Privacy Leakage from Logits Attack and Its Defense in Federated Distillation

Danyang Xiao, Diying Yang, Jialun Li, Xu Chen, Weigang Wu*
Sun Yat-sen University. {xiaody, yangdy6, lijlun3}@mail2.sysu.edu.cn. {chenxu35, wuweig}@mail.sysu.edu.cn

***Abstract*—Federated Distillation (FD), a popular variant of Federated Learning (FL), has attracted researchers' attention due to its ability to support heterogeneous model training. Generally, FD allows clients to upload logits associated with public datasets for knowledge transfer, yet logits may pose privacy risks. In this study, we provide the first demonstration of the impact of privacy risks caused by logits. Specifically, we design a data reconstruction attack against logits named L-Attack which can reveal sensitive information about the target client without access to the target model. Via the zeroth-order optimization technique, L-Attack involves training a server-side generator that unveils certain features of private data owned by the target client. To defend against L-Attack, we propose a label aggregation-based FD algorithm called LabelAvg which allows clients to upload predicted hard labels for knowledge transfer instead of logits. Due to the insufficient information in labels for distillation, LabelAvg provides a voting-based label smoothing mechanism that enables the server to construct smooth labels from received labels. The generated smooth labels which stand for the consensus among all clients, indicate the approximate probability distribution. Thus, these smoothed labels bear a striking similarity to logits and can be used for distillation. Analysis and experimental results prove LabelAvg is superior to baselines in terms of accuracy, privacy, and communication data volume.**

***Index Terms*—Federated Distillation, Privacy Leakage, Logits Attack, Distributed Training**

I. Introduction

Federated Distillation (FD) [1], a popular variant of Federated Learning (FL) [2], has risen to prominence recently. FD aggregates local models among clients via knowledge distillation, enabling heterogeneous training where participating local models need not have the same size, structure, or even model types.

From the perspective of data exchange, existing FD variants can be categorized into two paradigms: 1) model-based FD [3] and 2) logits-based FD [4], [5]. The former requires clients to send local models to the server for distillation, e.g., let all local models distill a global model on the server. In the latter, clients need to share logits [1] related to a public dataset with the server for knowledge transfer.

For model-based FD, due to the distillation operation, privacy-preserving techniques such as homomorphic encryption [6] may not be applied to uploaded local models. Therefore, local models in model-based FD may suffer from 'white-box' attacks such as model inversion attacks [7], [8]. Logits-based FD can effectively avoid 'white-box' attacks since the local model is inaccessible to the public. However, recent studies [9], [10] have raised concerns regarding privacy risks from logits, but no prior work has reported how to obtain privacy information from logits.

In this study, we design a GAN-based attack called **L**ogits **A**ttack (**L-Attack**) to reconstruct certain features of private data from logits related to public datasets. To the best of our knowledge, our work is the first to employ a specific attack method to validate that, in addition to parameters, gradients, and activations, logits related to the public dataset may also potentially cause privacy leakage risks in certain scenarios. While following FD protocols, such as FedMD [4] and FedED [4], the server has the ability to reveal sensitive information of clients via L-Attack.

Different from attacks against parameters, L-Attack against logits in FD has two key issues due to the access restrictions of local models: 1) how to construct a GAN silently in FD system? and 2) how to obtain the gradient used to update the server-side generator? L-Attack can handle the above challenges by 1) ingeniously designing the target model to act as a discriminator of GAN during FD training, and 2) computing approximate gradients used to update the server-side generator via zeroth-order optimization techniques. As a result, the trained server-side generator will construct simulated samples that are placed in the public dataset to steal private information by utilizing the discriminative power of the local model of the target client.

To defend L-Attack, perturbing logits, such as adopting Differential Privacy (DP) techniques [9], may be one of the effective ways, but these methods may cause a loss of trained model accuracy. Anomaly detection techniques are another methods that can defend against L-Attack. However, most anomaly detection techniques, such as clustering-based or statistical-based approaches, may incur significant computational overhead when dealing with a substantial number of samples during each communication round. Besides, these methods may suffer from high false positive rates.

In this paper, we propose a novel label aggregation-based FD algorithm called Label Averaging (shortened to **LabelAvg**), which allows clients to upload predicted hard labels instead of logits for knowledge transfer, so as to address attacks from logits including L-Attack. However, using these hard labels for distillation may not be conducive to knowledge transfer among clients, as hard labels lack useful information such as probability distribution. To address this issue, LabelAvg provides a voting-based label smoothing mechanism

[1]Logits (a.k.a., Class Scores) are predictions output from the neural network. Logits can be regarded as 'soft labels' containing predictive probability information for each category.

2158-3927/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN58291.2024.00029

that lets the server construct smoothed labels (bearing a similarity to logits) containing an approximate probability distribution based on received hard labels. More specifically, the approximate probability distribution is obtained by voting, that is, determining the frequency of each class by counting the number/vote of received labels for each class. Thus, generated smoothed labels can be regarded as the global consensus similar to average logits, and used for distillation. In particular, due to the lack of inter-class information in transmitted hard labels, LabelAvg instructs clients to send related label weights for better modeling probability distribution. In addition, due to the designed mechanism, LabelAvg can reduce communication data volume effectively while ensuring accuracy and protecting data privacy.

Contributions. Our contributions are as follows:

- **Attack.** A GAN-based attack named L-Attack is introduced. L-Attack can reconstruct private data from logits related to public data via ingenious design and zeroth-order optimization. Our work is the first to employ a specific attack method to validate that logits may pose privacy risks in FD.
- **Defense.** A defense method against L-Attack called LabelAvg is proposed. LabelAvg allows clients to send hard labels instead of logits for aggregation and enables the server to construct smoothed labels containing an approximate probability distribution for distillation. Relevant analysis and comprehensive experiments regarding LabelAvg are provided.

II. Related Work

A. Federated Distillation

Recent work introduces codistillation [11], a distributed distillation algorithm, which can distill several models into a single model so as to accelerate distributed training. Inspired by codistillation, some researchers proposed an FL framework called FedMD [5] which lets clients upload logits related to public datasets to the server for aggregation instead of model parameters. FedMD allows the server to calculate the average logits based on the received logits, which are then sent back to the clients. The clients utilize these average logits to align the outputs of their local models. Different from FedMD, under FedED [4] settings, average logits are utilized for distilling a global model on the server side.

In order to protect privacy, some Differential Privacy (DP)-based FDs, such as FedMD-LDP and FedMD-NFDP introduced in the literature [9], require each client to adopt DP techniques before uploading their logits to the server. Besides, some researchers propose FedDF [3] that aggregates the global model by combining FedAvg and Knowledge Distillation (KD) so as to improve the ability of data privacy protection. FedKT [1] reduces the risk of privacy leakage by a two-tier knowledge transfer structure. However, FedKT needs to perform distillation on the client and server sides, which brings a lot of computational overhead.

In order to reduce communication overhead, recent work [12] presents one-shot federated learning, where a central server learns a global model over all received local models in a single round of communication by drawing on ensemble learning and knowledge aggregation so as to reduce communication overhead. Recent work [13] presents a novel method named FedKD that is both communication-efficient and effective, based on adaptive mutual knowledge distillation and dynamic gradient compression techniques. Besides, some variants of FedMD are proposed for reducing communication data [14].

Recently, decentralized FD has attracted researchers'attention since it can avoid dependence on the server. Def-KT [15] is an effective decentralized FD that allows clients to share their local models for mutual knowledge transfer. Different from Def-KT, CMFD [16] allows clients to share logits related to public datasets for knowledge transfer instead of models.

To sum up, FDs mentioned above can be classified into two paradigms based on the type of data exchange, namely, logits-based FD, such as FedMD and FedMD-LDP, and model-based FD, such as FedDF and FedKT. Moreover, existing FDs can perform distillation operations on either the server or client side based on specific requirements, and most FDs require a public dataset for knowledge transfer. Table I compares the differences among mainstream FD algorithms.

TABLE I
DIFFERENCES AMONG MAINSTREAM FD ALGORITHMS.

Terms	References
Logits-based FD	[4], [5], [9], [10], [14], [16]
Model-based FD	[1], [3], [12], [15]
Distillation on the server side.	[1], [3], [4], [11], [12]
Distillation on the client side.	[1], [5], [9], [10], [14]–[16]
With the help of public datasets	[1], [3]–[5], [9]–[12], [14]
Distillation for global model updating	[1], [3], [4], [11], [12]
Distillation for local model updating	[5], [9], [10], [14], [15]
Central FD	[1], [3]–[5], [9]–[12], [14]
Decentralized FD	[15], [16]

B. Attacks against FL

Many existing works have studied the security [17] and risks [18], [19] in FL/FD, including risks from parameters, gradients, activation values, and logits, respectively.

Parameters. Inversion attack [8], [20] is one of the common threats to FL in which adversaries attempt to steal and infer sensitive information about training data from models. In the literature [7], authors design a GAN-based inversion attack to reconstruct raw data from the target model. This work assumes there exists a malicious client that can utilize GAN to extract sensitive information about target clients against model parameters downloaded from the server.

Gradients. Existing works [21], [22] convey that adversaries can reconstruct raw data from gradients. In the literature [23], authors design a reconstruction attack called DLG, an optimization algorithm that reconstructs raw data by minimizing the distance between mimicking gradients and captured real gradients. Subsequently, several variants of DLG are successively proposed [24]. Simultaneously, many defenses against gradient leakage are also introduced in the literature [25], [26]. In literature [27], authors propose Fragmented Federated Learning (FFL), a lightweight solution against

gradient-based reconstruction attacks. To reduce privacy risks from gradients, FFL contains a gradient-obscuring algorithm that allows clients to select some safe layers' gradients for submission. In literature [28], researchers provide an analysis of observations, explaining how data presentation leaks from gradients. Based on their observation, they propose a defense called Soteria which learns to perturb data representation (i.e., gradients). These perturbed data representations can prevent reconstruction attacks while keeping FL performance. Defenses introduced in [27], [28] can prevent gradient-based data reconstruction attacks in typical FL. However, the mentioned above defenses are not applicable to FD since the server and clients communicate with each other by logits instead of gradients. Our work proposes a novel defense that can resist logits-based data reconstruction attacks in FD.

Activations. Activations exchanged in Split Learning (one of FL variants) [29], [30] may suffer from reconstruction attacks [31], [32]. In the literature [31], authors propose a defense mechanism to prevent attacks against activations.

Logits. Recent studies [9], [10] have raised concerns regarding the privacy risks from logits, and proposed two DP-based FDs based on FedMD to perturb logits for privacy protection. Until now, there has been no empirical demonstration of the privacy risk caused by logits via specific attack methods.

III. Preliminaries

Suppose that there are N clients ($\{P_i\}_{i=1}^{N}$) in a FD system. Each client (P_i) holds a private dataset ($\mathcal{D}_{pri}^{(i)}$) and its local model parameters ($\theta_{local}^{(i)}$). The structure of the local models can be different. In addition, the server holds a public dataset ($\mathcal{D}_{pub}$) that everyone can access. Typically, the objective of FD training is to improve the performance of the local model by allowing local training on the private dataset and leveraging knowledge transfer from other clients with the help of the public dataset. For each local model ($\theta_{local}^{(i)}$), the objective can be simply defined as follows:

$$min_{\theta_{local}^{(i)}}[\mathcal{L}_{local}(\theta_{local}^{(i)}, \mathcal{D}_{pri}^{(i)}) + \mathcal{L}_{kd}(\theta_{local}^{(i)}, \mathcal{D}_{pub})], \quad (1)$$

where $\mathcal{L}_{local}(\cdot)$ and $\mathcal{L}_{kd}(\cdot)$ denote the training optimization procedure on the private dataset and the distillation optimization procedure on the public dataset, respectively. In this paper, we introduce the proposed attack under the basic setup of FedMD which is one of the mainstream FD algorithms. The details of FedMD are illustrated in Figure 1. Note that the proposed attack can also be applied to other FD algorithm settings such as FedED.

The procedure of FedMD

The server holds a public dataset ($\mathcal{D}_{pub}$) for distillation. The entire training procedure consists of the initialization phase and the distillation phase.

Initialization Phase: Each client (P_i) trains its local model over several iterations on the private dataset ($\mathcal{D}_{pri}^{(i)}$) and public dataset ($\mathcal{D}_{pub}$).

Distillation Phase: N clients follow 4 steps below to train their model:

Step 1: The server distributes the public dataset ($\mathcal{D}_{pub}$) to all clients. Each client (P_i) computes logits ($f_i(x_{pub})$) on public dataset ($\mathcal{D}_{pub}$) and then transmits the result to the server.

Step 2: The server computes average logits for each sample (x_{pub}) of the public dataset based on received logits:

$$\overline{f}(x_{pub}) := \frac{1}{N}\sum_{i=0}^{N-1} f_i(x_{pub}), x_{pub} \in \mathcal{D}_{pub}. \quad (2)$$

Step 3: Each client downloads all average logits ($\overline{f}(x_{pub})$) and uses them to compute the loss of distillation:

$$loss := \mathbb{E}_{(x_{pub})\sim\mathcal{D}_{pub}}[\ell_{kd}(f_i(x_{pub}), \overline{f}(x_{pub}))]. \quad (3)$$

Step 4: Each client (P_i) trains its local model over several iterations on private dataset($\mathcal{D}_{pri}^{(i)}$).

The training repeats the above 4 steps until each local model reaches a convergence point.

Fig. 1. The procedure of FedMD

IV. L-Attack: Privacy Risks from Logits

A. Threat Model

In general, our threat model follows literature [7], where there is a semi-honest adversary in the system along with other honest participants. Different from literature [7], the adversary in our work is concealed on the server side rather than the client side. More specifically, all participants adopt an FD protocol such as FedED [4] or FedMD [5] for distributed training. We assume that the attacker (i.e., the server) does not have information about the clients participating in FD, including private data, model types, and training configurations. All clients are willing to trust that logits-based FD is reliable, so they only follow the basic operations of the typical FD procedure, meaning they do not employ additional techniques to detect public samples and encrypt logits. As a result, all clients are potential victims.

Objectives. Based on special requirements, the server (adversary) chooses a client as a target (victim), and tries to reveal information about a class of data it does not hold. Also, the adversary surreptitiously influences the learning procedure to deceive a victim into releasing details about the target class.

Conditions. The adversary has access to a public dataset ($\mathcal{D}_{pub}$) that is similar to the clients' private dataset. In each communication round, the server receives logits from all clients regarding $\mathcal{D}_{pub}$. Also, we assume that the server periodically distributes partial samples of the updated public dataset to all clients, which is allowed in real-world applications since public datasets require continuous supplementation and refinement to enhance the performance of FD [5].

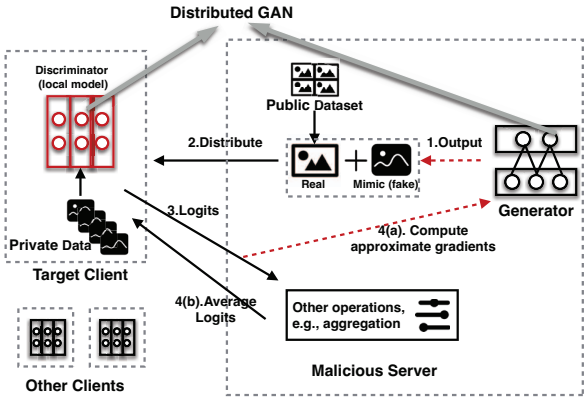


Fig. 2. The procedure of L-Attack

Ideas. The adversary employs a GAN-based attack by surreptitiously utilizing logits associated with public data to train a generator. This generator is used for reconstructing private data. Simultaneously, the adversary follows the protocol specification as viewed by his victims. E.g., the adversary sends and receives logits as required by the FD training.

B. The Procedure of L-Attack

The core of L-Attack is to train a distributed GAN used for generating data that simulates real data. These simulated data can ultimately infer sensitive information about private data. Figure 2 demonstrates the procedure of L-Attack. L-Attack is ingeniously designed to enable the local model (θ_{local}) of the target client to act as a discriminator of distributed GAN. Simultaneously, L-Attack trains the server-side generator (G and its parameter θ_g) of GAN with the help of the discriminative power of the target client's local model. As a result, the generator generates fake data that mimics certain characteristics of the private data of the target client. Due to the access restriction of the target model, the server can not access gradients used to update the generator. To alleviate this issue, L-Attack computes approximate gradients via the zeroth-order optimization technique. Algorithm 1 describes details of L-Attack. L-Attack allows the server to engage in FD as usual while enabling the server to silently attack the target client during certain communication rounds. All steps of the procedure are as follows:

Step 1: The server distributes the public dataset ($\mathcal{D}_{pub}$) to all clients. In particular, in attacking rounds, some simulated samples (x_{fake}) generated by the generator are put into the public dataset ($\mathcal{D}_{pub}$) (Lines 5-7 in Algorithm 1).

Step 2: The target client trains its local model over several iterations on the private dataset ($\mathcal{D}_{pri}$), followed by computing logits ($f_{target}(x_{pub})$) on the public dataset ($\mathcal{D}_{pub}$) and transmitting the result to the server (Lines 9-10 in Algorithm 1).

Step 3: Based on received logits, the server computes average logits ($\overline{f}(x_{pub})$) for each sample (x_{pub}) of the public dataset. In normal rounds, the server returns average logits directly to the target client without any operation (Line 18 in Algorithm 1). In attacking rounds, the server makes use of the target client's logits associated with simulated samples to compute approximate gradients via the zeroth-order optimization technique. These approximate gradients are used to update the server-side generator (Line 14 in Algorithm 1). In addition, to ensure that the target client continues the training procedure, the server takes the raw logits received from the target client as average logits and sends them back (Lines 11-15).

Step 4: After receiving average logits, the target client computes the loss of distillation so as to improve local model performance (Line 21 in Algorithm 1).

Algorithm 1 L-Attack Based on Zeroth-Order Optimization

1: **Input:** $\mathcal{D}_{pri}, \mathcal{D}_{pub}, \theta_g, \theta_{local}, x \in D_{pri}, x_{pub} \in \mathcal{D}_{pub}$
2: **Output:** fake data $(x_{fake}) \approx x$
3: **for** t in $[0, T)$ **do**
4: **Server:**
5: $x_{fake} \leftarrow G(\theta_g)$
6: $\mathcal{D}_{pub} \leftarrow \mathcal{D}_{pub} \cup x_{fake}$
7: Distribute public dataset D_{pub}
8: **Target Client:**
9: $\theta^{(t)}_{local} \leftarrow update(\theta^{(t-1)}_{local}, \mathcal{D}_{pri})$
10: $f_{target}(x_{pub}) \leftarrow inference(x_{pub})$
11: **if** it is an attacking round **then**
12: **Server:**
13: Compute $\frac{\partial \mathcal{L}}{\partial x}$ by Eq. (6)
14: Update θ_g by Eq. (4-5)
15: Return $f_{target}(x_{pub})$ to the target client
16: **else**
17: **Server:**
18: Compute and return $\overline{f}(x_{pub})$
19: **end if**
20: **Target Client:**
21: Compute $loss := \mathcal{L}_{kd}(f_i(x_{pub}), \overline{f}(x_{pub}))$
22: **end for**

C. Gradient Approximation

Training GAN needs to alternately update the discriminator (i.e., local model of target client) and generator (θ_g on the server side). In particular, the parameters of the generator are updated by gradient descent:

$$\theta_g^{(t+1)} := \theta_g^{(t)} - \eta \nabla_{\theta_g} \mathcal{L}, \tag{4}$$

where η and $\nabla_{\theta_g}\mathcal{L}$ denote the learning rate and the derivative of the loss function of the local model ($\mathcal{L}$), respectively. Based on Chain Rule, $\nabla_{\theta_g}\mathcal{L}$ can be decomposed into two components as shown in Eq. (5):

$$\nabla_{\theta_g}\mathcal{L} := \frac{\partial \mathcal{L}}{\partial \theta_g} := \frac{\partial \mathcal{L}}{\partial x} \times \frac{\partial x}{\partial \theta_g}. \tag{5}$$

Since the server does not have access to the local model of the target client (i.e., the local model is a 'black box'), the server can not obtain gradients ($\frac{\partial \mathcal{L}}{\partial x}$) associated with simulated samples calculated by the target model. To cope with this issue, L-Attack lets the server approximately compute $\frac{\partial \mathcal{L}}{\partial x}$ by zeroth-order optimization techniques.

Zeroth-order optimization [33], [34] refers to all optimization methods that do not require gradient information, while in general, it refers to optimization algorithms that estimate the

direction of parameter updates based on the idea of parameter sampling and differencing. In this paper, we estimate $\frac{\partial \mathcal{L}}{\partial x}$ by Forward Differences Method [33] defined by Eq. (6):

$$\frac{\partial \mathcal{L}}{\partial x} \approx \frac{1}{m}\sum_{i=1}^{m} d\frac{f(x+\epsilon u_i)-f(x)}{\epsilon}u_i, \tag{6}$$

where u_i is a random variable drawn from a d dimensional unit sphere with uniform probability and ϵ is a small positive constant called the smoothing factor. The gradient can be estimated by computing the forward difference using m random directions($u_1, ..., u_m$). Under L-Attack setting, random variables ($x + \epsilon u_i, i \in [1, m]$) associated with x act as simulated samples and are distributed to the target client.

D. Impact and Significance.

In practice, an adversary from the server only needs to stealthily maintain a generator on the server during FD training. Afterward, in addition to performing basic logits aggregation operations, the server utilizes L-Attack to optimize the generator model until the generated simulated images can leak private information. In certain scenarios, L-Attack makes the training of logits-based FDs very vulnerable. Our attack method can acquire information about the private data of the victim (target client) regarding specified categories/labels, obtaining an understanding of original samples' content. Furthermore, our work opens up a research door that explores generative models to trigger data reconstruction attacks in logits-based FD. We believe that future enhancements based on L-Attack may increase privacy risks targeting logits, calling for further research on defense algorithms.

V. LabelAvg: Defense against L-Attack

A. The Procedure of LabelAvg

As described in Algorithm 2, after the initialization phase, LabelAvg carries out the following steps in several epochs:

Communication: For each public data sample, each client infers/predicts several 'hard' labels, and computes the corresponding weights for these labels. At the end of this step, all labels and weights are sent to the server for aggregation.

Aggregation: After receiving predicted labels and relevant labels'weights, the server computes the updated consensus, i.e., smoothed labels, and then sends them to clients for distillation.

Distillation[2]**:** After receiving the updated consensus, each client trains its model to approach the consensus related to the public dataset, i.e., distillation operations.

Revisit: Each client trains its local model on their private dataset for several local epochs.

There exists two challenges in the above-mentioned procedure. **Challenge 1:** How to incorporate more inter-class information by computing weights when clients upload 'hard' labels? **Challenge 2:** How to utilize the received labels to construct smoothed labels that bear a similarity to logits? LabelAvg provides a voting-based label smoothing mechanism to address them.

[2]In literature, the distillation phase is also named by digest phase.

Algorithm 2 The Procedure of LabelAvg

Input: Private datasets $\{\mathcal{D}^{(i)}|i \in \{1, ..., N\}\}$, Public dataset $\mathcal{D}_{pub}$, T, N
Output: Local models parameters $\{\theta_i|i \in \{1, ..., N\}\}$.
Initialization: Each client i trains its local model to convergence on $\mathcal{D}_{pub}$ and $\mathcal{D}^{(i)}$.
for $t \in \{1, .., T\}$ **do**
  for $i \in \{1, .., N\}$ parallel **do**
    Communication: client i uploads labels and weights computed by Eq.(7),(8),(9),(10).
  end for
  Aggregation on the server: 1) Converts labels,weights into v and w,respectively. 2) Constructs v_{smooth} by Eq.(11)-(12). 3) Sends v_{smooth} to all clients.
  for $i \in \{1, .., N\}$ parallel **do**
    Distillation: client i trains θ_i on $\mathcal{D}_{pub}$ by minimizing $distillation\ loss \leftarrow CrossEntropy(\widehat{y}, v_{smooth})$.
    Revisit: client i iteratively trains θ_i on $\mathcal{D}^{(i)}$.
  end for
end for

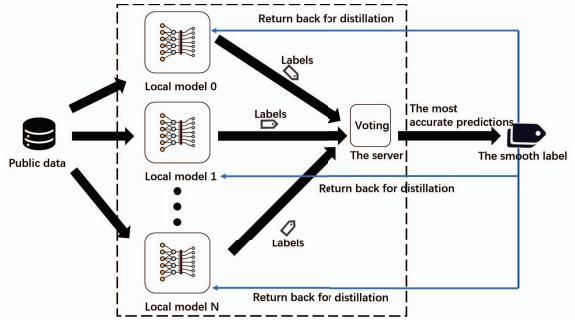


Fig. 3. The basic idea of LabelAvg. The dashed box can be seen as a process resembling ensemble learning. Note that the displayed process does not adopt the voting-based label smoothing mechanism.

B. Voting-Based Label Smoothing

Figure 3 illustrates the basic idea of LabelAvg and we design the core mechanism based on this idea. As usual, client P uses its local model (θ_P) to compute logits ($\widehat{y}$) on the public dataset, and then infers 'hard' labels (ζ) according to logits. Concretely, for each public data sample (x_i), Client P selects index values of the top K maximum element values of $\widehat{y}$ as potential labels, that is,

$$\widehat{y}_i := f(\theta_P, x_i), \tag{7}$$

$$(\mathcal{S}_P)_i := \{\zeta_i | \widehat{y}_{i,\zeta_i} > \kappa\}, \tag{8}$$

where $(\mathcal{S}_P)_i$ denotes the set of predicted labels for the given sample (x_i) and contains K labels that need to be sent to the server. In particular, $\widehat{y}_{i,\zeta_i}$ and κ denote the value of the ζ_i-th column of $\widehat{y}_i$ and the value of the K^{th} largest elements of $\widehat{y}_i$, respectively. Actually, $\widehat{y}_{\zeta_i}$ indicates the probability of ζ_i-th class label. $\zeta_i \in \{0, C-1\}$, where C denotes the number of categories.

Sending labels can reduce the provision of excessive information to the public, but it is not conducive to knowledge transfer during the distillation, that is, uploading labels alone may not fully capture inter-class information, which could limit the server's ability to obtain the probability distribution.

Solution to Challenge 1. To compensate for more inter-class information, the proposed mechanism calculates the weight for each uploaded label based on the relative magnitudes of element values of the corresponding logits, as the relative magnitudes of the element values can reflect the local model's preference for each category without disclosing the specific element values. Let $\widehat{y} = [\widehat{y}_0, \widehat{y}_1, ..., \widehat{y}_C]$ denote logits containing C classes. The computation of weights is defined as follows:

$$\zeta_{max} := \underset{\zeta}{\operatorname{argmax}}\ \widehat{y}_\zeta, \tag{9}$$

$$\beta_\zeta := \frac{\widehat{y}_\zeta}{\widehat{y}_{\zeta_{max}}}, \zeta \in \mathcal{S}, \tag{10}$$

where $\widehat{y}_{\zeta_{max}}$ denotes the maximum element value of original predicted logits ($\widehat{y}$) and β_ζ is the weight corresponding to label ζ. Consequently, these weights, together with the corresponding labels, are sent to the server. On the server side, these weights are converted into vectors (w_P) of client P, and used for computing v_{smooth} by Eq. (11).

Solution to Challenge 2. The proposed mechanism profiles the predicted probability distribution via voting, where each received label contributes one vote. More specifically, for the prediction of each public data sample, the server receives KN 'hard' labels from N clients (K labels per client). Inspired by voting-based ensemble methods, LabelAvg determines the frequency of each class by counting the number/vote of received labels for each class, and then derives probabilities of category from these frequencies. In addition, LabelAvg assigns additional labels' weights to better profile probabilities of categories. Finally, LabelAvg assembles probabilities of categories into a normalized vector resembling logits. For each public sample, the above operations can be defined as follows:

$$v_{smooth} := \frac{1}{KN} \sum_{P=1}^{N} w_P \odot v_P, \tag{11}$$

where v_P denotes multi-hot vector transformed from label set ($\mathcal{S}_P$) belonging to client P. v_{smooth} denotes the smoothed label where each column indicates the probability for each class. w_P denotes the labels' weight and can be configured to all-ones vector. In our work, we configure weights calculated in Eq. (10) as w_P. In particular, $\odot$ denotes element-wise multiplication operations between two vectors. From the perspective of voting-based ensemble methods, smoothed labels (v_{smooth}) can be seen as the final decision obtained by voting on the outputs of several local models which can be viewed as base models.

In particular, LabelAvg designs a mixing factor to adjust the confidence of the label with the maximum number of votes, so as to better profile predicted probabilities distribution:

$$(v_{smooth})_i := (1-\alpha) \cdot (v_{main})_i + \alpha \cdot (v_{smooth})_i, \tag{12}$$

where $(v_{main})_i$ denotes the one-hot vector transformed from the label with the maximum number of votes regarding sample i. Besides, $\alpha (0 \leq \alpha \leq 1)$ denotes the mix factor. For labeled public datasets, the ground-truth label regarding the public sample can be considered as $(v_{main})_i$, and it is blended into the generated smoothed label ($(v_{smooth})_i$) so as to reduce class bias caused by label smoothing.

After aggregation, the server returns generated smoothed labels to clients. Subsequently, for each public data sample $\{((x_{pub})_i, (v_{smooth})_i)\} \in \mathcal{D}_{pub}$, clients calculate the distillation loss to update local models via cross-entropy instead of KL divergence:

$$\mathcal{L}_{ce}((v_{smooth})_i, \widehat{y}_i) := -\sum_{j=0}^{C-1} (v_{smooth})_{i,j} log((\widehat{y})_{i,j}), \tag{13}$$

where $\widehat{y}_i$ is the local model's prediction for public data sample $((x_{pub})_i)$. In practice, the implementation is similar to existing FD algorithms like FedMD, with the main difference being that clients only need to send K predicted labels and their corresponding weights for each public data sample. During the aggregation process, for each client's label and corresponding weights, the server needs to transform them into vectors separately for ease of calculation.

C. Analysis and Discussion

We analyze and discuss LabelAvg to answer the following questions (**Q**):

Q1: Why can LabelAvg effectively transfer knowledge via smoothed labels?

Clients transfer knowledge through smoothed labels (i.e., the global consensus) under LabelAvg settings. We discuss the effectiveness of LabelAvg by analyzing smoothed labels from several perspectives.

• **Profiling the Approximate Probability Distribution.** The smoothed label, the consensus among all clients under the LabelAvg settings, may indicate an approximate ground-truth distribution. More specifically, In FD, the aggregated consensus, such as average logits in FedMD, and the generated smoothed labels in LabelAvg, can be considered as outputs from teacher models guiding client-side local models (student models). Thus, the probability distribution indicated by these outputs is crucial for the effectiveness of knowledge transfer in the distillation. The relevant evaluations are presented in the following experimental section.

• **Applying Effective Regularization.** The aggregated consensus in LabelAvg can be regarded as a variant of smoothed label regularization, in which we replace the uniform distribution with a probability-based distribution. More specifically, Label Smoothing (LS) [35]–[37] is employed to consider the non-ground truth distribution by setting non-ground truth terms to small weighted values instead of zeros, so that the model may not be too confident about the ground truth. Via LS, training models can avoid over-fitting. Instead of using hard labels for loss computing, typical LS [35] utilizes soft labels that are generated by mixing a uniform distribution with the distribution of hard labels. In particular, the probability ($q'(c|x_i)$) of the data sample(x_i) being class c in the smoothed label can be defined as:

$$q'(c|x_i) = (1-\alpha)q(c|x_i) + \alpha u(c|x_i), \tag{14}$$

where α and $u(c|x_i) = \frac{1}{C}$ denote the mixing factor and the uniform distribution used to smooth labels, respectively. $q(c|x_i)$ can be regarded as the probability needed to be 'smooth'. Instead of the uniform distribution, LabelAvg assigns probabilities of non-target categories based on voting. According to Eq. (11)-(12), for each non-target category (c), LabelAvg assigns its probabilities ($u(c|x_i)$) by:

$$u(c|x_i) := (v_{smooth})_c = \frac{\sum_{P=1}^{N}(w_P)_c(v_P)_c}{KN}. \quad (15)$$

Compared with typical LS, the smoothed label generated by LabelAvg can profile the predicted probability distribution. Evaluations are shown in the following experimental section.

• **Reducing Overfitting and Enhancing Robustness.** As indicated by the dashed box in Figure 3, the aggregation process of LabelAvg can be viewed as a voting-based ensemble learning process. Intuitively, LabelAvg enables local models to act as base models, and voting-based label smoothing mechanism can be considered a meta-model that outputs the most accurate predictions by combining all labels received from local models. During each distillation procedure, LabelAvg can provide higher confidence and stable predictions (i.e., smoothed labels) used for aligning the outputs of local models. Compared with logits, aligning with smoothed labels can reduce overfitting. As a result, LabelAvg can enhance the robustness of local models, indicating its effective handling of data heterogeneity. Our experiments demonstrate that LabelAvg performs well across scenarios under four different Non-IID settings.

Q2: What advantages does LabelAvg have compared to existing techniques?

We discuss LabelAvg along with existing defense techniques such as Anomaly Detection, Differential Privacy-based Logits, and Sparse Logits in terms of accuracy, costs, and privacy preservation.

• **LabelAvg vs. Anomaly Detection.** Detecting public data before the start of each round has proven to be an effective method. Anomaly detection techniques can keep the accuracy of the trained model. Compared with LabelAvg, several anomaly detection techniques have some shortcomings. In terms of privacy preservation, some anomaly detection techniques may achieve high false positive rates which sometimes may not prevent L-Attack. In terms of computational and communication costs, some techniques such as clustering-based or statistical-based approaches may incur significant computational overhead when dealing with a substantial number of samples during each communication round.

• **LabelAvg vs. Differential Privacy-based Logits.** In FD training, Differential privacy techniques can effectively deal with the trade-off between accuracy and the ability to protect logits by appropriately configuring hyperparameters. LabelAvg achieves higher accuracy and privacy protection capabilities. Importantly, it significantly reduces communication overhead compared to Differential Privacy-based Logits during training. Therefore, in some communication-sensitive FD scenarios, LabelAvg may be a preferable choice over differential privacy.

• **LabelAvg vs. Sparse Logits.** LabelAvg enables clients to send Top-K hard labels and their re-normalized weights. It appears that LabelAvg sparsifies logits before communication. Therefore, a question arises: why do we design LabelAvg instead of directly sparsing logits? In practice, we find that using sparse logits for knowledge transfer results in significant accuracy losses. We believe that, during the distillation process, aligning the outputs of local models with sparse logits may lead to model overfitting. Particularly, in sparse logits, some elements of the raw logits are still retained. Consequently, there remains a potential risk of privacy leakage in sparse logits.

VI. Experiments

We first demonstrate experimental results regarding L-Attack, followed by the evaluation regarding LabelAvg.

A. L-Attack Demonstration

Experimental Methodology. Our simulation system in experiments regarding L-Attack demonstration follows FedMD protocol [5], which is one of the common FDs. Figure 1 shows details of FedMD. Some important hyperparameters regarding simulation are as follows: The dimension of noise (i.e., the input of the generator) is set to 100. The parameter 'ϵ' (used in the forward differences method) is set to 0.1. For 'm' (used in the forward differences method), it is tailored to datasets, with values set to 5, 10, and 20. The learning rate and batch size are set to 0.001 and 100, respectively. Note that, we adopt the early stopping strategy and do not stop training until generated images satisfy the requirement.

In a simulation system with several clients adhering to FedMD's training protocol, the server uses L-Attack to steal sensitive information about the training data of target clients. Datasets used for demonstration are MNIST, Fashion-MNIST, and CelebA[3]. To simulate the distribution of the public dataset, for each experiment, 1/3 of the dataset is divided into the public dataset samples and each client only holds samples of two categories. Note that, in order to ensure the reliability of the conclusion, the divided public dataset does not include relevant samples about the categories owned by the target client. We believe that, via the above-mentioned data partitioning strategy, the public dataset and the private dataset can be considered as two independently unrelated datasets. For example, in the experiments on MNIST, the private dataset can be regarded as a handwritten digit dataset containing digits 0-6, while the public dataset can be seen as a handwritten digit dataset containing digits 7-9.

Gradient Visualization. We compare the differences between estimated gradients and real gradients by the gradient distribution histogram. Figure 4 demonstrates differences among real gradients in hindsight, as well as the estimated gradients with respect to FedMD and LabelAvg. We observed that L-Attack can approximate the distribution of the true gradient from unprotected logits, but it fails to effectively estimate the distribution of update gradients from smoothed

[3]https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

labels. This observation indicates that smoothed labels are effective in preventing the computation of approximate gradients, implying that LabelAvg can defend against L-Attack. Our data reconstruction attack simulation demonstrates the ability to protect data privacy (shown in Figure 13). Note that we only show a few representative examples, and most gradient examples during the training indicate similar results.

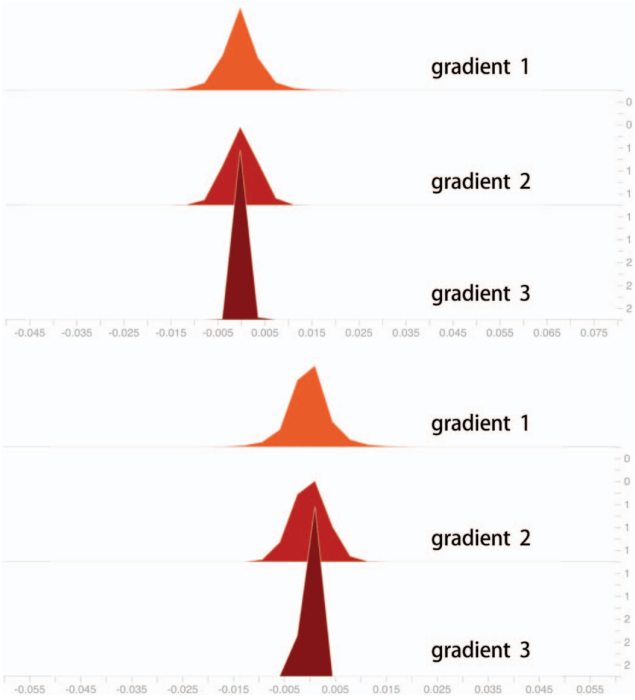


Fig. 4. Gradient visualization. Figures display the distribution range of element values in the gradients. Two subfigures respectively illustrate comparisons for two cases, where 'gradient 1' represents the real gradient, 'gradient 2' represents the gradient estimated by L-Attack from the logits regarding FedMD, and 'gradient 3' represents the gradient estimated by L-Attack from the smoothed labels regarding LabelAvg.The more similar the distribution shapes, the closer the associated gradients approximate each other.

Experiments on MNIST. In these experiments, clients holding samples corresponding to labels "0", "1", and "6" are regarded as target clients (victims). The goal of the server is to steal private information related to labels "0", "1", and "6". The experimental results are shown in Figure 5. From the results, we can observe that information about private samples can be captured by the server-side generator. The data with labels "0", "1" and "6" represent the local data of the handwritten numbers "0", "1" and "6", respectively.

Experiments on Fashion-MNIST. As in MNIST experimental setup, these groups of experiments assume that target clients have samples related to labels of "0", "3", and "9" respectively. The results are shown in Figure 6. From the results, simulated images generated by the server-side generator reveal information about private samples, i.e., the private samples labeled with "0", "3", and "9" represent the class of images about shirts, skirts, and shoes, respectively. In sum, we believe that simulated images reconstructed by the generator

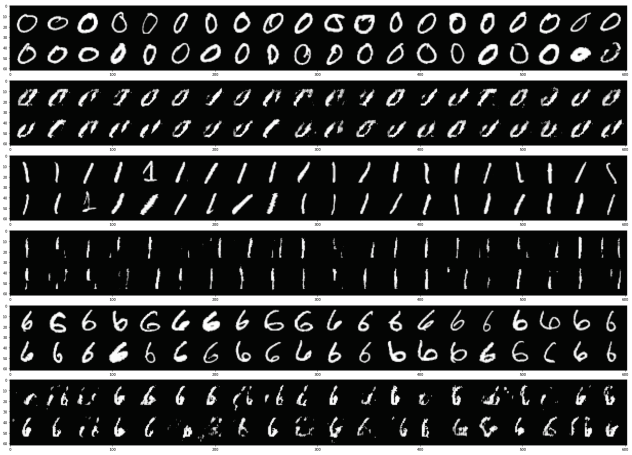


Fig. 5. Privacy leakage from logits. Each digit represents an image sample. Row 1, 3, and 5 respectively display raw MNIST images labeled as '0', '1', and '6'. Row 2, 4, and 6 respectively display simulated images generated by the server-side generator corresponding to label '0', '1', and '6'. We observe that simulated images and raw images with the same label bear high similarity.

reveal certain features of the raw data. In some cases, these features are of crucial importance to users, potentially enabling adversaries to deduce more sensitive information.

Fig. 6. Privacy leakage from logits. Each digit represents an image sample. Row 1, 3, and 5 respectively display raw Fashion-MNIST images labeled as '0', '3', and '9'. Row 2, 4, and 6 respectively display simulated images generated by the server-side generator corresponding to label '0', '3', and '9'. We observe that simulated images and raw images with the same label bear high similarity.

Fig. 7. Privacy leakage from logits. Each digit represents an image sample. The first row shows raw CelebA images. The second row shows simulated images reconstructed by the server.

Experiments on CelebA. Figure 7 demonstrates the experimental results on CelebA dataset. Different from experiments on MNIST and Fashion-MNIST, in CelebA experiments, the server does not have a reliable public dataset and uses dummy

samples as public dataset samples during the training. Besides, each client trains a binary classification model for the face recognition task. From the results, we conclude that, the server is able to steal private information about the target client's data even when it does not have available public datasets. In addition, the generator without any information can reveal certain features of face images. For people who place a high value on the privacy of their personal identity, even partial information can evoke fear.

Similarity Measure. Table II displays the similarity between raw data and generated data. Specifically, as in the literature [38], we calculate the cosine similarity between intermediate features of raw and generated images. These features are extracted by VGGNet-16. Additionally, we utilize randomly generated noisy images as a baseline. Note that other metrics such as SSIM are not applicable in this work. Experimental results unmistakably demonstrate a substantial similarity between the generated and private images, which quantitatively proves the hidden danger of L-Attack.

TABLE II
SIMILARITY MEASURE (LARGER IS BETTER)

samples	MNIST	Fashion-MNIST	CelebA
Noisy images **vs.** Raw images	-0.623	-0.529	-0.333
Generated images **vs.** Raw images	**0.962**	**0.923**	**0.948**

Attacks' Effectiveness in Training with Defense. L-Attack achieves effective attack results in training under simple defense settings. Based on the experimental setup regarding MNIST, we add noise to logits so as to defend against L-Attack. The noise is generated based on Laplace distribution. Figure 8 demonstrates results. Interestingly, L-Attack can effectively obtain private information. Although attacks by L-Attack may not always succeed under this experimental setting, the experimental results indicate that defending against L-Attack requires superior defense methods.

(a) Generated images related to '0' (b) Generated images related to '1'

Fig. 8. Data reconstruction from perturbed logits

Summary. In practice, fully mimicking the raw data proves challenging for L-Attack. Despite the limitations of L-Attack, the aforementioned risks must be considered for security-sensitive FD applications. Even if the generated samples only reveal certain features of the raw data, they can still pose a significant privacy risk to users in specific scenarios. For instance, consider a user whose mobile phone contributes to a logits-based FD system used to train a photo-enhanced model. In such a case, the server could employ L-Attack to extract scene features from the user's home photos, potentially exposing the user's address (real-world instances of such privacy breaches do exist). Moreover, if the user's phone contains private nude photos, even simulated images created by an attacker can evoke fear in the user [39], despite these images containing only partial information.

B. LabelAvg Evaluation

Experimental Settings. We build an FD system with different physical machines acting as clients/server. All approaches are implemented by PyTorch[4]. Each result is the average obtained from 3 experiments under different random seeds.

1) Datasets. Several well-established datasets are used in our experiments, including MNIST, FEMNIST, CIFAR-10, CINIC-10, and CIFAR-100. To demonstrate the generality of the proposed algorithm, the following several groups of experiments are conducted under different popular Non-IID settings, including the Dirichlet distribution [40] setting and the label distribution skew introduced in the Google research's literature [2].
2) Baselines. Five representative approaches are chosen as baselines: FedMD, FedMD-LDP, FedED, FedKD [10], and One-sided Training. In particular, FedMD-LDP [9] is a typical FD that perturbs logits by privacy differences and One-sided Training is the standalone training without FL settings. Besides, in some experiments, we also use FedMD with sparse logits as the baseline.
3) Models. Five representative DNNs including ResNet-18, ResNet-34, VGGNet-16, CNN, and MLP are trained via the above-mentioned approaches. The convolution filter in the 3-layer CNN has a shape of (5x5). The hidden layer dimension of MLP is set to 200. Specifically, under the heterogeneous setting, 1/3 of clients train VGGNet-16, 1/3 of clients train ResNet-18, and the rest train VGGNet-19. Specifically, under the heterogeneous setting, 1/3 of clients train VGGNet-16, 1/3 of clients train ResNet-18, and the rest train VGGNet-19.
4) Hyperparameters. Hyperparameters are tailored to datasets. E.g., the configurations regarding FMNIST and CIFAR-100 experiments align with those in the literature [5]. Here, we specifically elaborate on the hyperparameter configurations for experiments on CIFAR-10. Hyperparameter settings are listed in Table III. In experiments on CIFAR-10, we use Exp. I and Exp. II to refer to the experiments where CINIC-10 and CIFAR-100 serve as the public dataset, respectively. Due to varying knowledge transfer complexities across different public datasets, there are slight differences in hyperparameter settings between the two groups of CIFAR-10 experiments.

TABLE III
SETINGS REGARDING EXPERIMENTS ON CIFAR-10

Hyperparameter	Value
The number of clients	10
The number of epochs	100 for Exp. I and 200 for Exp. II.
The number of local epochs	2
The number of distillation epochs	1
The number of pre-trained epochs	20
The size of the mini-batch in local training	128
The size of the mini-batch in distillation	128 for Exp. I and 256 for Exp. II.
The size of the mini-batch in pre-training	128
The learning rate	0.1
The weight decay	1e-5

[4]https://pytorch.org

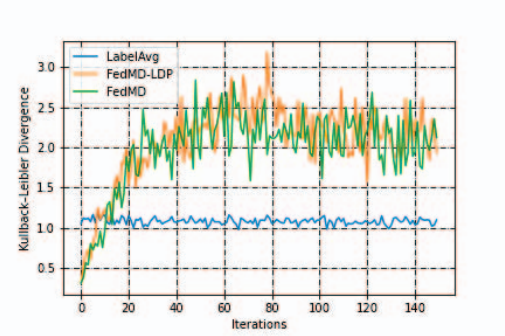


Fig. 9. KL divergence comparison under different Non-IID settings (regarding σ). We measure the KL divergence between the ground-truth distribution and the distributions generated by FedMD, FedMD-LDP, and LabelAvg, respectively. Lower is better. The y-axis represents the KL divergence. As the number of iterations increases, KL divergence between the ground-truth distribution and ours remains consistently low.

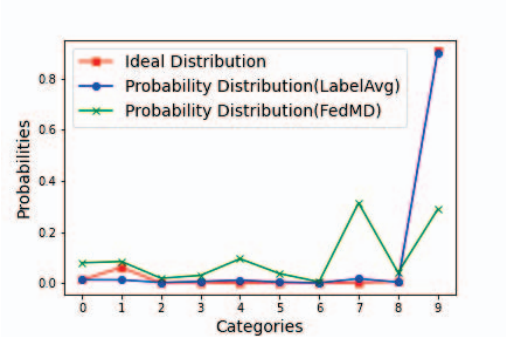


Fig. 10. Label information generated by different methods concerning several samples (one sample per subfigure). The x-axis represents categories, and the y-axis represents values(i.e., probabilities). The red, blue, and green lines correspond to labels generated from the ideal model, LabelAvg, and FedMD, respectively. Also, this group of figures can be regarded as a probability distribution comparison.

Evaluation of Smoothed Labels' Effectiveness. To analyze the distribution indicated by smoothed labels, we compare it with the probability distribution indicated by logits from an ideal teacher model. In particular, we assume there exists an ideal teacher model that possesses knowledge from all private and public data, and the probabilities distribution indicated in its logits aligns with the ground-truth distribution. In this work, we consider the converged CNN trained on CIFAR-10 (private data) and CINIC-10[5](public data) as an ideal teacher model. KL divergence analysis is shown in Fig. 9. Compared with FedMD and FedMD-LDP, the probability distribution obtained via LabelAvg is closer to that obtained via the ideal model, which may explain why client-side models in LabelAvg achieve better performance. We also show the label information generated from LabelAvg and FedMD concerning several samples in Figure 10. Moreover, the depicted ideal distribution is derived from the same ideal model setting mentioned earlier. Note that, the value in each column of smoothed labels and logits (vector form) represents the predicted probability for each class. From Figure 10, we observe that, with respect to some non-target categories, the differences between LabelAvg and the output of the ideal model are negligibly small. However, probabilities of those non-target categories under FedMD setting may be assigned larger values, which may disturb the discriminative ability of the trained model. E.g., category 0, 4, and 7 in Figure 10.

Comparison with Different 'K' Settings. Since LabelAvg transmits 'hard' labels and corresponding weights with the server instead of logits, it can naturally avoid attacks from logits. However, different numbers of transmitted labels (regarding K) may affect the accuracy of trained models. To evaluate the effect, we conduct experiments on FEMNIST and CIFAR-100 with different K settings, respectively. Cases 1- 10 in Table IV record experimental results. From the results, we conclude that LabelAvg can achieve comparable accuracy under different K settings. In general, the difference ranges from 1% to 3%. However, experiments regarding different datasets exhibit varying effectiveness under different K settings. Therefore, we configure the K values according to datasets in the following experiments so as to achieve the best performance. Note that we only configure K from 2 to 6, as configuring K to larger values may not reduce the communication data volume for LabelAvg.

Comparison under Homogeneous Settings. We conduct four groups of experiments with different settings: 1) training CNN on FEMNIST (MNIST as a public dataset) under the Non-IID setting introduced in the FedMD's literature [5], 2) training VGGNet-16 on CIFAR-10 (CIFAR-100 as a public dataset) under the Non-IID setting introduced in the literature [2], 3) training VGGNet-16 on CIFAR-10 (CINIC-10 as a public dataset) under the Dirichlet Non-IID setting, and 4) training CNN on CIFAR-100 (CIFAR-10 as a public dataset) under the Non-IID settings introduced in the literature [5]. Experimental results are shown in Table IV. Specifically, unlike other baselines, the purpose of FedED and FedKD is to train a global model. Therefore, we focus on the accuracy of the global model (recorded in the average column of Table IV). In general, LabelAvg is superior to baselines in terms of maximum, minimum, and average performance on all datasets under different Non-IID settings (Cases 11-34). In particular, LabelAvg can improve accuracy by 2-8% compared to one-sided training. From the results, we observe that the accuracy difference between LabelAvg and FedMD is smaller. However, LabelAvg can resist L-Attack while reducing communication volume based on configurations. Also, although in some cases the differences between LabelAvg, FedED, and FedKD are minimal, LabelAvg has the advantage of supporting personalized training. From the results in Table IV, we can observe that although LabelAvg outperforms baselines overall, the accuracy of all algorithms is generally low. We believe one of the main reasons contributes to the observed results: we set strict criteria for both private and public dataset configurations, including sample size and categories, so as to make the distribution of data more aligned with real-world applications [5]. As a result, the training task becomes more challenging, which leads to a decrease in accuracy. In particular, on one hand, the number of

[5]https://github.com/BayesWatch/cinic-10

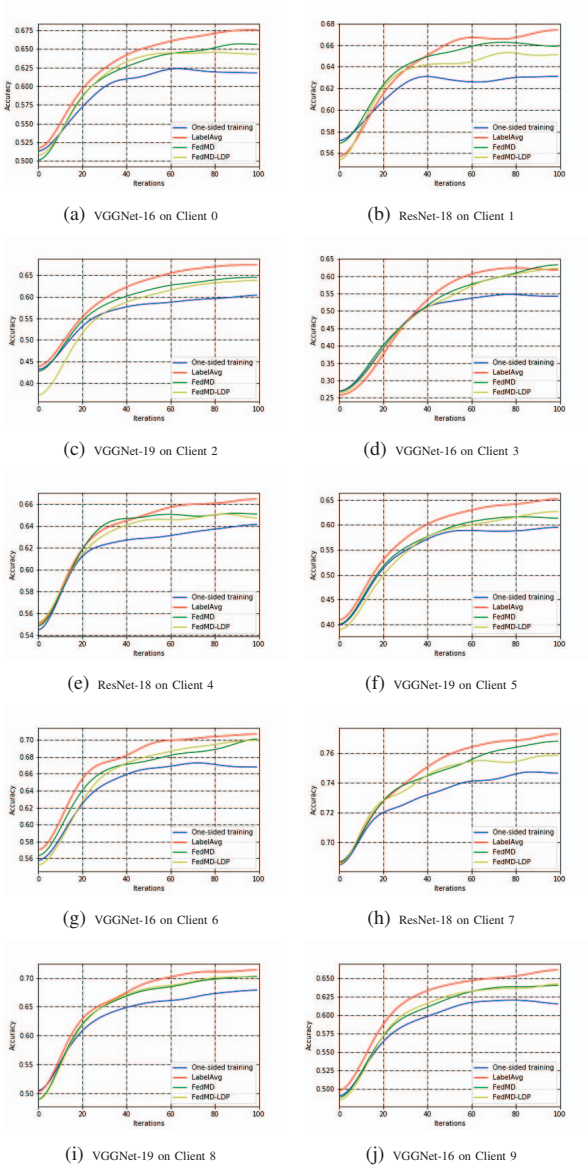
TABLE IV

ACCURACY (%) COMPARISON UNDER DIFFERENT SETTINGS. THE MAXIMUM, MINIMUM, AND AVERAGE VALUES RECORD THE BEST ACCURACY, THE WORST ACCURACY, AND THE AVERAGE ACCURACY OF LOCAL MODELS AMONG CLIENTS, RESPECTIVELY.

Cases	Datasets	Public Datasets	Models	Baselines	K	Maximum	Minimum	Average
Comparison with different K								
1	FEMNIST	MNIST	CNN	LabelAvg	2	70.58	54.20	62.84
2					3	74.06	53.75	61.55
3					4	**81.43**	**56.96**	**64.62**
4					5	74.85	54.71	63.87
5					6	78.29	54.87	64.60
6	CIFAR-100	CIFAR-10	CNN	LabelAvg	2	**48.50**	40.40	**43.07**
7					3	47.97	**41.97**	42.91
8					4	47.20	39.43	42.42
9					5	47.80	41.89	42.29
10					6	46.96	40.53	42.17
Comparison under homogeneous settings								
11	FEMNIST	MNIST	CNN	One-sided Training	-	66.46	47.06	55.82
12				FedMD	-	71.43	51.69	63.17
13				FedMD-LDP	-	71.06	52.37	62.13
14				FedED	-	-	-	58.91
15				FedKD	-	-	-	59.33
16				LabelAvg	4	**81.43**	**56.96**	**64.62**
17	CIFAR-10 ($\sigma = 0.5$)	CINIC-10	VGGNet-16	One-sided Training	-	72.49	44.92	56.00
18				FedMD	-	73.57	53.83	60.17
19				FedMD-LDP	-	72.18	53.33	56.22
20				FedED	-	-	-	60.38
21				FedKD	-	-	-	60.57
22				LabelAvg	5	**74.08**	**54.33**	**60.77**
23	CIFAR-10	CIFAR-100	VGGNet-16	One-sided Training	-	77.91	73.29	74.71
24				FedMD	-	78.14	74.69	75.72
25				FedMD-LDP	-	77.42	75.16	75.33
26				FedED	-	-	-	75.86
27				FedKD	-	-	-	75.9
28				LabelAvg	5	**78.33**	**75.79**	**76.03**
29	CIFAR-100	CIFAR-10	CNN	One-sided Training	-	44.96	34.99	38.78
30				FedMD	-	46.77	40.40	42.2
31				FedMD-LDP	-	47.19	40.40	41.20
32				FedED	-	-	-	40.13
33				FedKD	-	-	-	41.7
34				LabelAvg	3	**47.97**	**41.97**	**42.91**
Comparison under heterogeneous settings								
35	CIFAR-10 ($\sigma = 0.1$)	CINIC-10	VGGNet-16 ResNet- 18 VGGNet-19	One-sided Training	-	56.69	18.79	32.29
36				FedMD	-	56.62	18.31	32.34
37				FedMD-LDP	-	56.26	18.34	31.89
38				FedED	-	-	-	33.5
39				FedKD	-	-	-	34.01
40				LabelAvg	5	**58.81**	**19.10**	**34.25**
41	CIFAR-10 ($\sigma = 0.5$)	CINIC-10	VGGNet-16 ResNet-18 VGGNet-19	One-sided Training	-	73.15	45.76	55.89
42				FedMD	-	72.56	49.67	57.58
43				FedMD-LDP	-	73.57	50.61	57.52
44				FedED	-	-	-	58.33
45				FedKD	-	-	-	58.9
46				LabelAvg	5	**73.89**	**52.53**	**59.37**
47	CIFAR-10 ($\sigma = 0.9$)	CINIC-10	VGGNet-16 ResNet-18 VGGNet-19	One-sided Training	-	73.15	45.76	64.73
48				FedMD	-	72.56	49.67	66.91
49				FedMD-LDP	-	73.57	50.60	66.46
50				FedED	-	-	-	67.93
51				FedKD	-	-	-	67.33
52				LabelAvg	5	**73.89**	**52.53**	**68.01**

samples in the public dataset is limited. Our setting aligns with real-world scenarios because suitable public dataset samples are always limited in quantity. E.g., in two experiments regarding CIFAR-10, the number of samples in the public datasets (CIFAR-100 and CINIC-10) is only 3000. On the other hand, as in literature [5], in the experiments regarding CIFAR-100, we utilize superclasses (20 in total) instead of subclasses (100 in total) as data labels during training. However, we predict the subclasses of test samples during testing. Such evaluation effectively showcases the knowledge transfer effects in FD.

Comparison under Heterogeneous Settings. Theoretically, LabelAvg is the model-agnostic training approach, which means it can perform well under both homogeneous and heterogeneous settings. We conduct several groups of experiments on CIFAR-10 (CINIC-10 as the public dataset) under different Dirichlet distribution settings (α is set to 0.1, 0.5, and 0.9, respectively). To simplify the experimental design, we configure K as 5 in these experiments. Experimental results are listed in Table IV (Cases 35-52). All experimental results indicate that LabelAvg outperforms baselines. We demonstrate convergence regarding different local models in Figure 11. In FedED and FedKD, the global model is configured as VGGNet-16. Besides, curves for FedED and FedKD are not depicted in Figure 11, since these two algorithms naturally focus on the performance of the global model instead of client-side local models. Due to the different data distributions among each client, the same algorithm or model may exhibit variations across different clients. As a result, LabelAvg's performance may not be significant for individual client. However, in general, except for Client 3 in Figure 11(d), LabelAvg outperforms the baselines across most clients.



(a) VGGNet-16 on Client 0 (b) ResNet-18 on Client 1

(c) VGGNet-19 on Client 2 (d) VGGNet-16 on Client 3

(e) ResNet-18 on Client 4 (f) VGGNet-19 on Client 5

(g) VGGNet-16 on Client 6 (h) ResNet-18 on Client 7

(i) VGGNet-19 on Client 8 (j) VGGNet-16 on Client 9

Fig. 11. Performance of all client-sided models on CIFAR-10 under heterogeneous settings (α=0.9). The x-axis represents epochs while the y-axis represents accuracy. The red line represents LabelAvg. All curves are smooth.

Communication Data Volume Comparison. In LabelAvg implementation, K is a hyperparameter used to configure the number of labels transmitted. In each communication round(i.e., epoch), for each public data, each client needs to send K pairs of weight-label and receive one smoothed label. Thus, the smaller K, the less communication data volume. Table V lists data volume exchanged during federated

training via FedMD and LabelAvg, respectively. Compared with FedMD, LabelAvg can effectively reduce the data volume under the same setting (i.e., the number of public data samples and communication rounds). Note that, FedMD and FedMD-LDP send the same data type, hence their communication volumes are similar.

TABLE V
COMMUNICATION DATA VOLUME COMPARISON

Cases	The number of public data samples	The number of types of labels	Epochs	Algorithms	Data Volume (MB)
FEMNIST	120 samples of MNIST	16,including 6 types of public dataset'labels	50	FedMD	≈ 7.324
				FedMD-LDP	≈ 7.324
				LabelAvg(K=2)	≈ 4.577
				LabelAvg(K=3)	≈ 5.035
				LabelAvg(K=4)	≈ 5.493
				LabelAvg(K=5)	≈ 5.950
CIFAR-10	3000 samples of CIFAR-100	16,including 6 types of public dataset'labels	200	FedMD	≈ 732.42
				FedMD-LDP	≈ 732.42
				LabelAvg(K=2)	≈ 457.76
				LabelAvg(K=3)	≈ 503.54
				LabelAvg(K=4)	≈ 549.32
				LabelAvg(K=5)	≈ 595.09

Computational Overhead Comparison. As in many studies, we evaluate the computational overhead of algorithms by measuring GPU execution time (ms) in the same hardware environment and configurations. In particular, the differences in the computational overhead of algorithms primarily stem from how the clients handle logits and the number of public samples. Therefore, we compare the runtime of a single client regarding VGGNet-16 inference on a public dataset (with sample sizes of 1000, 3000, and 5000, respectively) across different baselines. The results are shown in Figure 12, indicating that apart from the sparsify method, the computational overhead among other algorithms is not significantly different. Note that, we do not compare FedED and FedKD because these two methods handle logits in the same way as FedMD.

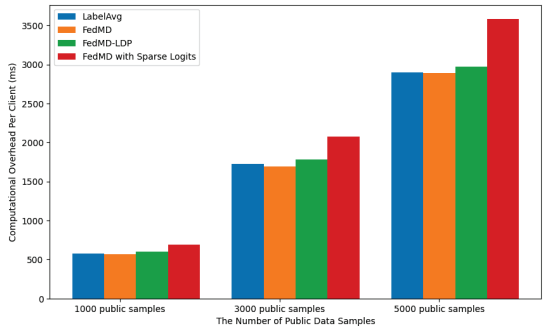


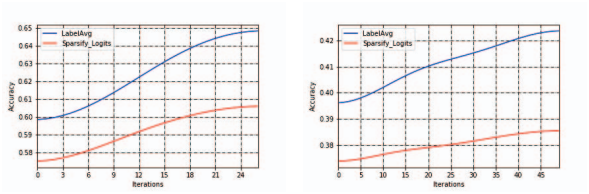
Fig. 12. Computational Overhead Comparison. Lower is better. Except for FedMD with sparse logits (red bar), the proposed method's execution time (blue bar) is close to the baseline.

The Ability to Defend L-Attack. We adhere to the MNIST experiment settings outlined in Section VI-A. We utilize smoothed labels to estimate gradients, which are then used to update the generator for data reconstruction. Experimental results are demonstrated in Figure 13. Clearly, L-attack is unable to effectively extract data information. According to the results of the above-mentioned gradient visualization experiments, we believe that the estimated gradients based on smoothed labels exhibit errors compared to real gradients. Indeed, we have conducted experiments on additional datasets, and the attacker's generator fails to generate simulated images that reveal sensitive information across all experimental results.

(a) (b) (c) (d) (e)

Fig. 13. Data reconstruction from smoothed labels

Smoothed Labels vs. Sparse Logits. In previous section, we discussed the advantages of LabelAvg compared to sparsify logits. In particular, we conducted experiments on FEMNIST and CIFAR-10 following the settings mentioned above. The experimental results are shown in Figure 14. For simplicity, in this group of experiments, we refer to FedMD with sparse logits as sparsify logits. We observe that, within the same communication round, sparsify logits fail to reach a satisfactory convergence point, as evidenced by the red line in Figure 14, whereas LabelAvg achieves high accuracy on both CIFAR-10 and FEMNIST. Compared with LabelAvg, sparsify logits result in a greater loss of accuracy in the trained model.



(a) Experiments on FEMNIST (b) Experiments on CIFAR-100

Fig. 14. Comparison between LabelAvg and FedMD with sparse logits.

VII. CONCLUSION

In this paper, we design an attack called L-Attack which reveals the risk from logits related to public datasets in FD. Also, we propose LabelAvg, a novel FD algorithm that can avoid L-Attack by transmitting weighted labels instead of logits. Experimental results prove LabelAvg is superior to baselines in terms of accuracy and communication overhead.

Moving forward, we aim to explore the following aspects in our future work:

- We will further explore attacks against logits. For example, we try to investigate other generative models, such as diffusion models, to reconstruct private data information from logits.
- We will explore new defense techniques against L-Attack. For instance, we aim to enhance the performance of FD algorithms based on logits sparsification, with the goal of achieving higher accuracy in trained models when sparsifying logits during training.

VIII. ACKNOWLEDGEMENT

Weigang Wu is the corresponding author. This research is partially supported by National Natural Science Foundation of China (No. 62372487).

REFERENCES

[1] Q. Li, B. He, and D. Song, "Practical one-shot federated learning for cross-silo setting," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*, 2021, pp. 1484–1490.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, (AISTATS 2017)*, 2017.

[3] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (NeurIPS 2020)*, 2020, pp. 2351–2363.

[4] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, "FedED: Federated learning via ensemble distillation for medical relation extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, (EMNLP 2020)*, 2020, pp. 2118–2128.

[5] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," *CoRR*, vol. abs/1910.03581, 2019. [Online]. Available: https://arxiv.org/abs/1910.03581

[6] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *2020 USENIX Annual Technical Conference, (USENIX ATC 2020)*, 2020, pp. 493–506.

[7] B. Hitaj, G. Ateniese, and Fernando, "Deep models under the gan: nformation leakage from collaborative deep learning," in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security, (CCS 2017)*, 2017, pp. 603–618.

[8] Y. Yang, H. Yuan, B. Hui, N. Z. Gong, N. Fendley, P. Burlina, and Y. Cao, "Fortifying federated learning against membership inference attacks via client-level input perturbation," in *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, (DSN 2023)*, 2023, pp. 288–301.

[9] L. Sun and L. Lyu, "Federated model distillation with noise-free differential privacy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, (IJCAI 2021)*, 2021, pp. 1563–1570.

[10] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. S. Doermann, and A. Innanje, "Preserving privacy in federated learning with ensemble cross-domain knowledge distillation," in *The Thirty-Sixth AAAI Conference on Artificial Intelligence, (AAAI 2022)*, 2022, pp. 11 891–11 899.

[11] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," in *6th International Conference on Learning Representations, (ICLR 2018)*, 2018.

[12] S. Salehkaleybar, A. Sharif-Nassab, and S. J. Golestani, "One-shot federated learning: Theoretical limits and algorithms to achieve them," *Journal of Machine Learning Research*, vol. 22, no. 189, pp. 1–47, 2021.

[13] C. Wu, F. Wu, R. Liu, L. Lyu, Y. Huang, and X. Xie, "Fedkd: Communication efficient federated learning via knowledge distillation," *Nature Communications*, vol. 13, no. 2032, pp. 1–8, 2022.

[14] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *CoRR*, vol. abs/1811.11479, 2018. [Online]. Available: https://arxiv.org/abs/1811.11479

[15] C. Li, G. Li, and P. K. Varshney, "Decentralized federated learning via mutual knowledge transfer," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1136–1147, 2022.

[16] A. Taya, T. Nishio, M. Morikura, and K. Yamamoto, "Decentralized and model-free federated learning: Consensus-based distillation in function space," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, no. 1, pp. 799–814, 2022.

[17] J. Huang, Z. Zhao, L. Y. Chen, and S. Roos, "Fabricated flips: Poisoning federated learning without data," in *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, (DSN 2023)*, 2023, pp. 274–287.

[18] Y. Yu, Q. Liu, L. Wu, R. Yu, S. L. Yu, and Z. Zhang, "Untargeted attack against federated recommendation systems via poisonous item embeddings and the defense," in *Thirty-Seventh AAAI Conference on Artificial Intelligence, (AAAI 2023)*, 2023.

[19] G. Chen, X. Zhang, Y. Su, Y. Lai, J. Xiang, J. Zhang, and Y. Zheng, "Win-win: A privacy-preserving federated framework for dual-target cross-domain recommendation," in *Thirty-Seventh AAAI Conference on Artificial Intelligence,(AAAI 2023)*, 2023.

[20] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, (CCS 2015)*, 2015, pp. 1322–1333.

[21] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in *IEEE Conference on Computer Communications, (INFOCOM 2022)*, 2022, pp. 580–589.

[22] W. Wei, L. Liu, Y. Wu, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *41st IEEE International Conference on Distributed Computing Systems,(ICDCS 2021)*, 2021, pp. 797–807.

[23] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, (NeurIPS 2019)*, 2019, pp. 14 747–14 756.

[24] X. He, C. Peng, and W. Tan, "Fast and accurate deep leakage from gradients based on wasserstein distance," *International Journal of Intelligent Systems*, vol. 2023, no. 1, pp. 1–12, 2023.

[25] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in *IEEE Conference on Computer Communications, (INFOCOM 2022)*, 2022, pp. 580–589.

[26] F. Wang, E. Hugh, and B. Li, "More than enough is too much: Adaptive defenses against gradient leakage in production federated learning," in *IEEE Conference on Computer Communications, (INFOCOM 2023)*, 2023, pp. 1–10.

[27] S. H. Na, H. G. Hong, J. Kim, and S. Shin, "Closing the loophole: Rethinking reconstruction attacks in federated learning from a privacy standpoint," in *Annual Computer Security Applications Conference, (ACSAC 2022)*, 2022, pp. 332–345.

[28] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2021)*, 2021, pp. 9311–9319.

[29] Y. Koda, J. Park, M. Bennis, K. Yamamoto, T. Nishio, and M. Morikura, "One pixel image and rf signal based split learning for mmwave received power prediction," in *Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 19)*, 2019, pp. 54–56.

[30] V. Turina, Z. Zhang, F. Esposito, and I. Matta, "Combining split and federated architectures for efficiency and privacy in deep learning," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 20)*. Association for Computing Machinery, 2020, pp. 562–563.

[31] D. Xiao, C. Yang, and W. Wu, "Mixing activations and labels in distributed training for split learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 3165–3177, 2022.

[32] D. Pasquini, G. Ateniese, and M. Bernaschi, "Unleashing the tiger: Inference attacks on split learning," in *Proceedings of the 24th ACM SIGSAC Conference on Computer and Communications Security, (CCS 2021)*, 2021, pp. 2113–2129.

[33] S. Kariyappa, A. Prakash, and M. K. Qureshi, "MAZE: data-free model stealing attack using zeroth-order gradient estimation," in *Proceedings of 2021 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2021)*, 2021, pp. 13 814–13 823.

[34] J. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of 2021 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2021)*, 2021, pp. 4771–4780.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016.

[36] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019 (NeurIPS 2019)*, 2019.

[37] C. Zhang, P. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M. Cheng, "Delving deep into label smoothing," *IEEE Transactions on Image Processing*, vol. 30, no. 2021, pp. 5984–5996, 2021.

[38] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *CoRR*, vol. abs/1306.6709, 2013. [Online]. Available: http://arxiv.org/abs/1306.6709

[39] J. Lou, X. Zhang, Y. Zhang, X. Li, X. Yuan, and N. Zhang, "Devils in your apps: Vulnerabilities and user privacy exposure in mobile notification systems," in *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, (DSN 2023)*, 2023, pp. 28–41.

[40] T. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *CoRR*, vol. abs/1909.06335, 2019. [Online]. Available: http://arxiv.org/abs/1909.06335