropy) for the multi-class classifier is

$$L(\boldsymbol{\theta}) = -\frac{1}{N}\sum_i (\boldsymbol{e}_i)^\top \ln \boldsymbol{\sigma}(\boldsymbol{z}_i), \ \boldsymbol{z}_i = \mathbf{W}^\top \boldsymbol{h}_i + \boldsymbol{b} \tag{47}$$

where $\boldsymbol{\sigma}$ refers to the softmax function applied at the output layer. The gradient with respect to the bias $\boldsymbol{b}$ is given by

$$\nabla_{\boldsymbol{b}} L(\boldsymbol{\theta}) = \frac{1}{N}\sum_i \boldsymbol{\sigma}(\boldsymbol{z}_i) - \boldsymbol{e}_i. \tag{48}$$

When initialized with $\mathbf{W}^0 = \mathbf{0}$, the update $\Delta\boldsymbol{b} = \boldsymbol{b}^1 - \boldsymbol{b}^0$ from (48) becomes

$$\Delta\boldsymbol{b} = -\eta\boldsymbol{\sigma}(\boldsymbol{b}^0) + \frac{\eta}{N}\sum_i \boldsymbol{e}_i \tag{49}$$

Rearranging the equations above, we get (18) which completes the proof. □

Table 3: US Census Model Architecture

Layer	Size	Activation function
(input)	104×1	-
Linear -1	32×104	none
Linear -2	16×32	none
Linear -3	8×16	none
Linear -4	2×16	softmax

Table 4: MNIST Model Architecture

Layer	Size	Activation function
(input)	$28 \times 28 \times 1$	-
Conv 2D-1	$3 \times 3 \times 1 \times 8$	ReLU
Maxpool-1	2×2	-
Conv 2D-2	$3 \times 3 \times 8 \times 16$	ReLU
Linear	10×3136	softmax

Table 5: CIFAR-10 and CIFAR-100 Model Architectures. CIFAR-100 has Linear-2 size 100×64

Layer	Size	Activation function
(input)	$32 \times 32 \times 3$	-
Conv 2D-1	$5 \times 5 \times 3 \times 32$	ReLU
Maxpool-1	3×3	-
Conv 2D-2	$5 \times 5 \times 32 \times 32$	ReLU
Maxpool-2	3×3	-
Linear -1	64×1568	ReLU
Linear-2	10×64	softmax

C A GENERIC LOCAL UPDATE

In the mini-batch update, a small batch of data is used to update the parameters and this process is repeated until all the data samples are incorporated in the update. Let $j = 1, 2, \dots J$ index the batch number and $\mathcal{N}_j$ be the set of data samples in batch j. Thus, there are J mini-batch updates. Update after a single iteration can be written as

$$\boldsymbol{b}^{j+1} = \boldsymbol{b}^{j} - \eta \sum_{i \in \mathcal{N}_j} \boldsymbol{\sigma}(\boldsymbol{z}_i^j) - \boldsymbol{e}_i, \tag{50}$$

$$\text{where } \boldsymbol{z}_i^j = (\mathbf{W}^j)^\top \boldsymbol{h}_i + \boldsymbol{b}_k^j \tag{51}$$

A generic local update at a client with J mini-batch updates per epoch, E epochs and weights not initialized to zero is given by

$$\boldsymbol{b}_k^{t+1} = \boldsymbol{b}^t - \eta \sum_{\ell=1}^{E} \sum_{j=1}^{J} \sum_{i \in \mathcal{N}_j} \boldsymbol{\sigma}(\boldsymbol{z}_i^{j,\ell}) + \eta E \sum_i \boldsymbol{e}_i \tag{52}$$

where $\boldsymbol{z}_i^{j,\ell} = (\mathbf{W}^{j,\ell})^\top \boldsymbol{h}_i + \boldsymbol{b}_k^{j,\ell}$, and $\boldsymbol{b}_k^{t+1} = \boldsymbol{b}_k^{J,E}$. Note that the adversary will have access to $\boldsymbol{b}_k^{t+1}$ since it is sent by the client to the server. It also knows $\boldsymbol{b}^t$ as it is sent by the server to all the clients. However, the sum of softmax values over different mini-batches is unknown to the adversary. The quantity of interest is $\sum_i \boldsymbol{e}_i$.

D PROOF OF COROLLARY 2

Proof. Given the conditions $E = 1$ and $\boldsymbol{W}^0 = \mathbf{0}$, from proposition 2 it is clear that the initialized bias estimator, $\hat{\boldsymbol{p}}_{k,1}^{\text{init}}$ is the same as in proposition 2. Thus, $\hat{\boldsymbol{p}}_{k,1}^{\text{init}} = \boldsymbol{p}$.

To prove that $\hat{\boldsymbol{p}}_{k,1}^{\text{soft-label}} = \boldsymbol{p}$ when $E = 1$ and $\boldsymbol{W}^0 = \mathbf{0}$, in (33), $\boldsymbol{\sigma}(\boldsymbol{z}_{c,\text{aux}})$ can be written as

$$\boldsymbol{\sigma}(\boldsymbol{z}_{c,\text{aux}}) = \boldsymbol{\sigma}(\boldsymbol{z}_{c,\text{aux}^1}) = \frac{1}{N_{\text{aux}}} \sum_{i=1}^{N_{\text{aux}}} \boldsymbol{\sigma}(\boldsymbol{b}^0) = \boldsymbol{\sigma}(\boldsymbol{b}^0), \ \forall c \tag{53}$$

Thus,

$$\sum_{c=1}^{C} [\boldsymbol{p}]_c \boldsymbol{\sigma}(\boldsymbol{z}_{c,\text{aux}}) = \boldsymbol{\sigma}(\boldsymbol{b}^0) \sum_{c=1}^{C} [\boldsymbol{p}]_c = \boldsymbol{\sigma}(\boldsymbol{b}^0) \tag{54}$$

Then, the estimator from (33) is

$$\boldsymbol{p}_{k,1}^{\text{soft-label}} = \frac{1}{\eta} \boldsymbol{b}_k^1 - \boldsymbol{b}^0 + \boldsymbol{\sigma}(\boldsymbol{b}^0) = \boldsymbol{p}, \tag{55}$$

which proves that $\boldsymbol{p}_{k,1}^{\text{soft-label}} = \boldsymbol{p}$ under the said conditions.

Now, to prove that $\hat{\boldsymbol{p}}_{k,1}^{\text{grad}} = \boldsymbol{p}$ when $E = 1$ and $\boldsymbol{W}^0 = \mathbf{0}$, from (35) we get

$$\boldsymbol{b}_k^1 - \boldsymbol{b}^0 = \sum_{c=1}^{C} [\boldsymbol{p}]_c (\boldsymbol{b}_{aux,c}^1 - \boldsymbol{b}^0) \tag{56}$$

Now, $\boldsymbol{b}_{aux,c}^1$ can be written as

$$\boldsymbol{b}_{aux,c}^1 = \boldsymbol{b}^0 - \frac{\eta}{N_{\text{aux}}} \sum_{i=1}^{N_{\text{aux}}} \boldsymbol{\sigma}(\boldsymbol{z}_{i,c}) - \boldsymbol{e}_c, \tag{57}$$

where $\boldsymbol{e}_c$ refers to the coordinate vector (one-hot vector) corresponding to class c. From the initial conditions, as in proposition 2, we have $\boldsymbol{z}_{i,c} = \boldsymbol{b}^0 \ \forall i, \forall c$. Then, we can write (57) as

$$\boldsymbol{b}_{aux,c}^1 = \boldsymbol{b}^0 - \eta \boldsymbol{\sigma}(\boldsymbol{b}^0) + \eta \boldsymbol{e}_c \tag{58}$$

Now, (56) becomes

$$\frac{1}{\eta}(\boldsymbol{b}_k^1 - \boldsymbol{b}^0) = \sum_{c=1}^{C} [\boldsymbol{p}]_c (\boldsymbol{e}_c - \boldsymbol{\sigma}(\boldsymbol{b}^0)) = \sum_{c=1}^{C} [\boldsymbol{p}]_c \boldsymbol{e}_c - \boldsymbol{\sigma}(\boldsymbol{b}^0) \sum_{c=1}^{C} [\boldsymbol{p}]_c \tag{59}$$

$$= \sum_{c=1}^{C} [\boldsymbol{p}]_c \boldsymbol{e}_c - \boldsymbol{\sigma}(\boldsymbol{b}^0) \tag{60}$$

The last equality is obtained since $\sum_{c=1}^{C} [\boldsymbol{p}]_c = 1$. Also, note that $\sum_{c=1}^{C} [\boldsymbol{p}]_c \boldsymbol{e}_c = \boldsymbol{p}$ by definition. Thus, rearranging (60) we get (39). □