Catastrophic Data Leakage in Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent studies show that private training data can be leaked through the gradient sharing mechanism deployed in machine learning systems, such as federated learning (FL). Increasing batch size to complicate data recovery is often viewed as a promising defense strategy against data leakage. In this paper, we revisit this defense premise and propose an advanced data leakage attack with theoretical justification to efficiently recover batch data from the shared aggregated gradients. We name our proposed method as *catastrophic data leakage in federated learning* (CAFE). Comparing to existing data leakage attacks, our extensive experimental results on vertical FL settings demonstrate the effectiveness of CAFE to perform large-batch data leakage attack with improved data recovery quality. We also propose a practical countermeasure to mitigate CAFE. Our results suggest that private data participated in standard FL, especially the vertical case, have a high risk of being leaked from the training gradients. Our analysis implies unprecedented and practical data leakage risks in those learning settings.

1 Introduction

Federated learning (FL) [7, 21] is an emerging machine learning framework where a central server and multiple workers collaboratively train a machine learning model. Some existing FL methods consider the setting where each worker has data of a different set of subjects but sharing common features. This setting is also referred to data partitioned or horizontal FL (HFL). Unlike the HFL setting, in many learning scenarios, multiple workers handle data about the same set of subjects, but each has a different set of features. This case is common in finance and healthcare applications [5]. In these examples, data owners (e.g., financial institutions and hospitals) have different records of those users in their joint user base, and so, by combining their features through FL, they can establish a more accurate model. We refer to this setting as feature-partitioned or vertical FL (VFL).

Compared with existing distributed learning paradigms, FL raises new challenges including data heterogeneity and privacy [18]. To protect data privacy, only model parameters and the change of parameters (e.g., gradients) are exchanged between server and workers [17, 13]. Recent works have studied how a malicious worker can embed backdoors or replace the global model in FL [1, 2, 24]. Furthermore, as exchanging gradients is often viewed as privacy-preserving protocols, little attention has been paid to information leakage from public shared gradients and batch identities.

In the context of data security and AI ethics, the possibility of inferring private user data from the gradients in FL has received growing interests [9, 12, 19], known as the *data leakage* problems. Previous works have made some explorations on data recovery through gradients. See Section 2 and Table 2 for details. However, existing approaches often have the limitation of scaling up large-batch data recovery and lack theoretical justification on the capability of data recovery, which may give a false sense of security that increasing the data batch size during training can prevent data leakage [27]. Some recent works provide sufficient conditions for guaranteed data recovery, but the assumptions

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

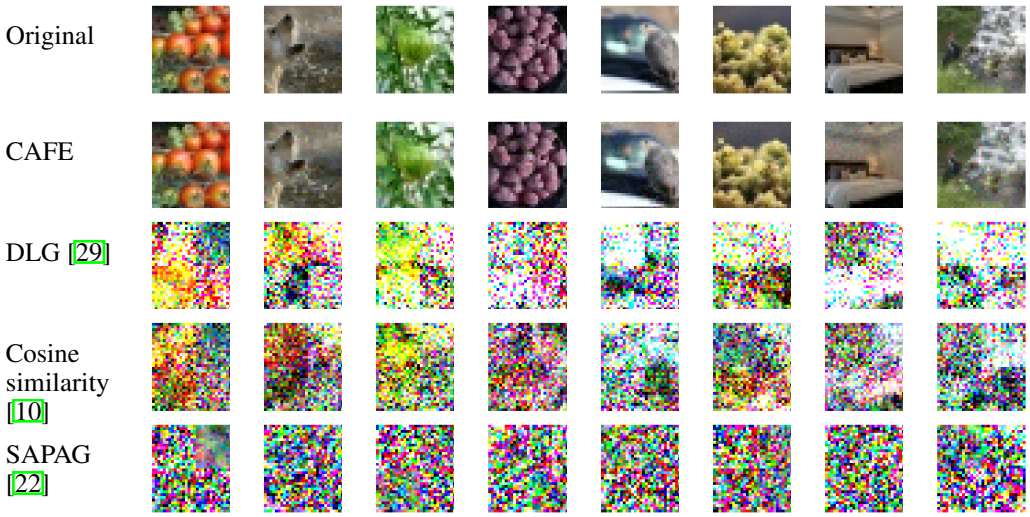


Figure 1: Visual comparison between CAFE (our proposed method) with the state-of-the-art data leakage attacks on Linnaeus-5 in VFL (batch size = 40 and batch ratio = 0.05).

are overly restrictive and can be impractical, such as requiring the number of classes to be much larger than the number of recovered data samples [26].

To address the limitation of scalability in data recovery and gain fundamental understanding on data leakage in FL, in this paper we propose an advanced data leakage attack with theoretical analysis on the data recovery performance, which we call *catastrophic data leakage in federated learning* (CAFE). As an illustration, Figure 1 demonstrates the effectiveness of CAFE for large-batch data recovery compared to existing methods. The contributions of this paper are summarized as follows.

1. We develop an advanced data leakage attack named CAFE to overcome the limitation of current data leakage attacks on FL. CAFE is able to recover large-scale data in general VFL protocols.
2. CAFE consists of three new designs for efficient data recovery: (I) Recover gradients of loss with respect to (w.r.t.) the outputs of the first fully connected (FC) layer in the model. (II) Recover inputs of the first FC layer. (III) Recover the original data. Its capability of large-batch data recovery is based on the novel use of data index and internal representation alignments in FL.
3. We establish a theoretical analysis on the algorithmic recovery performance of CAFE.
4. Extensive experiments on both static and dynamic FL training settings are used to validate the superior data recovery performance of CAFE over state-of-the-art methods.
5. We also propose an effective countermeasure to mitigate our CAFE attack algorithm.

2 Related Work

A popular method that is termed deep leakage from gradients (DLG) [29] infers training data in an efficient way without using any generative models or prior information. However, DLG lacks generalizability on model architecture and weight distribution initialization [22]. In [27], an analytical approach has been developed to extract accurate labels from the gradients. In [10], another analytical approach has been developed to derive the input before a fully connected (FC) layer. However, in [10], their method only works on a single sample input and fails to extend on a batch of data. In [20], the authors try to obtain the batch inputs before the FC layer through solving linear equations. However, they make strong assumptions for solving the equations and cannot guarantee data recovery in more general cases. In [8], the authors claim that a convolutional layer can always be converted to a FC layer. However, the gradients of the original convolutional layer are still different from the gradients of the converted FC layer, which impedes data recovery. Besides the new loss function proposed in [10], several previous works design new loss functions or regularizers based on DLG and try to make their algorithms work on more general models and weight distribution initialization. In [22], the authors use a Gaussian kernel based gradient difference as the distance measure. In [28],

Table 1: Comparison of CAFE with state-of-the-art data leakage attack methods in FL.

Method	Optimization terms	Reported maximal batch size	Training while attacking	Theoretical guarantee	Additional information other than gradients
DLG [29]	ℓ_2 distance between real and fake gradients	8	No	No	No
iDLG [27]	ℓ_2 distance	8	No	Yes	No
Inverting Gradients [10]	Cosine similarity, TV norm	8 100 (Mostly unrecognizable)	Yes	Yes	Number of local updates
A Framework for Evaluating Gradient Leakage [23]	ℓ_2 distance, label based regualrizer	8	No	Yes	No
SAPAG [22]	Gaussian kernel based funciton	8	No	No	No
R-GAP [28]	recursive gradient loss	5	No	Yes	The rank of matrix A defined in [28]
Theory oriented [20]	ℓ_2 distance, ℓ_1 distances of the recovered feature map	32	No	Yes	Number of Exclusive activated neurons
GradInversion[26]	Fidelity regularizers, Group consistency regularizers	48	No	No	Batch size $\ll$ number of classes & Non repeating labels in a batch
CAFE (ours)	ℓ_2 distance, TV norm, Internal representation norm	100 (our hardware limit)	Yes	Yes	Batch indices

the authors use a recursive method attack procedure to recover data from gradients. However, both in [22] and [28], the quality of recovery on batch data is degraded. A recent work [26] proposes an algorithm named GradInversion to reconstruct images from noise based on given gradients. However, their theories and algorithm are mostly built on strong assumptions and empirical observations, and they fail to be extended to more general cases. Although they successfully reconstruct a batch of training data, the reported batch size still can not be larger than 48.

3 CAFE: Catastrophic Data Leakage in Federated Learning

In this section, we will introduce some necessary background of FL and present our novel attack method. We consider the attack scenario where a honest-but-curious server follows the regular FL protocols but intends to recover clients' private data based on the aggregated gradients. Our method is termed *CAFE: Catastrophic data leakage in federated learning*. While CAFE can be applied to any type of data, without loss of generality, we use image datasets throughout the paper.

3.1 Preliminaries

Data in VFL. FL can be categorized into horizontal and vertical FL settings [14]. In this paragraph, we provide necessary background of VFL.

Consider a set of M clients: $\mathcal{M} = \{1, 2, \dots, M\}$. A dataset of N samples $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ are maintained by the M local clients, where n is the data point index. Each client m in $\mathcal{M}$ is associated with a unique features set. A certain data point $\mathbf{x}_n$ in $\mathcal{D}$ can be denoted by $\mathbf{x}_n = [\mathbf{x}_{n,1}^{\mathrm{T}}, \mathbf{x}_{n,2}^{\mathrm{T}}, \dots, \mathbf{x}_{n,M}^{\mathrm{T}}]^{\mathrm{T}}$ where $\mathbf{x}_{n,m}$ is the m-th block of the n-th sample vector. The label space $\{y_n\}_{n=1}^N$ can be regarded as a special feature and is partitioned to the server or a certain local worker. Throughout this paper, we mainly study the VFL setting. CAFE can also be applied to HFL if the data indices in each batch are known.

Random batch indices in each iteration. For a given batch size K, we define a set of binary vectors as $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_i, \dots\}, |\mathcal{S}| = \binom{N}{K}$. For each vector $\mathbf{s}_i \in \mathbb{R}^N$ in $\mathcal{S}$, its n-th element $\mathbf{s}_i[n]$ can be either 0 or 1. There are totally K elements of ones in $\mathbf{s}_i$. The other $N - K$ elements are 0. In each iteration t the server randomly selects one element from set $\mathcal{S}$ denoted by $\mathbf{s}^t$. The selected batch sample indices in the t-th iteration are denoted by $\mathcal{I}(t) = \{n | \mathbf{s}^t[n] = 1\}, |\mathcal{I}(t)| = K$, where $\mathbf{s}^t[n]$ is the nth element in $\mathbf{s}^t$. The selected batch samples in the t-th iteration are denoted by $\mathcal{D}(\mathbf{s}^t) = \{\{\mathbf{x}_i, y_i\} | \mathbf{s}^t[n] = 1\}$. An example of the set of binary vectors $\mathcal{S}$ is shown in Appendix B.

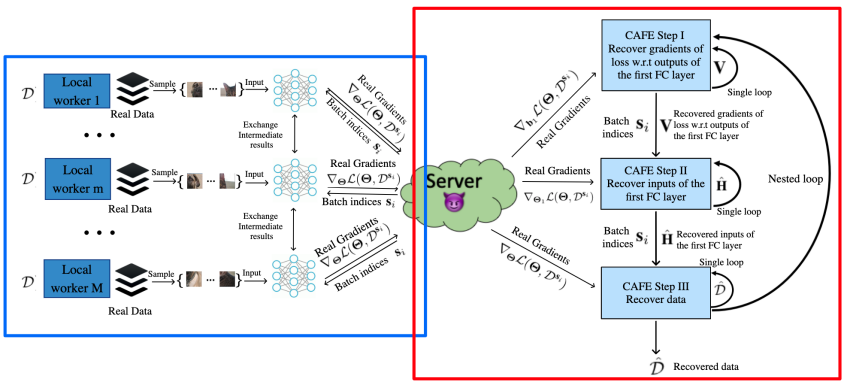


Figure 2: Overview of CAFE. The left part (blue box) performs the regular FL protocol and the right part (red box) illustrates the main steps of CAFE.

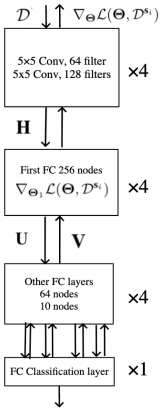


Figure 3: Model structure in VFL.

Loss function and gradients. We assume that the model is a neural network parameterized by $\mathbf{\Theta}$, where the first fully connected layer is parameterized by $\mathbf{\Theta}_1 \in \mathbb{R}^{d_1 \times d_2}$ and its bias is $\mathbf{b}_1 \in \mathbb{R}^{d_2}$. The loss function on the batch data $\mathcal{D}(\mathbf{s}^t)$ and on the entire training data $\mathcal{D}$ is, respectively, denoted by

$$\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) = \frac{1}{K} \sum_{i \in \mathcal{I}(t)} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_i, y_i) \quad \text{and} \quad \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_i, y_i). \tag{1}$$

The gradients of losses w.r.t parameters is denoted by

$$\nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) = \frac{1}{K} \sum_{i \in \mathcal{I}(t)} \frac{\partial \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_i, y_i)}{\partial \mathbf{\Theta}} \in \mathbb{R}^{d_1 \times d_2} \tag{2}$$

and similarly for $\nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$.

3.2 Why Large-batch Data Leakage Attack is Difficult?

We motivate the design of our algorithm by providing some intuition on why performing large-batch data leakage from aggregated gradients is difficult [29]. Assume that K images are selected as the input for a certain learning iteration. We define the selected batch data as $\mathcal{D}' = \{\mathbf{x}_i, y_i\}$. Likewise, the batched 'recovered data' is denoted by $\hat{\mathcal{D}}' = \{\hat{\mathbf{x}}_i, \hat{y}_i\}$. Then the objective function is

$$\hat{\mathcal{D}}' = arg \min_{\{\hat{\mathbf{x}}_i, \hat{y}_i\}} \left\| \frac{1}{K} \sum_{(\mathbf{x}_n, y_n) \in \{\mathbf{x}_i, y_i\}} \nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_n, y_n) - \frac{1}{K} \sum_{(\hat{\mathbf{x}}_n, \hat{y}_n) \in \{\hat{\mathbf{x}}_i, \hat{y}_i\}} \nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \hat{\mathbf{x}}_n, \hat{y}_n) \right\|^2. \tag{3}$$

Note that in (3), the dimensions of the aggregated gradients is fixed. However, as the K increases, the cardinality of $\hat{\mathcal{D}}'$ and $\mathcal{D}'$ rise. When K is sufficiently large, it will be more challenging to find the "right" solution $\hat{\mathcal{D}}'$ of (3) corresponding to the ground-truth dataset $\mathcal{D}'$. On the other hand, CAFE addresses this large-batch issue by data index alignment for batch data recovery, which can effectively exclude undesired solutions. We discuss a specific example in Appendix C.

3.3 CAFE implementation

In this subsection, we explain our algorithm CAFE in detail, which consists of three steps. As shown in Figure 2, CAFE first recovers the gradients of loss with respect to (w.r.t) the output of the first fully connected (FC) layer in the model. Then, it recovers the input of the first FC layer. Finally, it recovers the training data in the last step.

In VFL, the server sends public key to local workers and decides the data indices in each iteration of training and evaluation [6, 25]. During the training process, local workers exchange their intermediate results with others to compute gradients and upload them. Therefore, the server has the access to *both* the model parameters and their gradients. Since data are vertically partitioned among different

workers, *data index alignment* turns out to be an inevitable step in the vertical training process, which provides the server (the attacker) an opportunity to control the selected batch data indices.

Prerequisite: Notably, CAFE can be readily applied to existing VFL protocols where the batch data indices is assigned or other deep learning protocols as long as the batch data indices are given. In Figure 2, the blue part represents a normal VFL paradigm and the red part represents the attack.

Step I: Recover the gradients of loss w.r.t the output of the first FC layer. As shown in Figure 3, for a certain data point $\mathbf{x}_n$, we have the input of the first FC layer as $\mathbf{h}_n = h(\mathbf{\Theta}_c, \mathbf{x}_n) \in \mathbb{R}^{d_1}$ where h is the forward function and $\mathbf{\Theta}_c$ is the parameters before the first FC layer. Let $\mathbf{u}_n$ denote the output of the first FC layer in the neural network, given by

$$\mathbf{u}_n = \mathbf{\Theta}_1^{\mathrm{T}} \mathbf{h}_n + \mathbf{b}_1 \in \mathbb{R}^{d_2}. \tag{4}$$

For the training data $\mathcal{D}$, the corresponding input before the first FC layer is concatenated as $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times d_1}$ and the corresponding output of the first FC layer is concatenated as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]^{\mathrm{T}} \in \mathbb{R}^{N \times d_2}$. The gradients of loss w.r.t $\mathbf{U}$ can be denoted by

$$\begin{aligned} \nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) &= \frac{1}{N} [\nabla_{\mathbf{u}_1} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_1, y_1), \nabla_{\mathbf{u}_2} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_2, y_2), \ldots, \nabla_{\mathbf{u}_N} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_N, y_N)]^{\mathrm{T}} \\ &= \frac{1}{N} \Big[\frac{\partial \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_1, y_1)}{\partial \mathbf{u}_1}, \frac{\partial \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_2, y_2)}{\partial \mathbf{u}_2}, \ldots, \frac{\partial \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_N, y_N)}{\partial \mathbf{u}_N} \Big]^{\mathrm{T}} \in \mathbb{R}^{N \times d_2}. \end{aligned} \tag{5}$$

For a batch of data in the t-th iteration $\mathcal{D}(\mathbf{s}^t)$, we have

$$\nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) = \sum_{n=1}^{N} \mathbf{s}^t[n] \nabla_{\mathbf{u}_n} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)). \tag{6}$$

Although we do not have access to $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$ as gradients are only given w.r.t. the model parameters, we can successfully recover it through the optimization method. We randomly initialize a variable followed by uniform distribution $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_N]^{\mathrm{T}} \sim \mathcal{U}^{N \times d_2}$ where $\mathbf{v}_i = [\mathrm{v}_{\mathrm{i},1}, \mathrm{v}_{\mathrm{i},1}, \ldots, \mathrm{v}_{\mathrm{i},\mathrm{d}_2}]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{d}_2}$.

Given $\nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$, we recover $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$ by minimizing the following objective function

$$\mathbf{V}^* = \arg\min_{\mathbf{V}} \mathcal{F}_1(\mathbf{V}) = \arg\min_{\mathbf{V}} \mathbb{E}_{\mathbf{s}_i \sim \mathbf{Unif}(\mathcal{S})} \left\| \mathbf{V}^{\mathrm{T}} \mathbf{s}_i - \nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)) \right\|_2^2. \tag{7}$$

In each iteration t, the objective function of step I can be denoted by

$$\mathcal{F}_1^t(\mathbf{V}) = \left\| \mathbf{V}^{\mathrm{T}} \mathbf{s}^t - \nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) \right\|_2^2. \tag{8}$$

The first step of CAFE are summarized in Algorithm 1, which enjoys the following guarantee.

Theorem 1. The objective function in (7) is strongly convex in $\mathbf{V}$. For a fixed $\mathbf{\Theta}$, applying SGD to (7) guarantees to converge to the ground truth almost surely.

Step I is essential in CAFE because we separate the gradients of loss w.r.t each single input of the first FC layer from the aggregated gradients in this step.

Step II: Recover inputs of the first FC layer. Using the chain rule we have $\nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) = \mathbf{H}^{\mathrm{T}} \nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) \in \mathbb{R}^{d_1 \times d_2}$. We randomly initialize a variable $\hat{\mathbf{H}} = [\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_i, \ldots, \hat{\mathbf{h}}_N]^{\mathrm{T}}$ following from uniform distribution $\hat{\mathbf{H}} = [\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_i, \ldots, \hat{\mathbf{h}}_N]^{\mathrm{T}} \sim \mathcal{U}^{N \times d_1}$ where $\hat{\mathbf{h}}_i \in \mathbb{R}^{d_1}$. Given $\nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$ and $\mathbf{V}^*$, we recover $\mathbf{H}$ by minimizing the following objective

$$\hat{\mathbf{H}}^* = \arg\min_{\hat{\mathbf{H}}} \mathcal{F}_2(\hat{\mathbf{H}}) = \arg\min_{\hat{\mathbf{H}}} \mathbb{E}_{\mathbf{s}_i \sim \mathbf{Unif}(\mathcal{S})} \Big\| \sum_{n=1}^{N} \mathbf{s}_i[n] \hat{\mathbf{h}}_n (\mathbf{v}_n^*)^{\mathrm{T}} - \nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)) \Big\|_F^2. \tag{9}$$

In each iteration t, the objective function of step II can be denoted by

$$\mathcal{F}_2^t(\hat{\mathbf{H}}) = \Big\| \sum_{n=1}^{N} \mathbf{s}^t[n] \hat{\mathbf{h}}_n \mathbf{v}_n^{*\mathrm{T}} - \nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) \Big\|_F^2. \tag{10}$$

Through the first two steps, parts of the information about the data have already been leaked. In some simple models such as logistic regression or models only containing FC layers, the attack will recover the data only by implementing the first two steps. Step II also has the following guarantee.

Algorithm 1 Recover the gradients $\nabla_{\mathbf{U}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D})$ (regular VFL and attacker)

Given model parameters $\mathbf{\Theta}$ and $\mathbf{V} \sim \mathcal{U}^{N\times d_1}$
for $t = 1, 2, \ldots, T$ **do**
  Server select $\mathbf{s}^t$ from $\mathcal{S}$ and generate $\mathcal{I}(t)$
  Server broadcasts $\mathbf{\Theta}$ and $\mathcal{I}(t)$ to all workers
  for $m = 1, 2, \ldots, M$ **do**
    Worker m takes real batch data
    Worker m exchanges intermediate results with other workers and computes $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$.
    Worker m uploads $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$.
  end for
  Server computes $\nabla_{\mathbf{b}_1}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$
  Server computes $\mathcal{F}_1^t(\mathbf{V})$ in (8)
  Server computes $\nabla_{\mathbf{V}}\mathcal{F}_1^t(\mathbf{V})$
  Server updates $\mathbf{V}$ with $\nabla_{\mathbf{V}}\mathcal{F}_1^t(\mathbf{V})$
end for

Algorithm 2 Recover the input of the first FC layer $\mathbf{H}$ (regular VFL and attacker)

Given a model and its parameters $\mathbf{\Theta}$, trained $\mathbf{V}$, initialize $\hat{\mathbf{H}} \sim \mathcal{U}^{N\times d_2}$
for $t = 1, 2, \ldots, T$ **do**
  Server select $\mathbf{s}^t$ from $\mathcal{S}$ and generate $\mathcal{I}(t)$
  Server broadcasts $\mathbf{\Theta}$ and $\mathcal{I}(t)$ to all workers
  for $m = 1, 2, \ldots, M$ **do**
    Worker m takes real batch data
    Worker m exchanges intermediate results with other workers and computes $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$.
    Worker m uploads $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$.
  end for
  Server computes $\nabla_{\mathbf{\Theta}_1}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$
  Server computes $\mathcal{F}_2^t(\hat{\mathbf{H}})$ in (10)
  Server computes $\nabla_{\hat{\mathbf{H}}}\mathcal{F}_2^t(\hat{\mathbf{H}})$
  Server updates $\hat{\mathbf{H}}$ with $\nabla_{\hat{\mathbf{H}}}\mathcal{F}_2^t(\hat{\mathbf{H}})$
end for

Algorithm 3 CAFE (Nested-loop)

Given model parameters $\mathbf{\Theta}$, initialize $\mathbf{V} \sim \mathcal{U}^{N\times d_1}, \hat{\mathbf{H}} \sim \mathcal{U}^{N\times d_2}, \hat{\mathcal{D}} = \{\hat{\mathbf{x}}_n, \hat{y}_n\}_{n=1}^N$
Run Algorithm 1 for T iterations
Run Algorithm 2 for T iterations
for $t = 1, 2, \ldots, T$ **do**
  Run Step 3-10 in Algorithm 1 once
  Server computes $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$
  Server computes the fake global aggregated gradients $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \hat{\mathcal{D}}(t))$
  Server computes CAFE loss $\mathcal{F}_3^t(\hat{\mathcal{D}})$ in (12)
  Server computes $\nabla_{\hat{\mathcal{D}}}\mathcal{F}_3^t(\hat{\mathcal{D}})$ in (13)
  Server updates $\hat{\mathcal{D}}$ with $\nabla_{\hat{\mathcal{D}}}\mathcal{F}_3^t(\hat{\mathcal{D}})$
end for

Algorithm 4 CAFE (Single-loop)

Given model parameters $\mathbf{\Theta}$, initialize $\mathbf{V} \sim \mathcal{U}^{N\times d_1}, \hat{\mathbf{H}} \sim \mathcal{U}^{N\times d_2}, \hat{\mathcal{D}} = \{\hat{\mathbf{x}}_n, \hat{y}_n\}_{n=1}^N$
for $t = 1, 2, \ldots, T$ **do**
  Run Step 3-10 in Algorithm 1 once
  Server computes $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$ including $\nabla_{\mathbf{b}_1}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)), \nabla_{\mathbf{\Theta}_1}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$
  Run Step 12 - 14 in Algorithm 1 once
  Run Step 9 - 11 in Algorithm 2 once
  Server computes CAFE loss $\mathcal{F}_3^t(\hat{\mathcal{D}})$ in (12)
  Server computes $\nabla_{\hat{\mathcal{D}}}\mathcal{F}_3^t(\hat{\mathcal{D}})$ in (13)
  Server updates $\hat{\mathcal{D}}$ with $\nabla_{\hat{\mathcal{D}}}\mathcal{F}_3^t(\hat{\mathcal{D}})$
end for

Theorem 2. If $N < d_1$ and $Rank(\mathbf{V}^*) = N$, the objective function $\mathcal{F}_2(\mathbf{H})$ is strongly convex. When $\mathbf{\Theta}$ keeps unchanged, applying SGD guarantees convergence.

Step III: Recover data. We randomly initialize the fake data and fake labels followed by uniform distribution $\hat{\mathcal{D}} = \{\hat{\mathbf{x}}_n, \hat{y}_n\}_{n=1}^N$ and $\hat{\mathcal{X}} = \{\hat{\mathbf{x}}_n\}_{n=1}^N$. According to equation (4), we have $\widetilde{\mathbf{h}}_n = h(\mathbf{\Theta}_c, \hat{\mathbf{x}}_n) \in \mathbb{R}^{d_1}$. Given $\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t))$ and $\hat{\mathbf{H}}^*$, our objective function in the last step is

$$\hat{\mathcal{D}}^* = \arg\min_{\hat{\mathcal{D}}} \mathcal{F}_3(\hat{\mathcal{D}}) = \arg\min_{\hat{\mathcal{D}}} \mathbb{E}_{\mathbf{s}_i\sim\mathbf{Unif}(\mathcal{S})}\Big[\alpha\big\|\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)) - \nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \hat{\mathcal{D}}^{\mathbf{s}_i})\big\|_2^2 + \beta TV(\hat{\mathcal{X}}^{\mathbf{s}_i}) \cdot \mathbb{1}_\xi(\hat{\mathcal{X}}^{\mathbf{s}_i}) + \gamma \sum_{n=1}^{N} \big\|\mathbf{s}_i[n](\hat{\mathbf{h}}_n^* - \widetilde{\mathbf{h}}_n)\big\|_2^2\Big] \tag{11}$$

where α, β and γ are coefficients, the indicator function $\mathbb{1}_\xi(\hat{\mathcal{X}}^{\mathbf{s}_i})$ is 1 if the total variation norm $TV(\hat{\mathcal{X}}^{\mathbf{s}_i})$ is larger than ξ and 0 otherwise, and $\hat{\mathcal{D}}^{\mathbf{s}_i} = \{\{\hat{\mathbf{x}}_n, \hat{y}_n\} | \mathbf{s}_i[n] = 1\}$.

In each iteration t, the objective function of step III can be denoted by

$$\mathcal{F}_3^t(\hat{\mathcal{D}}) = \alpha\big\|\nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) - \nabla_{\mathbf{\Theta}}\mathcal{L}(\mathbf{\Theta}, \hat{\mathcal{D}}(t))\big\|_2^2 + \beta TV(\hat{\mathcal{X}}^{\mathbf{s}^t}) \cdot \mathbb{1}_\xi(\hat{\mathcal{X}}^{\mathbf{s}^t}) + \gamma \sum_{n=1}^{N} \big\|\mathbf{s}^t[n](\hat{\mathbf{h}}_n^* - \widetilde{\mathbf{h}}_n)\big\|_2^2 \tag{12}$$

The first term in (12) is the ℓ_2 norm in [29]. The second term is the TV norm and the last term is the internal representation norm regularizer. The gradients of $\mathcal{F}_3^t(\hat{\mathcal{D}})$ in (12) w.r.t $\hat{\mathcal{D}}$ are denoted by

$$\nabla_{\hat{\mathcal{D}}}\mathcal{F}_3^t(\hat{\mathcal{D}}) = \{\nabla_{\hat{\mathbf{x}}_n}\mathcal{F}_3^t(\hat{\mathcal{D}}), \nabla_{\hat{y}_n}\mathcal{F}_3^t(\hat{\mathcal{D}})\}_{n=1}^N \tag{13}$$

We will provide an ablation study in Section 4.2 to demonstrate the utility of these regularizers. To ensure attacking efficiency, we consider two reasonable settings on the flexible update protocols: Algorithm 3: CAFE (Nested) and Algorithm 4: CAFE (Single-loop). Empirically, Algorithm 4 will take less iterations than those of Algorithm 3, and thus extra iterations will be saved. More details can be seen in the experiment results in Section 4.2.

3.4 Defense strategy: Leveraging fake gradients as an countermeasure to CAFE

Although CAFE comes with theoretical recovery guarantees, the underlying premise is that the clients will upload true (correct) gradients for aggregation. Therefore, we propose an intuitive and practical approach to mitigate CAFE by requiring each client to upload fake gradients, resulting in incorrect data recovery via CAFE. More specifically, to solve the problem of leakage from true gradients, we design a defense called **Fake Gradients** and summarize it in Algorithm 5 of Appendix F. The main idea of this defense is that attackers will aim to match wrong gradients and invert incorrect input of the first fully connected layer so that attackers cannot recover the true training data. The defending strategy in Algorithm 5 can be added between line 8 and 9 in Algorithms 1 and 2.

In particular, each local user can randomly generate gradients with the normal distribution $\mathcal{N}(0, \sigma^2)$ and sort the elements in descending order (Line 1, 2). At the same time, local users also sort their true gradients in descending order and record indexes of the sorted items (Line 7). Then, one computes the L_2-norm distance between a true gradient and all fake gradients to find the nearest fake gradient (Line 8). Afterwards, we pair fake gradients to match true gradients by the sorted order (Line 13). This an important step so that we can keep large/small values at the same positions of true gradients. Namely, it is the critical point of why we can keep the training performance by using fake gradients. Finally, local users upload the fake gradients to the server.

Impact on model training. Chen et al. [4] propose that if the distance between the actual gradient and the gradient surrogate is smaller than a decreasing threshold, using the gradient surrogate to update the model still guarantees convergence. Building upon the results in [4], we set a sufficient threshold such that the distance between the fake gradients and the true gradients are smaller than the threshold. In this case, we can still achieve the learning performance as if true gradients are used.

4 Experiments

We conduct experiments on MNIST [16], CIFAR-10 [15] and Linnaeus 5 [3] datasets in VFL settings. The objective function (8) in CAFE step I is optimized by stochastic gradient descent (GD) method. The objective function (10) in CAFE step II is optimized by Adam method. The objective function (12) in CAFE step III is optimized by normalized GD. Our algorithm recovers all the data participating in VFL with a relative large batch size (more than 40). Scaling up to our hardware limits (RTX 2080 and TITAN V), CAFE can leak as many as 800 images in the VFL setting including 4 workers with a betch size as large as 100. The model structure is shown in Figure 3.

Evaluation metrics. To measure the data leakage performance, we introduce peak signal-to-noise ratio (PSNR) value with mean squared error (MSE) defined in (14). Higher PSNR value of leaked data represents better performance of data recovery.

$$\mathrm{MSE_c}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{\mathrm{HW}} \sum_{\mathrm{i}=1}^{\mathrm{H}} \sum_{\mathrm{j}=1}^{\mathrm{W}} [\mathbf{x}_{\mathrm{ijc}} - \hat{\mathbf{x}}_{\mathrm{ijc}}]^2; \ \mathrm{PSNR}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{\mathrm{C}} \sum_{\mathrm{c}=1}^{\mathrm{C}} \Big[\log_{10} \Big(\frac{(\max_{\mathrm{i,j}} \mathbf{x}_{\mathrm{ijc}})^{20})}{(\mathrm{MSE_c}(\mathbf{x}, \hat{\mathbf{x}}))^{10}} \Big) \Big]. \tag{14}$$

4.1 Comparison with the state-of-the-art

We compare with prior arts using the batch size of 40 images in each iteration. For fair comparisons, all methods were run on the the same model and iterations. Three state-of-the-art methods for image recovery are compared with CAFE:

i) DLG [29]: The deep gradient leakage method is equivalent to replacing the objective function in (12) with the squared ℓ_2 norm distance.

ii) Cosine Similarity [10]: The objective function is equivalent to replacing the objective function in (12) with the linear combination of cosine similarity and TV norm of the recovered images.

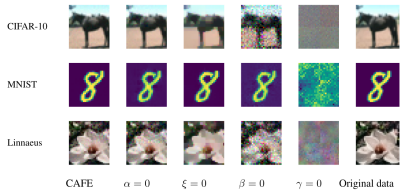


Figure 4: Visual comparison on the effect of auxiliary regularizers.

Figure 5: Visual comparison of the real and recovered data using ordinary and fake gradients.

iii) SAPAG [22]: The objective function is equivalent to replacing the objective function in (12) with the Gaussian kernel based function.

We did not compare to other methods in Table 2 because methods such as R-GAP [28] and Theory oriented [20] require certain conditions which may fail to satisfy. Other methods such as GradInversion [26] have certain requirements on batch size and number of classes that are not applicable.

CAFE outperforms these methods both qualitatively (Figure 1) and quantitatively (Table 2). Its PSNR values are always above 30 at the end of each CAFE attacking process, suggesting high data recovery quality. However, the PSNR of other methods are below 10 on all the three datasets.

Table 2: Comparison with the state-of-the-art (4 workers, batch size = 40, batch ratio = 0.05)

PSNR / Dataset / Method	CIFAR10	MNIST	Linnaeus 5
CAFE	31.83	43.15	33.22
DLG[29]	9.29	7.96	7.14
Cosine Similarity [10]	7.38	7.84	8.31
SAPAG[22]	6.07	3.86	6.74

Table 3: PSNR via Batch size (800 data samples in total)

PSNR / Dataset / Batch size	CIFAR-10	MNIST	Linnaeus 5
10	30.83	32.60	28.00
20	35.70	39.00	30.53
40	31.83	43.15	33.22
80	36.87	47.05	30.43
100	38.94	47.50	29.18

4.2 Ablation study

We test CAFE under different batch size, network structure and with/without auxiliary regularizers.

(i) PSNR via Batch size. Table 3 shows that the PSNR values always keep above 30 on CIFAR-10, above 32 on MNIST and above 28 on Linnaeus 5 when the batch size increases with fixed number of workers and number of total data points. The result implies that the increasing batch size has almost no influence on data leakage performance of CAFE and it fails to be an effective defense.

(ii) Effect of regularizers. Table 4 demonstrates the impact of auxiliary regularizers. From Figure 4, adjusting the threshold ξ prevents images from being over blurred during the reconstruction process. TV norm can eliminate the noisy patterns on the recovered images and increase the PSNR. We also find that the last term in (12), the internal representation norm regularizer, contributes most to the data recovery. In Table 4, CAFE still performs well without the first term ($\alpha = 0$) in (12). The reason is that the domination of the internal representation norm regularizer allow data to be fully recovered. Notably, CAFE also performs well on MNIST even without the second term ($\beta = 0$) in (12). It is mainly for the reason that MNIST is a simple dataset that CAFE can successfully recover it without the help of TV norm regularizer.

(iii) Nested loop vs Single loop. We compare both modes of CAFE (Algorithms 3 and 4) on all datasets. In Table 5, the number of iterations in the brackets represent the maximum iterations in each step. For the CAFE single-loop mode, if the objective function in step I (8) decreases below 10^{-9}, we break the loop and move on to step II. If the objective function in step II (10) decreases below 5×10^{-9}, we break the loop and move on to step III. For evaluation, when the PSNR value reaches 27 on CIFAR-10, 30 on Linnaeus, 38 on MNIST, we stop both algorithms and record the iteration numbers. As shown in Table 5, CAFE nested-loop requires fewer number of iterations. Meanwhile, it is difficult to set the loop stopping conditions in the CAFE single-loop mode. In particular, $\mathbf{V}^*$ and $\hat{\mathbf{H}}^*$ with low recovery precision may impact the data recovery performance.

Table 4: Effect of auxiliary regularizers
(4 workers, batch size = 40, batch ratio = 0.05)

PSNR / Datasets / Algorithm	CIFAR-10	Linnaeus	MNIST
CAFE	31.83	33.22	43.15
CAFE ($\alpha = 0$)	33.93	28.62	31.93
CAFE ($\xi = 0$)	25.57	25.29	34.51
CAFE ($\beta = 0$)	18.25	23.22	31.98
CAFE ($\gamma = 0$)	12.51	12.37	6.34

Table 5: Nested vs Single loop for CAFE
(4 workers, batch size = 40, batch ratio = 0.05)

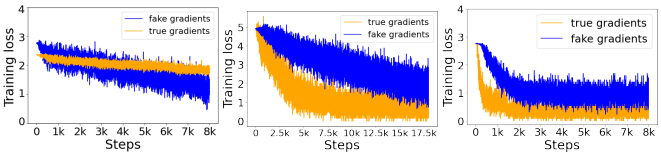
Iterations / mode / Datasets	CIFAR-10	MNIST	Linnaues 5
Nested	7300 (8000)	6600 (8000)	12400 (20000)
Single loop Step I	8000 (8000)	8000 (8000)	12428 (20000)
Single loop Step II	2404 (8000)	8000 (8000)	20000 (20000)
Single loop Step III	1635 (8000)	2468 (8000)	20000 (20000)

4.3 Attacking while training

Previous works have shown that DLG performs better on an untrained model than a trained one [10]. We also implement CAFE in the 'attacking while training' scenario, in which the VFL process is continuously updating. When the network is training, the selected batch data and the parameters of the model change every iteration, which may cause the attack loss to diverge. However, according to our experimental results in Table 6, CAFE is able to recover training images when the learning rate (lr) is relatively small. Increasing the learning rate renders data leakage more difficult because the model is making more sizeable parameter changes in each iteration, which can be regarded as an effective defense strategy. However, when the lr is relatively small, the attacker still has some chances (e.g. when the training adopts learning rate warm up [11]) to recover the data.

Table 6: Attacking while training in VFL.

PSNR(lr) / Setting / Dataset	1	2	3
CIFAR10	31.24 (10^{-4})	27.62 (5×10^{-4})	25.22 (10^{-3})
MNIST	31.82 (10^{-4})	28.42 (5×10^{-4})	23.60 (10^{-3})
Linnaeus	30.74 (10^{-6})	21.45 (5×10^{-5})	20.68 (10^{-4})



(a) CIFAR-10 (b) Linnaeus (c) MNIST

Figure 6: Training loss of true gradients and fake gradients on CIFAR-10, Linnaeus and MNIST.

4.4 Mitigation of CAFE via fake gradients

To demonstrate how fake gradients defend against CAFE (Section 3.4), we conduct CAFE with unchanged $\mathbf{\Theta}$, which is the strongest data leakage attack setting. We use the SGD optimizer with learning rate as 0.1, $\sigma^2 = 1.1$, and $\nu = 1000$ for fake gradients. Figure 5 shows a comparison between the visual image quality of the data recovered by CAFE on CIFAR-10 when the ordinary gradients and fake gradients are used, respectively. The PSNR of recovered data in CAFE on ordinary and fake gradients is 28.68 and 7.67, respectively. Moreover, Figure 6 shows that the training process with fakes gradients behaves in a similar way to the one with true gradients, confirming that the use of fake gradients does not lose the training efficacy.

5 Conclusion

In this paper, we uncover the risk of *catastrophic data leakage in federated learning (CAFE)* through a novel algorithm that can perform large-batch data leakage with high data recovery quality and theoretical guarantees. Extensive experimental results demonstrate that CAFE can recover large-scale private data from the shared aggregated gradients on vertical FL settings, overcoming the batch limitation problem in current data leakage attacks. We also propose an effective countermeasure using fake gradients to mitigate the potential risks of CAFE.

References

[1] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv, preprint:1807.00459*, 2018.

[2] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*, Long Beach, California, 2019, pp. 634–643.

[3] G. K. L. Chaladze, "Linnaeus 5 dataset for machine learning," 2017.

[4] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[5] T. Chen, X. Jin, Y. Sun, and W. Yin, "VAFL: a method of vertical asynchronous federated learning," *arXiv preprint:2007.06081*, 2020.

[6] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Secureboost: A lossless federated learning framework," *arXiv, eprint:1901.08755*, 2019.

[7] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. Broomfield, CO: USENIX Association, Oct. 2014, pp. 571–582.

[8] L. Fan, K. Ng, C. Ju, T. Zhang, C. Liu, C. S. Chan, and Q. Yang, "Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks," 2020.

[9] 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*. New York, NY, USA: Association for Computing Machinery, 10 2015, pp. 1322–1333.

[10] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients – how easy is it to break privacy in federated learning?" *arXiv, eprint:2003.14053*, 2020.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[12] B. Hitaj, G. Ateniese, and F. Pérez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," *arXiv, eprint:1702.07464*, 2017.

[13] F. N. Iandola, K. Ashraf, M. W. Moskewicz, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," *arXiv, eprint:1511.00175*, 2015.

[14] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint:1912.04977*, Dec. 2019.

[15] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[16] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[17] M. Li, "Scaling distributed machine learning with the parameter server," in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*. USA: USENIX Association, 08 2014, pp. 1–1.

[18] H. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, Ft. Lauderdale, FL, USA, 2017.

[19] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov, "Inference attacks against collaborative learning," in *Proceedings of the 35th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2018, p. 148–162.

[20] X. Pan, M. Zhang, Y. Yan, J. Zhu, and M. Yang, "Theory-oriented deep leakage from gradients via linear equation solver," 2020.

[21] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1310–1321.

[22] Y. Wang, J. Deng, D. Guo, C. Wang, X. Meng, H. Liu, C. Ding, and S. Rajasekaran, "Sapag: A self-adaptive privacy attack from gradients," *arXiv, eprint:2009.06228*, 2020.

[23] W. Wei, L. Liu, M. Loper, K. H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," 2020.

[24] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019.

[26] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Conference on Computer Vision and Pattern Recognition*, 2021.

[27] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv, eprint:2001.02610*, 2020.

[28] J. Zhu and M. B. Blaschko, "R-GAP: Recursive gradient attack on privacy," in *International Conference on Learning Representations*, 2021.

[29] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems 32*, Vancouver, Canada, 2019, pp. 14 774–14 784.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4.3.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We uncover a new risk of catastrophic data leakage in federated learning and propose a countermeasure to mitigate the negative impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]