A Framework for Evaluating Client Privacy Leakages in Federated Learning

Wenqi Wei(✉), Ling Liu, Margaret Loper, Ka-Ho Chow,
Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu

Georgia Institute of Technology, Atlanta, GA 30332, USA
{wenqiwei,khchow,memregursoy,staceytruex,yanzhaowu}@gatech.edu,
ling.liu@cc.gatech.edu, margaret.loper@gtri.gatech.edu

Abstract. Federated learning (FL) is an emerging distributed machine learning framework for collaborative model training with a network of clients (edge devices). FL offers default client privacy by allowing clients to keep their sensitive data on local devices and to only share local training parameter updates with the federated server. However, recent studies have shown that even sharing local parameter updates from a client to the federated server may be susceptible to gradient leakage attacks and intrude the client privacy regarding its training data. In this paper, we present a principled framework for evaluating and comparing different forms of client privacy leakage attacks. We first provide formal and experimental analysis to show how adversaries can reconstruct the private local training data by simply analyzing the shared parameter update from local training (e.g., local gradient or weight update vector). We then analyze how different hyperparameter configurations in federated learning and different settings of the attack algorithm may impact on both attack effectiveness and attack cost. Our framework also measures, evaluates, and analyzes the effectiveness of client privacy leakage attacks under different gradient compression ratios when using communication efficient FL protocols. Our experiments additionally include some preliminary mitigation strategies to highlight the importance of providing a systematic attack evaluation framework towards an in-depth understanding of the various forms of client privacy leakage threats in federated learning and developing theoretical foundations for attack mitigation.

Keywords: Privacy leakage attacks · Federated learning · Attack evaluation framework

1 Introduction

Federated learning enables the training of a high-quality ML model in a decentralized manner over a network of devices with unreliable and intermittent network connections [5,14,20,26,29,33]. In contrast to the scenario of prediction on edge devices, in which an ML model is first trained in a highly controlled Cloud environment and then downloaded to mobile devices for performing predictions, federated learning brings model training to the devices while supporting continuous learning on device. A unique feature of federated learning is to decouple the ability of conducting machine learning from the need of storing all training data in a centralized location [13].

Although federated learning by design provides the default privacy of allowing thousands of clients (e.g., mobile devices) to keep their original data on their own devices, while jointly learn a model by sharing only local training parameters with the server. Several recent research efforts have shown that the default privacy in FL is insufficient for protecting the underlaying training data from privacy leakage attacks by gradient-based reconstruction [9,32,34]. By intercepting the local gradient update shared by a client to the FL server before performing federated averaging [13,19,22], the adversary can reconstruct the local training data with high reconstruction accuracy, and hence intrudes the client privacy and deceives the FL system by sneaking into client confidential training data illegally and silently, making the FL system vulnerable to client privacy leakage attacks (see Sect. 2.2 on Threat Model for detail).

In this paper, we present a principled framework for evaluating and comparing different forms of client privacy leakage attacks. Through attack characterization, we present an in-depth understanding of different attack mechanisms and attack surfaces that an adversary may leverage to reconstruct the private local training data by simply analyzing the shared parameter updates (e.g., local gradient or weight update vector). Through introducing multi-dimensional evaluation metrics and developing evaluation testbed, we provide a measurement study and quantitative and qualitative analysis on how different configurations of federated learning and different settings of the attack algorithm may impact the success rate and the cost of the client privacy leakage attacks. Inspired by the attack effect analysis, we present some mitigation strategies with preliminary results to highlight the importance of providing a systematic evaluation framework for comprehensive analysis of the client privacy leakage threats and effective mitigation methods in federated learning.

2 Problem Formulation

2.1 Federated Learning

In federated learning, the machine learning task is decoupled from the centralized server to a set of N client nodes. Given the unstable client availability [21], for each round of federated learning, only a small subset of $K_t <$ clients out of all N participants will be chosen to participate in the joint learning.

Local Training at a Client: Upon notification of being selected at round t, a client will download the global state $w(t)$ from the server, perform a local training computation on its local dataset and the global state, i.e., $w_k(t+1) = w_k(t) - \eta \nabla w_k(t)$, where $w_k(t)$ is the local model parameter update at round t and ∇w is the gradient of the trainable network parameters. Clients can decide its training batch size B_t and the number of local iterations before sharing.

Update Aggregation at FL Server: Upon receiving the local updates from all K_t clients, the server incorporates these updates and update its global state, and initiates the next round of federated learning. Given that local updates can be in the form of either gradient or model weight update, thus two update aggregation implementations are the most representative. In *distributed SGD* [17,18,29,30], each client uploads the local gradients to the FL server at each round and the server iteratively aggregates the local gradients from all K_t clients into the global model: $w(t+1) = w(t) - \eta \sum_{k=1}^{K_t} \frac{n_t}{n} \nabla w_k(t)$, where η is the global learning rate and $\frac{n_t}{n}$ is the weight of client k. n_k is the number of data points at client k and n indicates the amount of total data from all participating clients at round t. In *federated averaging* [3,20], each client uploads the local training parameter update to the FL server and the server iteratively performs a weighted average of the received weight parameters to update the global model: $w(t+1) = \sum_{k=1}^{K_t} \frac{n_t}{n} w_k(t+1)$, where $\Delta w_k(t)$ denote the difference between the model parameter update before the iteration of training and the model parameter update after the training for client k.

2.2 Threat Model

In an FL system, clients are the most vulnerable attack surface for the client privacy leakage (CPL) attack. To focus on client gradient leakage vulnerability due to sharing its local gradient update, we assume that clients may be partially compromised: (i) an adversary may intercept the local parameter update prior to sending it via the client-to-server communication channel to the FL server and (ii) an adversary may gain access to the saved local model executable (checkpoint data) on the compromised client with no training data visible and launch white-box gradient leakage attacks to steal the private training data. Note that local data and model may not be compromised at the same time for many reasons, e.g., separated storage of data and model, or dynamic training data.

We also assume that the federated server is honest and will truthfully perform the aggregation of local parameter updates and manage the iteration rounds for jointly learning. However, the FL server may be curious and may analyze the periodic updates from certain clients to perform client privacy leakage attacks and gain access to the private training data of the victim clients. Given that the gradient-based aggregation and model weight-based aggregation are mathematically equivalent, one can obtain the local model parameter difference from the local gradient and the local training learning rate. In this paper, gradient-based aggregation is used without loss of generality.

2.3 The Client Privacy Leakage (CPL) Attack: An Overview

The client privacy leakage attack is a gradient-based feature reconstruction attack, in which the attacker can design a gradient-based reconstruction learning algorithm that will take the gradient update at round t, say $\nabla w_k(t)$, to be shared by the client, to reconstruct the private data used in the local training computation. For federated learning on images or video clips, the reconstruction algorithm will start by using a dummy image of the same resolution as its attack initialization seed, and run a test of this attack seed on the intermediate local model, to compute a gradient loss using a vector distance loss function between the gradient of this attack seed and the actual gradient from the client local training. The goal of this reconstruction attack is to iteratively add crafted small noises to the attack seed such that the generated gradient from this reconstructed attack seed will approximate the actual gradient computed on the local training data. The reconstruction attack terminates when the gradient of the attack seed reconstructed from the dummy initial data converges to the gradient of the training data. When the gradient-based reconstruction loss function is minimized, the reconstructed attack data will also converge to the training data with high reconstruction confidence.

Algorithm 1. Gradient-based Reconstruction Attack

```
1: Inputs:
   $f(x; w(t))$: Differentiable learning model, $\nabla w_k(t)$: gradients produced by the local training on private training data $(x; y)$ at client $k$, $w(t)$, $w(t+1)$: model parameters before and after the current local training on $(x; y)$, $\eta_k$ learning rate of local training
   Attack configurations: $INIT(x.type)$: attack initialization method, $\mathbb{T}$: attack termination condition, $\eta'$: attack optimization method, $\alpha$: regularizer ratio.
2: Output: reconstructed training data $(x_{rec}; y_{rec})$
3: Attack procedure
4: if $w_k(t+1)$: then
5:     $\Delta w_k(t) \leftarrow w_k(t+1) - w(t)$
6:     $\nabla w_k(t) \leftarrow \frac{\Delta w_k(t)}{\eta_k}$
7: end if
8: $x^0_{rec} \leftarrow INIT(x.type)$
9: $y_{rec} \leftarrow \arg\min_i(\nabla_i w_k(t))$
10: for $\tau$ in $\mathbb{T}$ do
11:     $\nabla w^{\tau}_{att}(t) \leftarrow \frac{\partial loss(f(x^{\tau}_{rec}, w(t)), y_{rec})}{\partial w(t)}$
12:     $D^{\tau} \leftarrow ||\nabla w^{\tau}_{att}(t) - \nabla w_k(t)||^2 + \alpha||f(x^{\tau}_{rec}, w(t)) - y_{rec}||^2$
13:     $x^{\tau+1}_{rec} \leftarrow x^{\tau}_{rec} - \eta' \frac{\partial D^{\tau}}{\partial x^{\tau}_{rec}}$
14: end for
```

Algorithm 1 gives a sketch of the client privacy leakage attack method. Line 4–6 convert the weight update to gradient when the weight update is shared between the FL server and the client. The learning rate η_k for local training is assumed to be identical across all clients in our prototype system. Line 8 invokes the dummy attack seed initialization, which will be elaborated in Sect. 3.1. Line 9 is to get the label from the actual gradient shared from the local training. Since the local training update towards the ground-truth label of the training input data should be the most aggressive compared to other labels, the sign of gradient for the ground-truth label of the private training data will be different than other classes, and its absolute value is usually the largest. Line 10–14 presents the iterative reconstruction process that produces the reconstructed private training data based on the client gradient update. If the reconstruction algorithm converges, then the client privacy leakage attack is successful, or else the CPL attack is failed. Line 12–13 show that when the L_2 distance between the gradients of the attack reconstructed data and the actual gradient from the private training data is minimized, the reconstructed attack data from the dummy seed converges to the private local training data, leading to the client privacy leakage. In Line 12, a label-based regularizer is utilized to improve the stability of the attack optimization. An alternative way to reconstruct the label of the private training data is to initialize a dummy label and feed it into the iterative approximation algorithm for attack optimization [34], in a similar way as the content reconstruction optimization.

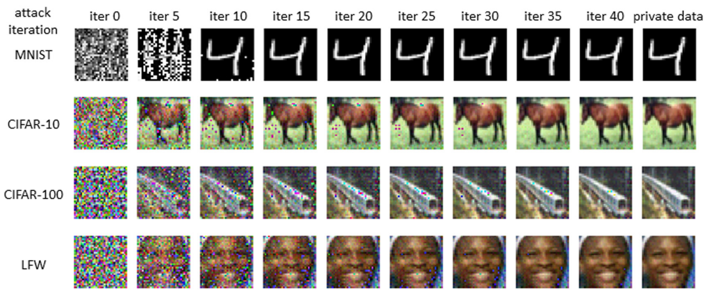

Fig. 1. Example illustration of the Client Privacy Leakage attack

Figure 1 provides a visualization of four illustrative example attacks over four datasets: MNIST [16], CIFAR10 [15], CIFAR100 [15], and LFW [12]. We make two interesting observations from Algorithm 1. *First*, multiple factors in the attack method could impact the attack success rate (ASR) of the client privacy leakage attack, such as the dummy attack seed data initialization method (Line 8), the attack iteration termination condition ($\mathbb{T}$), the selection of the gradient loss (distance) function (Line 12), the attack optimization method (Line 13). *Second*, the configuration of some hyperparameters in federated learning may also impact the effectiveness and cost of the CPL attack, including batch size, training data resolution, choice of activation function, and whether the gradient update is uploaded to the FL server using baseline communication protocol or a communication efficient method. In the next section, we present the design of our evaluation framework to further characterize the client privacy leakage attack of different forms and introduce cost-effect metrics to measure and analyze the adverse effect and cost of CPL attacks. By utilizing this framework, we provide a comprehensive study on how different attack parameter configurations and federated learning hyperparameter configurations may impact the effectiveness of the client privacy leakage attacks.

3 Evaluation Framework

3.1 Attack Parameter Configuration

Attack Initialization: We first study how different methods for generating the dummy attack seed data may influence the effectiveness of a CPL attack in terms of reconstruction quality or confidence as well as reconstruction time and convergence rate. A straightforward dummy data initialization method is to use a random distribution in the shape of dummy data type and we call this baseline the random initialization (CPL-random). Although random initialization is also used in [9,32,34], our work is, to the best of our knowledge, the first study on variations of attack initiation methods. To understand the role of random seed in the CPL attack, it is important to understand the difference of the attack reconstruction learning from the normal deep neural network (DNN) training. In a DNN training, it takes as the training input both the fixed data-label pairs and the initialization of the learnable model parameters, and iteratively learn the model parameters until the training converges, which minimizes the loss with respect to the ground truth labels. In contrast, the CPL attack performs reconstruction attack by taking a dummy attack seed input data, a fixed set of model parameters, such as the actual gradient updates of a client local training, and the gradient derived label as the reconstructed label y_{rec}, its attack algorithm will iteratively reconstruct the local training data used to generate the gradient, $\nabla w_k(t)$, by updating the dummy synthesized seed data, following the attack iteration termination condition $\mathbb{T}$, denoted by $\{x_{rec}^0, x_{rec}^1, ...x_s{}^{\mathbb{T}}\} \in \mathbb{R}^d$, such that the loss between the gradient of the reconstructed data x_{rec}^i and the actual gradient $\nabla w_k(t)$ is minimized. Here x_{rec}^0 denotes the initial dummy seed.

Theorem 1 ***(CPL Attack Convergence Theorem).*** *Let x_{rec}^* be the optimal synthesized data for $f(x)$ and attack iteration $t \in \{0, 1, 2, ...T\}$. Given the convexity and Lipschitz-smoothness assumption, the convergence of the gradient-based reconstruction attack is guaranteed with:*

$$f(x_{rec}^{\mathbb{T}}) - f(x_{rec}^*) \leq \frac{2L||x_{rec}^0 - x_{rec}^*||^2}{\mathbb{T}}. \tag{1}$$

The above CPL Attack Convergence theorem is derived from the well-established Convergence Theorem of Gradient Descent. Due to the limitation of space, the formal proof of Theorem 1 is provided in the Appendix. Note that the convexity assumption is generally true since the d-dimension trainable synthesized data can be seen as a one-hidden-layer network with no activation function. The fixed model parameters are considered as the input with optimization of the least square estimation problem as stated in Line 12 of Algorithm 1.

According to Theorem 1, the convergence of the CPL attack is closely related to the initialization of the dummy data x_{rec}^0. This motivates us to investigate different ways to generate dummy attack seed data. Concretely, we argue that different random seeds may have different impacts on both reconstruction learning efficiency (confidence) and reconstruction learning convergence (time or the number of iteration steps). Furthermore, using geometrical initialization as those introduced in [24] not only can speed up the convergence but also ensure attack stability. Consider a single layer of the neural network: $g(x) = \sigma(wx + b)$, a geometrical initialization considers the form $g(x) = \sigma(w_*(x - b_*)$ instead of directly initialing w and b with random distribution. For example, according to [31], the following partial derivative of the geometrical initialization.

$$\frac{\partial g}{\partial w_*} = \sigma'(w_*(x - b_*))(x - b_*), \tag{2}$$

is more independent of translation of the input space than $\frac{\partial g}{\partial w} = \sigma'(wx + b)x$, and is therefore more stable.

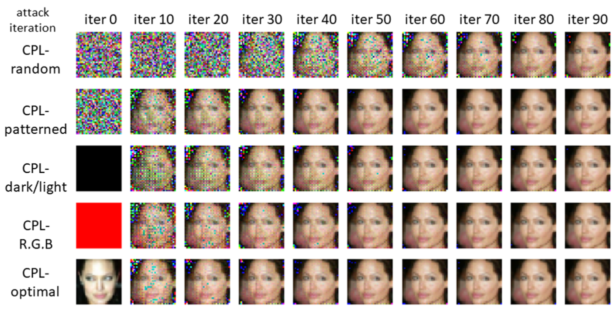

Fig. 2. Visualization of different initialization

Figure 2 provides a visualization of five different initialization methods and their impact on the CPL attack in terms of reconstruction quality and convergence (#iterations). In addition to CPL-random, CPL-patterned is a method that uses patterned random initialization. We initialize a small portion of the dummy data with a random seed and duplicate it to the entire feature space. An example of the portion can be 1/4 of the feature space. CPL-dark/light is to use a dark (or light) seed of the input type (size), whereas CPL-R.G.B. is to use red or green or blue solid color seed of the input type (size). CPL-optimal refers to the theoretical optimal initialization method, which uses an example from the same class as the private training data that the CPL attack aims to reconstruct. We observe from Fig. 2 that CPL-patterned, CPL-R.G.B., and CPL-dark/light can outperform CPL-random with faster attack convergence and more effective reconstruction confidence. We also include CPL-optimal to show that different CPL initializations can effectively approximate the optimal way of initialization in terms of reconstruction effectiveness.

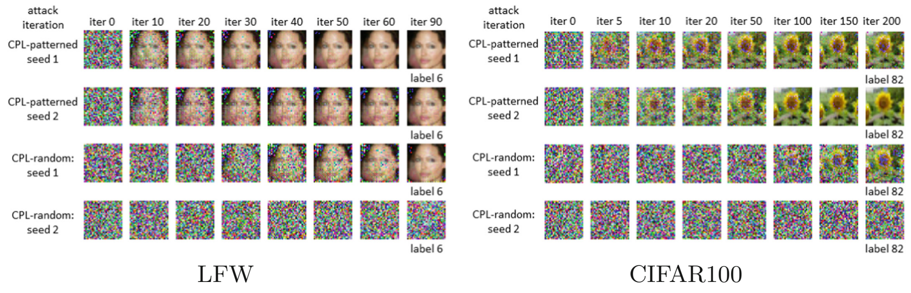

Fig. 3. Effect of different random seed

Figure 3 shows that the CPL attacks are highly sensitive to the choice of random seeds. We conduct this set of experiments on LFW and CIFAR100, and both confirm consistently our observations: different random seeds lead to diverse convergence processes with different reconstruction quality and confidence. From Fig. 3, we observe that even with the same random seed, attack with patterned initialization is much more efficient and stable than the CPL-random. Moreover, there are situations where the private label of the client training data is successfully reconstructed but the private content reconstruction fails (see the last row for both LFW and CIFAR100).

Attack Termination Condition: The effectiveness of a CPL attack is also sensitive to the setting of the attack termination condition. Recall Sect. 2.3 and Algorithm 1, there are two decision factors for termination. One is the maximum attack iteration [34] and the other is the L_2-distance threshold of the gradient loss [32], i.e., the difference between the gradient from the reconstructed data and the actual gradient from the local training using the private local data. Other gradient loss functions, e.g., cosine similarity, entropy, can be used to replace the L_2 function. Table 1 compares six different settings of attack iterations for configuring termination condition. A small attack iteration cap may cause the reconstruction attack to abort before it can succeed the attack, whereas a larger attack iteration cap may increase the cost of attack. The result in Table 1 confirms that choosing the cap for attack iterations may have an impact on both attack effectiveness and cost.

Table 1. Effect of termination condition

maximum attack iteration		10	20	30	50	100	300
LFW	CPL-patterned	0	0.34	0.98	1	1	1
	CPL-random	0	0	0	0.562	0.823	0.857
CIFAR10	CPL-patterned	0	0.47	0.93	0.973	0.973	0.973
	CPL-random	0	0	0	0	0.356	0.754
CIFAR100	CPL-patterned	0	0	0.12	0.85	0.981	0.981
	CPL-random	0	0	0	0	0.23	0.85

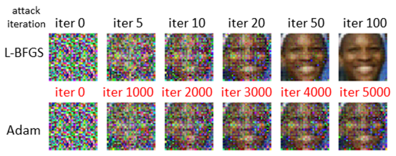

Fig. 4. Effect of attack optimization

Attack Optimization: Optimization methods, such as Stochastic Gradient descent, Adam, and Adagrad can be used to iteratively update the dummy data during the reconstruction of a CPL attack. While the first-order optimization techniques are easy to compute and less time consuming, the second-order techniques are better in escaping the slow convergence paths around the saddle points [4]. Figure 4 shows a comparison of L-BFGS [6] and Adam and their effects on the CPL-patterned attack for LFW dataset. It takes fewer attack iterations to get high reconstruction quality using the L-BFGS compared to using Adam as the optimizer in the reconstruction attack, confirming that choosing an appropriate optimizer may impact on attack effectiveness.

3.2 Hyperparameter Configurations in Federated Learning

Batch Size: Given that all forms of CPL attack methods are reconstruction learning algorithms that iteratively learn to reconstruct the private training data by inferencing over the actual gradient to perform iterative updates on the dummy attack seed data, it is obvious that a CPL attack is most effective when working with the gradient generated from the local training data of batch size 1. Furthermore, when the input data examples in a batch of size B belongs to only one class, which is often the case for mobile devices and the non-i.i.d distribution of the training data [33], the CPL attacks can effectively reconstruct the training data of the entire batch. This is especially true when the dataset has low inter-class variation, e.g., face and digit recognition. Figure 5 shows the visualization of performing a CPL-patterned attack on the LFW dataset with four different batch sizes. With a large batch size, the detail of a single input is being neutralized, making the attack reconstruction captures more of the shared features of the entire batch rather than specific to a single image.

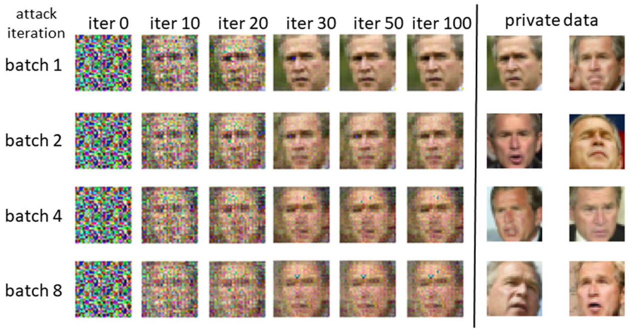

Fig. 5. Effect of batch size in CPL-patterned attacks on LFW

Training Data Resolution: In contrast to the early work [34] that fails to attack images of resolution higher than 64×64, we argue that the effectiveness of the CPL attack is mainly attributed to the model overfitting to the training data. In order to handle higher resolution training data, we double the number of filters in all convolutional layers to build a more overfitted model. Figure 6 shows the scaling results of CPL attack on the LFW dataset with input data size of 32×32, 64×64, and 128×128. CPL-random requires a much larger number of attack iterations to succeed the attack with high reconstruction performance. CPL-patterned is a significantly more effective attack for all three different resolutions with 3 to $4\times$ reduction in the attack iterations compared to CPL-random. We also provide an example of attacking the 512×512 Indiana University Chest X-Rays image of very high resolution in Fig. 7.

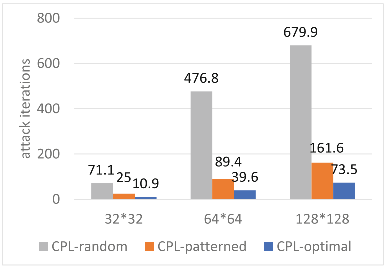

Fig. 6. Attack iteration of the training data scaling using LFW dataset

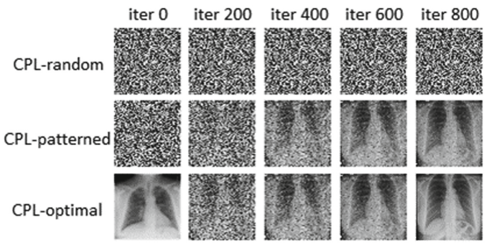

Fig. 7. Extreme scaling case of attacking 512×512 Chest X-ray image

Activation Function: The next hyperparameter of FL is the activation function used in model training. We show that the performance of the CPL attacks is highly related to the choice of the activation function. Figure 8 compares the attack iterations and attack success rate of CPL-patterned attack with three different activation functions: Sigmoid, Tanh, and LeakReLU. Due to space constraint, the results on MNIST is omitted. We observe that ReLU naturally prevents the full reconstruction of the training data using gradient because the gradient of the negative part of ReLU will be 0, namely, that part of the trainable parameters will stop responding to variations in error and will not get adjusted during optimization. This dying ReLU problem takes out the gradient information needed for CPL attacks. In comparison, both Sigmoid and Tanh are differentiable bijective and can pass the gradient from layer to layer in an almost lossless manner. LeakyReLU sets a slightly inclined line for the negative part of ReLU to mitigate the issue of dying ReLU and thus is vulnerable to CPL attacks.

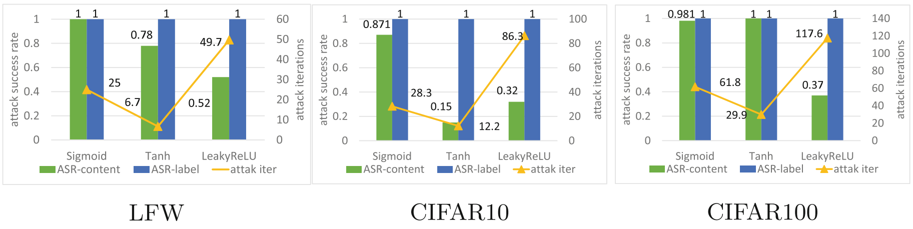

Fig. 8. Effect of activation function on the CPL attack.

Motivated by the impact of activation function, we argue that any model components that discontinue the integrity and uniqueness of gradients can hamper CPL attacks. We observe from our experiments that an added dropout structure enables different gradient in every query, making $\nabla w_{att}^{\tau}(t)$ elusive and unable to converge to the uploaded gradients. By contrast, pooling cannot prevent CPL attacks since pooling layers do not have parameters.

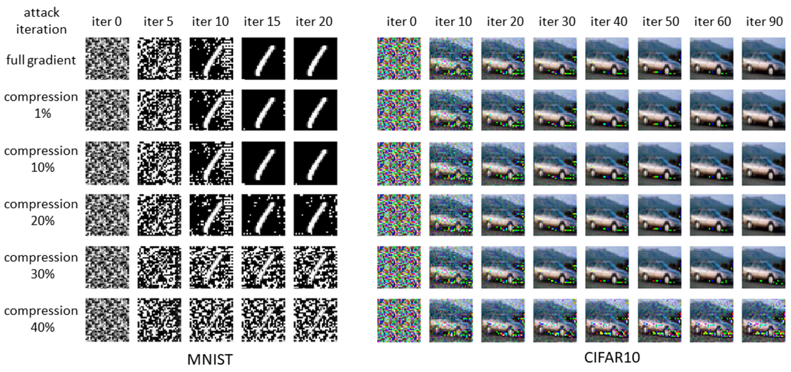

Fig. 9. Illustration of the CPL attack under communication-efficient update

Baseline Protocol v.s. Communication-Efficient Protocol: In the baseline communication protocol, the client sends a full vector of local training parameter update to the FL server in each round. For federated training of large models on complex data, this step is known to be the bottleneck of Federated Learning. Communication-efficient FL protocols were proposed [14,20] to improve the communication efficiency of parameter update and sharing by employing high precision vector compression mechanisms, such as structured updates and sketched updates. As more FL systems utilize a communication-efficient protocol to replace the baseline protocol, it is important to study the impact of using a communication efficient protocol on the performance of the CPL attacks, especially compared to the baseline client-to-server communication protocol. In this set of experiments, we measure the performance of CPL attacks under varying gradient compression percentage θ, i.e., θ percentage of the gradient update will be discarded in this round of gradient upload. We employ the compression method in [17] as it provides a good tradeoff between communication-efficiency and model training accuracy. It leverages sparse updates and sends only the important gradients, i.e., the gradients whose magnitude larger than a threshold, and further measures are taken to avoid losing information. Locally, the client will accumulate small gradients and only send them when the accumulation is large enough. Figure 9 shows the visualization of the comparison on MNIST and CIFAR10. We observe that compared to the baseline protocol with full gradient upload, using the communication efficient protocol with θ up to 40%, the CPL attack remains to be effective for CIFAR10. Section 4.2 provides a more detailed experimental study on CPL attacks under communication-efficient FL protocol.

3.3 Attack Effect and Cost Metrics

Our framework evaluates the adverse effect and cost of CPL attacks using the following metrics. For data-specific metrics, we average the evaluation results over all successful reconstructions.

Attack Success Rate (ASR) is the percentage of successfully reconstructed training data over the number of training data being attacked. We use ASRc and ASRl to refer to the attack success rate on content and label respectively.

MSE uses the root mean square deviation to measure the similarity between reconstructed input x_{rec} and ground-truth input x: $\frac{1}{M}\sum_{i=1}^{M}(x(i) - x_{rec}(i))^2$ when the reconstruction is successful. M denotes total number of features in the input. MSE can be used on all data format, e.g. attributes and text. A smaller MSE means the more similar reconstructed data to the private ground truth.

SSIM measures the structural similarity between two images based on a perception-based model [28] that considers image degradation as perceived change.

Attack Iteration measures the number of attack iterations required for reconstruction learning to converge and thus succeed the attack, e.g., L_2 distance of the gradients between the reconstructed data and the local private training data is smaller than a pre-set threshold.

4 Experiments and Results

4.1 Experiment Setup

We evaluate CPL attacks on four benchmark datasets: MNIST, LFW, CIFAR10, CIFAR100. MNIST consists of 70000 grey-scale hand-written digits images of size 28×28. The 60000:10000 split is used for training and testing data. Labeled Faces in the Wild (LFW) people dataset has 13233 images from 5749 classes.

The original image size is 250 × 250 and we slice it to 32 × 32 and extract the 'interesting' part. Our experiments only consider 106 classes, each with more than 14 images. For a total number of 3735 eligible LFW data, a 3:1 train-test ratio is applied. CIFAR10 and CIFAR100 both have 60000 color images of size 32 × 32 with 10 and 100 classes respectively. The 50000:10000 split is used for training and testing. We perform CPL attacks with the following configurations as the default unless otherwise stated. The initialization method is patterned, the maximum attack iterations are 300, the optimization method is L-BFGS with attack learning rate 1. The attack is performed with full gradient communication. For each dataset, the attack is performed on 100 images with 10 different random seeds. For MNIST and LFW, we use a LeNet model with 0.9568 benign accuracy on MNIST and 0.695 on LFW. For CIFAR10 and CIFAR100, we apply a ResNet20 with benign accuracy of 0.863 on CIFAR10 and CIFAR100. We use 100 clients as the total client population and at each communication round, 10% of clients will be selected randomly to participate in federated learning.

4.2 Gradient Leakage Attack Evaluation

Comparison with Other Gradient Leakage Attacks. We first conduct a set of experiments to compare the CPL-patterned attack with two existing gradient leakage attacks: the deep gradient attack [34], and the gradient inverting attack [9], which replaces the L_2 distance function with cosine similarity and performs the optimization on the sign of the gradient. We measure the attack results on the four benchmark image datasets in Table 2. For all four datasets, the CPL attack is a much faster and more efficient attack with the highest attack success rate (ASR) and the lowest attack iterations on both content and label reconstruction. Meanwhile, the high SSIM and low MSE for CPL attack indicate the quality of the reconstructed data is almost identical to the private training data. We also observe that gradient inverting attack [9] can achieve a high ASR compared to deep gradient attack [34] but at a great cost of attack iterations. Note that the CPL attack offers slightly higher ASR compared to [9] but at much lower attack cost in terms of the required attack iteration.

Table 2. Comparison of different gradient leakage attacks

	CIFAR10			CIFAR100			LFW			MNIST		
	CPL	[34]	[9]	CPL	[34]	[9]	CPL	[34]	[9]	CPL	[34]	[9]
attack iter	**28.3**	114.5	6725	**61.8**	125	6813	**25**	69.2	4527	**11.5**	18.4	3265
ASRc	**0.973**	0.754	0.958	**0.981**	0.85	0.978	**1**	0.857	0.974	**1**	0.686	0.784
ASRl	**1**	0.965	**1**	**1**	0.94	**1**	**1**	0.951	**1**	**1**	0.951	**1**
SSIM	**0.9985**	0.9982	0.9984	**0.959**	0.953	0.958	**0.998**	0.997	0.9978	**0.99**	0.985	0.989
MSE	**2.2E-04**	2.5E-04	**2.2E-04**	**5.4E-04**	6.5E-04	**5.4E-04**	**2.2E-04**	2.9E-04	2.3E-04	**1.5E-05**	1.7E-05	1.6E-05

Table 3. Comparison of different geometrical initialization in CPL attacks

dataset \ initialization		baseline	patterned		dark/light		RGB			optimal
		random	2*2	4*4	dark	light	R	G	B	insider
CIFAR10	attack iter	91.14	28.3	**24.8**	34	52.3	35.9	77.5	79.1	23.2
	ASR	0.871	0.973	**0.976**	0.99	1	0.99	0.96	0.96	1
CIFAR100	attack iter	125	61.8	57.2	**57.5**	65.3	59.4	61.3	62.4	35.3
	ASR	1	0.981	0.995	**1**	1	1	0.88	1	1
LFW	attack iter	71.1	25	18.6	34	50.8	**20.3**	28	42.1	13.3
	ASR	0.86	1	0.997	1	1	**1**	1	1	1

Variation Study: Geometrical Initialization. This set of experiments measure and compare the four types of geometrical initialization methods: patterned random, dark/light, R.G.B., and optimal. For optimal initialization, we feed a piece of data that is randomly picked from the training set. This assumption is reasonable when different clients hold part of the information about one data item. Table 3 shows the result. We observe that the performance of all four geometrical initializations is always better than the random initialization (see the bold highlight in Table 3). Note that the optimal initialization is used in this experiment as a reference point as it assumes the background information about the data distribution. Furthermore, the performance of geometrical initializations is also dataset-dependent. CPL attack on CIFAR100 requires a longer time and more iterations to succeed than CPL on CIFAR10 and LFW.

Variation Study: Batch Size and Iterations. Motivated by the batch size visualization in Fig. 5, we study the impact of hyperparameters used in local training, such as batch size and the number of local training iterations, on the performance of CPL attack. Table 4a shows the results of the CPL attack on the LFW dataset with five different batch sizes. We observe that the ASR of CPL attack is decreased to 96%, 89%, 76%, and 13% as the batch size increases to 2, 4, 8, and 16. The CPL attacks at different batch sizes are successful at the attack iterations around 25 to 26 as we measure attack iterations only on successfully reconstructed instances. Table 4b shows the results of the CPL attack under five different settings of local iterations before sharing the gradient updates. We show that as more iterations are performed at local training before sharing the gradient update, the ASR of the CPL attack is decreasing with 97%, 85%, and 39% for iterations of 5, 7, and 9 respectively. This result confirms our analysis in Sect. 3.2 that a larger batch size for local training prior to sharing the local gradient update may help mitigate the CPL attack because the shared gradient data capture more shared features among the images in the batch instead of more specific to an individual image.

Table 4. Effect of local training hyperparameters on CPL attack (LFW)

batch size	1	2	4	8	16
ASR	1	0.96	0.89	0.76	0.13
attack iter	25	25.7	25.6	26.1	25.7
SSIM	0.998	0.635	0.525	0.456	0.401
MSE	2.2E-04	6.7E-03	8.0E-03	9.0E-03	1.0E-02

(a) Effect of Batch size on CPL

local iter	1	3	5	7	9
attack iter	25	42.5	94.2	95.6	97.9
ASR	1	1	0.97	0.85	0.39
SSIM	0.998	0.981	0.898	0.659	0.471
MSE	2.2E-04	6.8E-04	1.8E-03	3.8E-03	5.5E-03

(b) Effect of local training iterations

Variation Study: Leakage in Communication Efficient Sharing. This set of experiments measures and compares the gradient leakage in CPL under baseline protocol (full gradient sharing) and communication-efficient protocol (significant gradient sharing with low-rank filer). Table 5 shows the result. To illustrate the comparison results, we provide the accuracy of the baseline protocol and the communication-efficient protocol of varying compression percentages on all four benchmark datasets in Table 5(a). We make two interesting observations. (1) CPL attack can generate high confidence reconstructions (high ASR, high SSIM, low MSE) for MNIST and CIFAR10 at compression rate 40%, and for

Table 5. Effect of CPL attack under communication-efficient FL protocols

benign acc	0	1%	10%	20%	30%	40%	50%	70%	80%	90%
LFW	0.695	0.697	0.705	0.701	0.71	0.709	0.713	0.711	0.683	0.676
CIFAR100	0.67	0.673	0.679	0.685	0.687	0.695	0.689	0.694	0.676	0.668
CIFAR10	0.863	0.864	0.867	0.872	0.868	0.865	0.868	0.861	0.864	0.859
MNIST	0.9568	0.9567	0.9577	0.957	0.9571	0.9575	0.9572	0.9576	0.9573	0.9556

(a) Benign accuracy of four datasets with varying compression rates

	compression	original	1%	10%	20%	30%	40%	50%	70%	80%	90%
LFW	attack iter	25	25	24.9	24.9	25	24.8	25	24.6	24.5	300
	ASR	1	1	1	1	1	1	1	1	1	0
	SSIM	0.998	0.9996	0.9997	0.9978	0.9978	0.9975	0.998	0.9981	0.951	0.004
	MSE	2.2E-04	1.8E-04	1.7E-04	4.9E-04	4.8E-04	5.1E-04	4.5E-04	4.6E-04	1.6E-03	1.6E-01
CIFAR100	attack iter	61.8	61.8	61.8	61.7	61.7	61.5	61.8	60.1	59.8	300
	ASR	1	1	1	1	1	1	1	1	1	0
	SSIM	0.959	0.9994	0.9981	0.9981	0.998	0.9983	0.9982	0.9983	0.895	0.016
	MSE	5.4E-04	3.3E-04	3.7E-04	3.7E-04	3.8E-04	3.5E-04	3.6E-04	3.7E-04	1.5E-03	1.2E-01
CIFAR10	attack iter	28.3	28.3	28.1	26.5	25.8	25.3	300	300	300	300
	ASR	1	1	1	1	1	1	0	0	0	0
	SSIM	0.9985	0.9996	0.9996	0.9997	0.9992	0.87	0.523	0.0017	0.0019	0.0018
	MSE	2.2E-04	1.3E-04	1.2E-04	1.2E-04	2.1E-04	3.1E-03	9.6E-03	3.3E-01	3.3E-01	3.3E-01
MNIST	attack iter	11.5	11.5	11.2	10.7	7.2	300	300	300	300	300
	ASR	1	1	1	1	1	0	0	0	0	0
	SSIM	0.99	0.9899	0.9891	0.9563	0.9289	0.8889	0.8137	0.425	0.433	0.43
	MSE	2.4E-04	2.4E-04	2.2E-04	1.7E-03	8.8E-03	2.8E-02	5.8E-02	2.7E-01	2.7E-01	2.7E-01

(b) Attack performance of four datasets with varying compression rates

Table 6. Mitigation with Gaussian noise and Laplace noise

	CIFAR100				LFW			
Gaussian noise	original	G-10e-4	G-10e-3	G-10e-2	original	G-10e-4	G-10e-3	G-10e-2
benign acc	0.67	0.664	0.647	0.612	0.695	0.692	0.653	0.636
attack iter	61.8	61.8	61.8	300	25	25	25	300
ASR	1	1	1	0	1	1	1	0
SSIM	0.9995	0.9976	0.8612	0.019	0.998	0.9976	0.8645	0.013
MSE	5.4E-04	6.9E-04	4.1E-03	3.0E-01	2.2E-04	3.7E-04	3.0E-03	1.9E-01

Laplace noise	original	L-10e-4	L-10e-3	L-10e-2	original	L-10e-4	L-10e-3	L-10e-2
benign acc	0.67	0.651	0.609	0.578	0.695	0.683	0.632	0.597
attack iter	61.8	61.8	61.8	300	25	25	25	300
ASR	1	1	1	0	1	1	1	0
SSIM	0.9995	0.9956	0.7309	0.017	0.998	0.9965	0.803	0.009
MSE	5.4E-04	6.4E-04	6.4E-03	3.1E-01	2.2E-04	4.0E-04	3.9E-03	2.0E-01

CIFAR100 and LFW at the compression rate of 90%. Second, as the compression percentage increases, the number of attack iterations to succeed the CPL attack decreases. This is because a larger portion of the gradients are low significance and are set to 0 by compression. When the attack fails, it indicates that the reconstruction cannot be done even with the infinite(∞) attack iterations, but we measure SSIM and MSE of the failed attacks at the maximum attack iterations of 300. (2) CPL attacks are more severe with more training labels in the federated learning task. A possible explanation is that informative gradients are more concentrated when there are more classes.

4.3 Mitigation Strategies

Gradient Perturbation with Additive Noise. We consider Gaussian noise and Laplace noise with zero means and different magnitude of variance in this set of experiments. Table 6 provides the mitigation results on CIFAR100 and LFW. In both cases, the client privacy leakage attack is largely mitigated at some cost of accuracy if we add sufficient Gaussian noise (G-10e-2) or Laplace noise (L-10e-2). Figure 10 illustrates the effect of the additive noise using four examples. The large additive noise we use to obfuscate the gradient while performing the reconstruction, the small SSIM and the large MSE indicate the more dissimilar between the gradient of the reconstructed data and the gradient of the original sensitive input, leading to poor quality of the CPL attack.

Gradient Squeezing with Controlled Local Training Iterations. Instead of sharing the gradient from the local training computation at each round t, we schedule and control the sharing of the gradient only after M iterations of local training. Table 7 shows the results of varying M from 1 to 10 with step 1. It shows that as M increases, the ASR of CPL attack starts to decrease, with 97.1%, 83.5%, 50% and 29.2% for M = 3, 5, 8 and 10 respectively for CIFAR10, and with 100%, 97%, 78% and 7% for M = 3, 5, 8 and 10 respectively for LFW. An

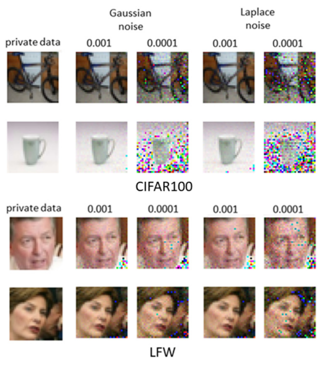

Fig. 10. Effect of additive noise

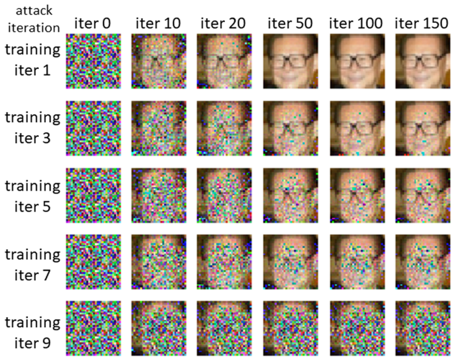

Fig. 11. Effect of local training

example of gradient squeezing with controlled local training iterations is provided in Fig. 11. This preliminary mitigation study shows that clients in federated learning may adopt some attack resilient optimizations when configuring their local training hyperparameters.

5 Related Work

Privacy in federated learning has been studied in two contexts: training-phase privacy attacks and prediction-phase privacy attacks. Gradient leakage attacks, formulated in CPL of different forms, or those in literature [9,32,34], are one type of privacy exploits in the training phase. In addition, Aono et al [1,2] proposed a privacy attack, which partially recovers private data based on the proportionality between the training data and the gradient updates in multi-layer perceptron models. However, their attack is not suitable for convolutional neural networks because the size of the features is far larger than the size of convolution weights. Hitaj et al [11] poisoned the shared model by introducing mislabeled samples into the training. In comparison, gradient leakage attacks are more aggressive since client privacy leakage attacks make no assumption on direct access to the training data as those in training poisoning attacks and yet can compromise the private training data by reconstruction attack based on only the local parameter updates to be shared with the federated server.

Privacy exploits at the prediction phase include model inversion, membership inference, and GAN-based reconstruction attack [10,11,27]. Fredrikson et al. [7] proposed the model inversion attack to exploit confidence values revealed along with predictions. Ganju et al. [8] inferred the global properties of the training data from a trained white-box fully connected neural network. Membership attacks [23,25] exploited the statistical differences between the model prediction on its training set and the prediction on unseen data to infer the membership of training data.

Table 7. Mitigation with controlled local training iterations

	local iter	1	2	3	4	5	6	7	8	9	10
CIFAR10	ASR	0.973	0.971	0.921	0.875	0.835	0.758	0.612	0.5	0.406	0.292
	SSIM	0.9985	0.9981	0.997	0.956	0.915	0.901	0.893	0.822	0.748	0.715
	MSE	2.2E-04	2.5E-04	2.9E-04	1.1E-03	2.4E-03	2.5E-03	2.7E-03	3.0E-03	4.5E-03	5.0E-03
	attack iter	28.3	29.5	31.6	35.2	42.5	71.5	115.3	116.3	117.2	117.5
CIFAR100	ASR	0.981	0.977	0.958	0.949	0.933	0.893	0.842	0.78	0.557	0.437
	SSIM	0.9959	0.996	0.996	0.959	0.907	0.803	0.771	0.666	0.557	0.505
	MSE	5.4E-04	5.8E-04	6.9E-04	1.1E-03	1.7E-03	2.2E-03	3.5E-03	4.2E-03	6.4E-03	6.9E-03
	attack iter	61.8	63.8	66.5	72.4	78.3	95.3	113.7	114.1	114.3	114.4
LFW	ASR	1	1	1	1	0.97	0.91	0.85	0.78	0.39	0.07
	SSIM	0.998	0.996	0.981	0.976	0.898	0.811	0.659	0.573	0.471	0.41
	MSE	2.2E-04	4.3E-04	6.8E-04	8.6E-04	1.8E-03	2.8E-03	3.8E-03	4.3E-03	5.5E-03	6.5E-03
	attack iter	25	34.7	42.5	68.3	94.2	95.5	95.6	98.3	97.9	98.1
MNIST	ASR	1	0.82	0.57	0.44	0.25	0.06	0	0	0	0
	SSIM	0.99	0.982	0.974	0.963	0.954	0.935	0.583	0.576	0.581	0.574
	MSE	1.5E-05	2.3E-04	2.8E-04	1.2E-03	1.5E-03	2.4E-03	1.7E-02	1.7E-02	1.7E-02	1.7E-02
	attack iter	11.5	34.7	93.2	96.7	97.1	96.5	300	300	300	300

6 Conclusion

We have presented a principled framework for evaluating and comparing different forms of client privacy leakage attacks. This framework showed that an effective mitigation method against client gradient leakage attacks should meet the two important criteria: (1) the defense can mitigate the gradient leakage attacks no matter how the attacker configures his reconstruction algorithm to launch the attack; and (2) the defense can mitigate the attacks no matter how the FL system is configured for joint training. Extensive experiments on four benchmark datasets highlighted the importance of providing a systematic evaluation framework for an in-depth understanding of the various forms of client privacy leakage threats in federated learning and for developing and evaluating different mitigation strategies.

Acknowledgements. The authors acknowledge the partial support from NSF CISE SaTC 1564097, NSF 2038029 and an IBM Faculty Award.

7 Appendices

7.1 Proof of Theorem 1

Assumption 1 ***(Convexity).*** *we say $f(x)$ is convex if*

$$f(\alpha x + (1-\alpha)x') \leq \alpha f(x) + (1-\alpha) f(x'), \tag{3}$$

where x, x' are data point in $\mathbb{R}^d$, and $\alpha \in [0,1]$.

Lemma 1. *If a convex $f(x)$ is differentiable, we have:*

$$f(x') - f(x) \geq \langle \nabla f(x), x' - x \rangle. \tag{4}$$

Proof. Equation 3 can be rewritten as: $\frac{f(x'+\alpha(x-x'))-f(x')}{\alpha} \leq f(x) - f(y)$. When $\alpha \to 0$, we complete the proof.

Assumption 2 ***(Lipschitz Smoothness).*** *With Lipschitz continuous on the differentiable function $f(x)$ and Lipschitz constant L, we have:*

$$||\nabla f(x) - \nabla f(x') \leq L||x - x'||, \tag{5}$$

Lemma 2. *If $f(x)$ is Lipschitz-smooth, we have:*

$$f(x^{t+1}) - f(x^t) \leq -\frac{1}{2L}||\nabla f(x^T)||_2^2 \tag{6}$$

Proof. Using the Taylor expansion of $f(x)$ and the uniform bound over Hessian matrix, we have

$$f(x') \leq f(x) + \langle \nabla f(x), x' - x \rangle + \frac{L}{2}||x' - x||_2^2. \tag{7}$$

By inserting $x' = x - \frac{1}{L}\nabla f(x)$ into Eq. 5 and Eq. 7, we have:

$$f(x - \frac{1}{L}\nabla f(x)) - f(x) \leq -\frac{1}{L}\langle \nabla f(x), \nabla f(x) \rangle + \frac{L}{2}||\frac{1}{L}\nabla f(x)||_2^2 = -\frac{1}{2L}||\nabla f(x)||_2^2$$

Lemma 3 ***(Co-coercivity).*** *A convex and Lipschitz-smooth $f(x)$ satisfies:*

$$\langle \nabla f(x') - \nabla f(x), x' - x \rangle \geq \frac{1}{L}||\nabla f(x') - \nabla f(x)|| \tag{8}$$

Proof. Due to Eq. 5,

$$\langle \nabla f(x') - \nabla f(x), x' - x \rangle \geq \langle \nabla f(x') - \nabla f(x), \frac{1}{L}(\nabla f(x') - \nabla f(x)) \rangle = \frac{1}{L}||\nabla f(x') - \nabla f(x)||$$

Then we can proof the attack convergence theorem: $f(x^T) - f(x^*) \leq \frac{2L||x^0 - x^*||^2}{T}$.

Proof. Let $f(x)$ be convex and Lipschitz-smooth. It follow that

$$\begin{aligned} ||x^{t+1} - x^*||_2^2 &= ||x^t - x^* - \frac{1}{L}\nabla f(x^t)||_2^2 \\ &= ||x^t - x^*||_2^2 - 2\frac{1}{L}\langle x^t - x^*, \nabla f(x^t) \rangle + \frac{1}{L^2}||\nabla f(x^t)||_2^2 \\ &\leq ||x^t - x^*||_2^2 - \frac{1}{L^2}||\nabla f(x^t)||_2^2 \end{aligned} \tag{9}$$

Equation 9 holds due to Eq. 8 in Lemma 3. Recall Eq. 6 in Lemma 2, we have:

$$f(x^{t+1}) - f(x^*) \leq f(x^t) - f(x^*) - \frac{1}{2L}||\nabla f(x^t)||_2^2. \tag{10}$$

By applying convexity,

$$\begin{aligned} f(x^t) - f(x^*) &\leq \langle \nabla f(x^t), x^t - x^* \rangle \\ &\leq ||\nabla f(x^t)||_2 ||x^t - x^*|| \\ &\leq ||\nabla f(x^t)||_2 ||x^1 - x^*||. \end{aligned} \tag{11}$$

Then we insert Eq. 11 into Eq. 10:

$$f(x^{t+1}) - f(x^*) \leq f(x^t) - f(x^*) - \frac{1}{2L}\frac{1}{||x^1 - x^*||^2}(f(x^t) - f(x^*))^2$$

$$\Rightarrow \frac{1}{f(x^t) - f(x^*)} \leq \frac{1}{f(x^{t+1}) - f(x^*)} - \beta \frac{f(x^t) - f(x^*)}{f(x^{t+1}) - f(x^*)} \tag{12}$$

$$\Rightarrow \frac{1}{f(x^t) - f(x^*)} \leq \frac{1}{f(x^{t+1}) - f(x^*)} - \beta \tag{13}$$

$$\Rightarrow \beta \leq \frac{1}{f(x^{t+1}) - f(x^*)} - \frac{1}{f(x^t) - f(x^*)}, \tag{14}$$

where $\beta = \frac{1}{2L}\frac{1}{||x^1 - x^*||^2}$. Equation 12 is done by divide both side with $(f(x^{t+1}) - f(x^*))(f(x^t) - f(x^*))$ and Eq. 13 utilizes $f(x^{t+1}) - f(x^*) \leq f(x^t) - f(x^*)$. Then, following by induction over $t = 0, 1, 2, ..T - 1$ and telescopic cancellation, we have

$$T\beta \leq \frac{1}{f(x^T) - f(x^*)} - \frac{1}{f(x^0) - f(x^*)} \leq \frac{1}{f(x^T) - f(x^*)}.$$

$$T\beta \leq \frac{1}{f(x^T) - f(x^*)} - \frac{1}{f(x^0) - f(x^*)} \leq \frac{1}{f(x^T) - f(x^*)} \tag{15}$$

$$\Rightarrow \frac{T}{2L}\frac{1}{||x^1 - x^*||^2} \leq \frac{1}{f(x^T) - f(x^*)} \tag{16}$$

$$\Rightarrow f(x^T) - f(x^*) \leq \frac{2L||x^0 - x^*||^2}{T}. \tag{17}$$

Thus complete the proof.

References

1. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning: revisited and enhanced. In: Batten, L., Kim, D.S., Zhang, X., Li, G. (eds.) ATIS 2017. CCIS, vol. 719, pp. 100–110. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-5421-1_9
2. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2017)

3. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. arXiv preprint arXiv:1807.00459 (2018)
4. Battiti, R.: First-and second-order methods for learning: between steepest descent and Newton's method. Neural Comput. **4**(2), 141–166 (1992)
5. Bonawitz, K., et al.: Towards federated learning at scale: System design. In: Proceedings of the 2nd SysML Conference, pp. 619–633 (2018)
6. Fletcher, R.: Practical Methods of Optimization. Wiley, Hoboken (2013)
7. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1322–1333 (2015)
8. Ganju, K., Wang, Q., Yang, W., Gunter, C.A., Borisov, N.: Property inference attacks on fully connected neural networks using permutation invariant representations. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 619–633 (2018)
9. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients-how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053 (2020)
10. Hayes, J., Melis, L., Danezis, G., De Cristofaro, E.: Logan: evaluating privacy leakage of generative models using generative adversarial networks. arXiv preprint arXiv:1705.07663 (2017)
11. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 603–618 (2017)
12. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. In: Technical report (2008)
13. Kamp, M., et al.: Efficient decentralized deep learning by dynamic model averaging. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11051, pp. 393–409. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10925-7_24
14. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. In: NIPS Workshop on Private Multi-Party Machine Learning (2016)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. In: Technical report (2009)
16. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist10, 34 (1998)
17. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: reducing the communication bandwidth for distributed training. In: International Conference on Learning Representations (2018)
18. Liu, W., Chen, L., Chen, Y., Zhang, W.: Accelerating federated learning via momentum gradient descent. IEEE Trans. Parallel Distrib. Syst. (2020)
19. Ma, C., et al.: Adding vs. averaging in distributed primal-dual optimization. In: Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 1973–1982 (2015)
20. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
21. McMahan, B., Ramage, D.: Federated learning: Collaborative machine learning without centralized training data. Google Res. Blog **3** (2017)

22. McMahan, H.B., Moore, E., Ramage, D., Arcas, B.A.: Federated learning of deep networks using model averaging. corr abs/1602.05629 (2016). arXiv preprint arXiv:1602.05629 (2016)
23. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 691–706. IEEE (2019)
24. Rossi, F., Gégout, C.: Geometrical initialization, parametrization and control of multilayer perceptrons: application to function approximation. In: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN 1994), vol. 1, pp. 546–550. IEEE (1994)
25. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE (2017)
26. Vanhaesebrouck, P., Bellet, A., Tommasi, M.: Decentralized collaborative learning of personalized models over networks. In: Artificial Intelligence and Statistics (2017)
27. Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., Qi, H.: Beyond inferring class representatives: user-level privacy leakage from federated learning. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 2512–2520. IEEE (2019)
28. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
29. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
30. Yao, X., Huang, T., Zhang, R.X., Li, R., Sun, L.: Federated learning with unbiased gradient aggregation and controllable meta updating. arXiv preprint arXiv:1910.08234 (2019)
31. Zhang, Q., Benveniste, A.: Wavelet networks. IEEE Trans. Neural Netw. **3**(6), 889–898 (1992)
32. Zhao, B., Mopuri, K.R., Bilen, H.: iDLG: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610 (2020)
33. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. arXiv preprint arXiv:1806.00582 (2018)
34. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems, pp. 14747–14756 (2019)