# A Framework for Evaluating Gradient Leakage Attacks 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@gatech.edu, ling.liu@cc.gatech.edu,
margaret.loper@gtri.gatech.edu
{staceytruex, memregursoy, khchow,yanzhaowu}@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 also 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 [37,26,40,19,5,44]. 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 [17].

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 [10,45,43]. By intercepting the local gradient update shared by a client to the FL server before performing federated averaging [28,25,17], 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 Section 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.

The rest of the paper is organized as follows: Section 2 presents the overview of the problem statement, including defining federated learning to establish some terminology required for our attack analysis, the threat model that describes the baseline assumptions about clients, FL server and adversaries, as well as the general formulation of client privacy leakage attack. Section 3 describes three main components of our evaluation framework: the impact of attack parameter configurations on attack effectiveness and cost, the impact of FL hyperparameter configurations on attack effectiveness and cost, and the quantitative measurement metrics for attack effect and cost evaluation. Section 4 reports the measurement study and experimental analysis results on four benchmark image datasets. We conclude the paper with related work and a summary.

## 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 [27], 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 $\nabla w$ 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:

**Distributed SGD.** At each round, each of the $K_t$ clients trains the local model with the local data and uploads the local gradients to the FL server. The server iteratively aggregates the local gradients from all $K_t$ clients into the global model, and check if the convergence condition of FL task is met and if not, it starts the next iteration round [23,24,40,41].

$$w(t+1) = w(t) - \eta \sum\nolimits_{k=1}^{K_t} \frac{n_t}{n} \nabla w_k(t),$$

where $\eta$ is the global learning rate and $\frac{n_t}{n}$ is the weight of client $k$. Here we adopt the same notation as in reference [26] so that $n_k$ is the number of data points at client $k$ and $n$ indicates the amount of total data from all participating clients at round $t$.

**Federated averaging.** At each round, each of the $K_t$ clients 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, and starts the next iteration round $t+1$ unless it reaches the convergence [3,26].

$$w(t+1) = \sum\nolimits_{k=1}^{K_t} \frac{n_t}{n} w_k(t+1).$$

Let $\Delta w_k(t)$ denote the difference between the model parameter update before the iteration of training and the model parameter update after the training for client $k$. Below is a variant of this method [11]:

$$w(t+1) = w(t) + \sum\nolimits_{k=1}^{K_t} \frac{n_t}{n} \Delta w_k(t).$$

**Efficiency of FL Communication Protocol.** The baseline communication protocol is used in many early federated learning implementations: the client sends a full vector of local training parameter update back 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. The communication efficient FL protocols have been proposed [26,19], which improves communication-efficiency of parameter update sharing by employing high precision vector compression mechanisms, such as structured updates and sketched updates. The

former directly learns an update from a pre-specified structure, such as a low-rank matrix and random masks. The latter compresses the learned full vector of model parameter update to ensure a high compression ratio with a low-value loss before sending it to the server.

Our framework will study the impact of different configurations of FL hyper-parameters on the success rate and cost of privacy leakage attacks. For instance, we will show that both baseline protocol and communication efficient protocol (e.g., sketched updates) are vulnerable to client gradient leakage attacks (see Section 3.2 and Section 4.2).

## 2.2   Threat Model

In an FL system, clients are the most vulnerable attack surface for the client privacy leakage (CPL) attack because the client is the one sending its local training parameter update to the FL server. We assume that clients can be compromised in a limited manner: an adversary cannot gain access to the private training data but may intercept the local parameter update to be sent to the FL server and be able to access and run the saved local model executable (checkpoint data) on the compromised client to launch white-box gradient leakage attack.

On the other hand, we assume that the federated server is honest but curious. Namely, the FL server will honestly 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. It is worth noting that even if the network connection between client and server is secure, the client privacy leakage attack can happen on a compromised client before the local parameter update is prepared for upload to the server.

Finally, distributed training can happen in either federated scenario with a central server [22] or a decentralized scenario where clients are connected via a peer to peer network [33]. In the decentralized setting, a client node will first perform the local computation to update its local model parameter and then sends the updated gradients to its neighbor nodes. The client privacy leakage attack can happen in both scenarios on the compromised client nodes. The federated scenario is assumed in this paper.

## 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 gives a sketch of the client privacy leakage attack method.

---

**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: $x_{rec}^0 \leftarrow INIT(x.type)$
9: $y_{rec} \leftarrow \arg\min_i(\nabla_i w_k(t))$
10: **for** $\tau$ in $\mathbb{T}$ **do**
11:     $\nabla w_{att}^\tau(t) \leftarrow \frac{\partial loss(f(x_{rec}^\tau, w(t)), y_{rec})}{\partial w(t)}$
12:     $D^\tau \leftarrow ||\nabla w_{att}^\tau(t) - \nabla w_k(t)||^2 + \alpha||f(x_{rec}^\tau, w(t)) - y_{rec}||^2$
13:     $x_{rec}^{\tau+1} \leftarrow x_{rec}^\tau - \eta' \frac{\partial D^\tau}{\partial x_{rec}^\tau}$
14: **end for**

---

In Algorithm 1, 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 $\eta_k$ 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 Section 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, and 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 [45], in a similar way as the content reconstruction optimization. Figure 1 provides a visualization of four illustrative example attacks over four datasets: LFW [15], CIFAR100 [20], MNIST [21], and CIFAR10 [20].
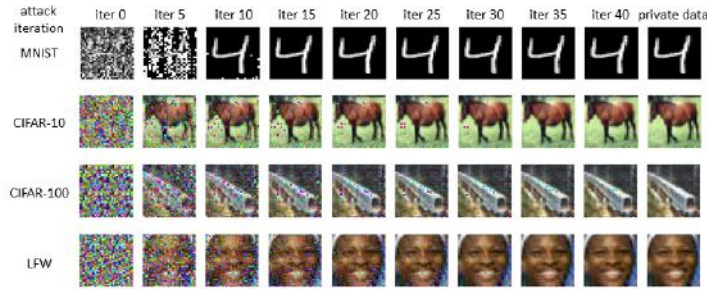


Fig. 1: Example illustration of the Client Privacy Leakage attack

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 [45,43,10], this 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 [12]. 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 the CPL Attack Convergence Theorem, 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 [32] 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 [42], the following partial derivative of the

geometrical initialization.

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

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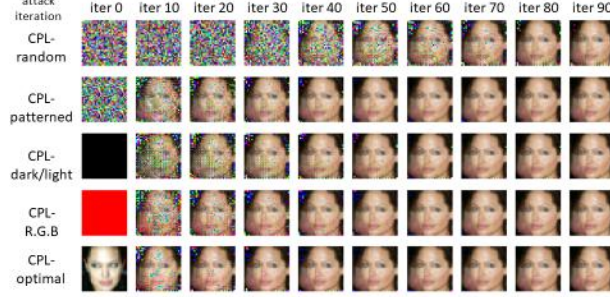


Fig. 2: Visualization of different initialization

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 Figure 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 theoretical optimal initialization in terms of reconstruction effectiveness.

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 Figure 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 Section 2.3 and Algorithm 1, there are two decision factors for termination. One is the maximum attack iteration and the other is the $L_2$-distance threshold of the gradient loss,
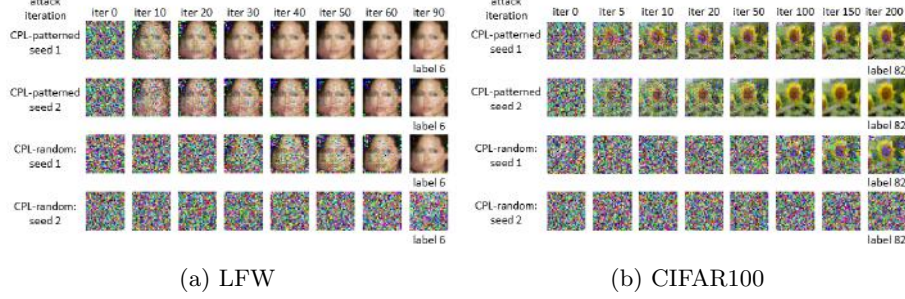
(a) LFW                    (b) CIFAR100

Fig. 3: Effect of different random seed

i.e., the difference between the gradient from the reconstructed data and the actual gradient from the local training using the private local data.

Consider the configuration of the maximum number of attack iterations, Table 1 compares the six different settings of attack iteration termination condition. It shows that when it is too small, such as 10 or 20, no matter how to optimize the attack initialization method, the CPL attack will fail for all or most of the datasets. However, when it is set to sufficiently larger, say 100 or 300, choosing a good attack initialization method matters more significantly. For example, CPL-patterned can get an attack success rate higher than the CPL-random for all three datasets. Also, CPL-random shows unstable attack performance: For LFW, it has slightly higher ASR when choosing the termination of 300 over that of 100 iterations. However, for CIFAR10 and CIFAR100, it has significantly higher ASR when choosing the termination of 300 iterations over that of 100. This set of experiments also shows the good configuration of the total number of termination iterations can be a challenging problem.

The second factor used for setting the attack termination condition is the $L_2$-distance threshold of the gradient difference between the reconstructed data and the private local data. This factor is dataset-dependent, and our experiments with the four benchmark datasets show that a $L2$-distance threshold, such as 0.0001, is a good option in terms of generalization.

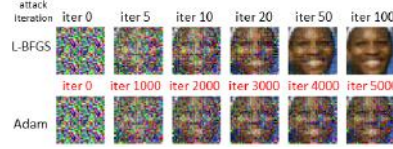| maximum attack iteration | | 10 | 20 | 30 | 50 | 100 | 300 |
|---|---|---|---|---|---|---|---|
| LFW | CPL-patterned | 0 | 0.34 | 0.98 | 1 | 1 | 1 |
| | CPL-random | 0 | 0 | 0 | 0.562 | 0.823 | 0.857 |
| CIFAR10 | CPL-patterned | 0 | 0.47 | 0.93 | 0.973 | 0.973 | 0.973 |
| | CPL-random | 0 | 0 | 0 | 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 |

Table 1: Effect of termination condition



Fig. 4: Effect of attack optimization

**Gradient Loss (Distance) Function:** In the CPL attack Algorithm 1, we use $L_2$ distance function as the gradient loss function. There are other alternative vector distance functions, such cosine similarity, entropy, and so forth, can be applied. [16] has studied the impact of different distance functions on training efficiency. It shows that using some loss functions might lead to slower training,

while others can be more robust to noise in the training set labeling and also slightly more robust to noise in the input space.

**Attack optimization:** Optimization methods, such as Stochastic Gradient descent [31], Momentum [30], Adam [18], and Adagrad [6] 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 [7] and Adam and their effects on the CPL-patterned attack for LFW dataset. It shows that choosing an appropriate optimizer can significantly improve attack effectiveness. In the rest of the paper, L-BFGS is used in our experiments.

### 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 or two classes, which is often the case for mobile devices and the non-i.i.d distribution of the training data [44], 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.
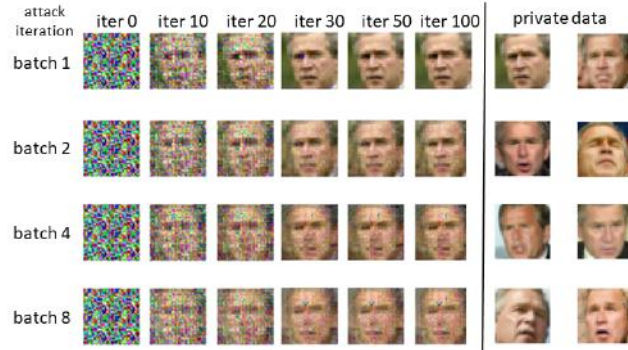


Fig. 5: Effect of batch size in CPL-patterned attacks on LFW

**Training Data Resolution:** In contrast to the early work [45] that fails to attack images of resolution higher than $64 \times 64$, 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 $32 \times 32, 64 \times 64$, and $128 \times 128$. CPL-random requires a much larger number of attack iterations in order 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 $512 \times 512$ Indiana University Chest X-Rays image of very high resolution in Figure 7.
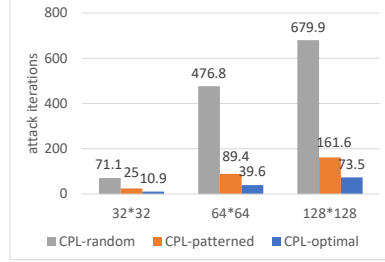


Fig. 6: Effect of data scaling (LFW)
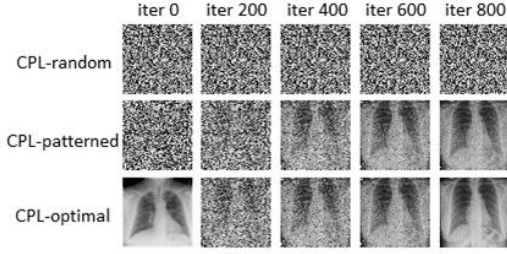


Fig. 7: Attacking $512 \times 512$ X-ray image
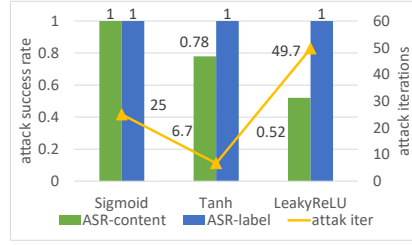


(a) LFW
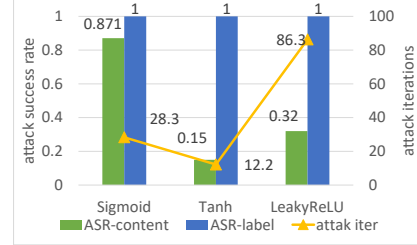


(b) CIFAR10
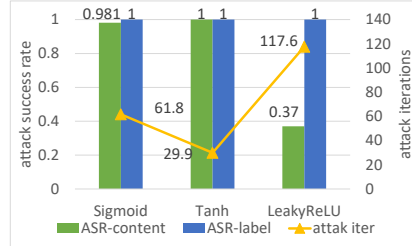


(c) CIFAR100
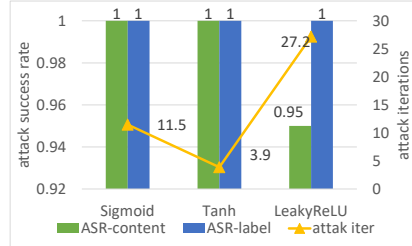


(d) MNIST

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

**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. 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.

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.

**Baseline v.s. Communication-efficient Protocols:** We have discussed the communication-efficient parameter update protocol using low-rank filers in Section 2.3. 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 $\theta$, i.e., $\theta$ percentage of the gradient update will be discarded in this round of gradient upload. We employ the compression method in [23] as it provides a good trade-off 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 baseline protocol with full gradient upload, using the communication efficient protocol with $\theta$ up to 40%, the CPL attack remains to be effective at the maximum attack iterations of 26 with 100% attack success rate for CIFAR10.

### 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 $x$: $\frac{1}{M}\sum_{i=1}^{M}(x(i) - x_{rec}(i))^2$
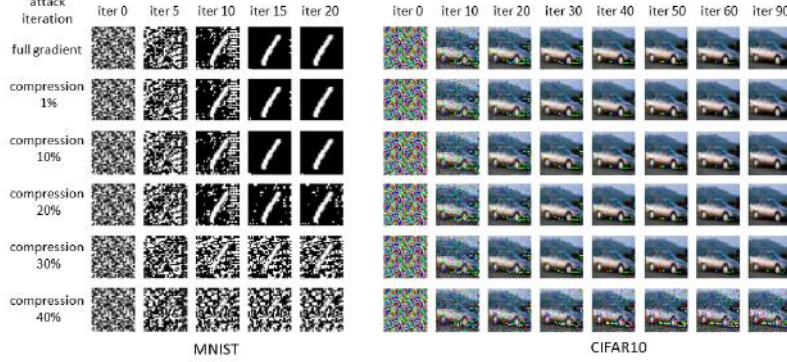
Fig. 9: Illustration of the CPL attack under communication-efficient update

when the reconstruction is successful. $M$ denotes total number of features in the input. MSE can be used on all data format such as 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 [39] that considers image degradation as perceived change.

$$SSIM(x, x') = \frac{(2\mu_x \mu_{x'} + c_1)(2\sigma_{xx'} + c_2)}{(\mu_x^2 + \mu_{x'}^2 + c_1)(\sigma_x^2 + \sigma_{x'}^2 + c_2)},$$

where $\mu_x$ and $\mu_{x'}$ are the average of $x$ and $x'$, $\sigma_x^2$ and $\sigma_{x'}^2$ are the variance of $x$ and $x'$. $\sigma_{xy}$ is the covariance of $x$ and $x'$. $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator. $L$ is the dynamic range of the pixel-values. $k_1 = 0.01$ and $k_2 = 0.03$ are constant by default. We use SSIM to evaluate all image datasets. The closer SSIM to 1, the better the attack quality in terms of image reconstruction. SSIM is designed to improve on traditional methods such as MSE on image similarity.

**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 image datasets: MNIST, LFW, CIFAR10, CIFAR100. MNIST consists of 70000 grey-scale hand-written digits images of size $28 \times 28$. 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 $250 \times 250$ and we slide it to $32 \times 32$ to 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 each consists of 60000 color images of size $32 \times 32$ with 10 classes and 100 classes respectively. The 50000:10000 split is used for training and testing.

We perform CPL attacks with the following attack configurations as the default unless otherwise stated. The initialization method is patterned, the maximum attack iteration is 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 the 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 [45], and the gradient inverting attack [10], which replaces the $L_2$ distance function with cosine similarity and performs the optimization on the sign of the gradient. We first measure the attack results on the four benchmark image datasets. Table 2 shows that CPL is a much faster and more efficient attack with the highest attack success rate (ASR) and lowest attack iterations on both content and label reconstruction for all four datasets. Also, the very high SSIM and low MSE for CPL indicate the quality of the reconstructed data is almost identical to the private training data. We also observe that gradient inverting attack [10] can lead to high ASR compared to deep gradient attack [45] using $L_2$ distance but at a great cost of attack iterations. Note that CPL using $L_2$ distance offers slightly higher ASR compared to [10] but at much lower attack cost in terms of attack iterations required.

| | CIFAR10 | | | CIFAR100 | | | LFW | | | MNIST | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPL | [45] | [10] | CPL | [45] | [10] | CPL | [45] | [10] | CPL | [45] | [10] |
| 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 2: Comparison of Different Gradient Leakage Attacks

In addition, we include two attribute datasets: UCI Adult Income and Breast Cancer Wisconsin in our comparison experiments. UCI Adult dataset includes 48842 records with 14 attributes such as age, gender, education, marital status, occupation, working hours, and native country. The binary classification task is to predict if a person makes over $50K a year based on the census attributes. Breast Cancer Wisconsin has 569 records with 32 attributes that are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These

features describe characteristics of the cell nuclei present in the image. The task is to identify if the record indicates benign or malignant cancer. For two attributes dataset, a multi-layer-perceptron with one hidden layer is used and has an accuracy of 0.8646 on UCI Adult and 0.986 on Breast Cancer Wisconsin. Categorical features in the two attribute datasets are processed with one-hot encoding. Table 3 shows the results. For the breast cancer dataset, similar observation is found compared to Table 2. CPL is the most effective attack with the highest ASR, and the gradient inverting attack [10] is the second with ASRc of 78% and ASRl of 94%, compared to the deep gradient attack [45] with ASRc of 35% and ASRl of 56%. For the UCI Adult dataset, all three have good and similar attack performance.

| UCI Adult | CPL | [45] | [10] |
|---|---|---|---|
| ASRc | 1 | 0.99 | 1 |
| ASRl | 1 | 1 | 1 |
| MSE | 1.82E-04 | 3.23E-04 | 4.97E-03 |

| Breast Cancer | CPL | [45] | [10] |
|---|---|---|---|
| ASRc | 1 | 0.35 | 0.78 |
| ASRl | 1 | 0.56 | 0.94 |
| MSE | 4.61E-04 | 6.15E-04 | 6.29E-04 |

Table 3: Comparison of gradient leakage attacks on two attribute datasets

**Comparison with other training data inference attack.** This set of experiments compares client privacy leakage attacks with two existing training data inference attacks: Melis et al [29] is based on membership-attack [34,35] and can make partial property inference on a carefully selected subset of data. Aono et al [2] is an attack that only works on the first layer of a multi-layer perceptron and can synthesize data that is proportional to the private training data. Figure 10 shows the comparison result.As the gap of the MSE magnitude is too large, the results are plotted in the log scale. It shows that the adverse effect of the CPL attack is the most detrimental with the hightest SSIM and lowest MSE, followed by Aono et al as the second, and Melis et al has much smaller SSIM and much larger MSE, indicating the low attack inference quality.



(a) SSIM

(b) MSE in log scale
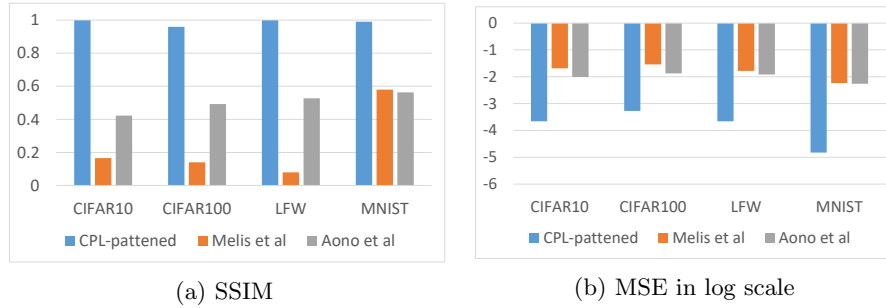
Fig. 10: Comparison of CPL attack with Melis [29] and Aono [2]

**Variation Study: Geometrical Initialization.** This set of experiments measure and compare the four types of geometrical initialization methods: patterned random, dark/light, RGB, 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 4 shows the result. We observe that the performance of all four

geometrical initializations is always better than the random initialization. Note that the optimal initialization is used in this experiment as a theoretical optimal 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.

| 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 |

Table 4: Comparison of different geometrical initialization in CPL attacks
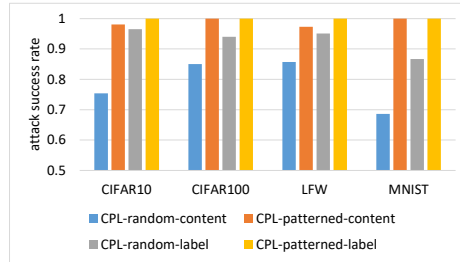


Fig. 11: Effect of Attack Initialization

| attack iter variance | | avg | min | max | variance |
|---|---|---|---|---|---|
| MNIST | CPL-patterned | 11.5 | 7 | 19 | 3.844 |
| | CPL-random | 18.4 | 13 | 65 | 12.27 |
| LFW | CPL-patterned | 25 | 18 | 125 | 14.32 |
| | CPL-random | 69.2 | 53 | 186 | 25.87 |
| CIFAR10 | CPL-patterned | 28.3 | 17 | 86 | 18.11 |
| | CPL-random | 114.5 | 67 | 272 | 39.31 |
| CIFAR100 | CPL-patterned | 61.8 | 24 | 133 | 25.3 |
| | CPL-random | 125 | 70 | 240 | 41.2 |

Table 5: Variance of attack iterations

| 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

Table 6: Effect of local training hyperparameters on CPL attack (LFW)

Figure 11 compares the four geometric initialization methods with respect to the effectiveness of CPL attack on four benchmark datasets. Table 5 measures the number of iterations to succeed the CPL attack in terms of average, min, max, and variance for each of the four datasets. We observe from Figure 11 that the geometrical initialization can largely increase the attack success rate in addition to faster attack convergence. Furthermore, the attack success rate on the label reconstruction is consistently higher than the attack success rate on content reconstruction, which further confirms the observation in Figure 3 (last row for LFW and CIFAR100). It indicates that separating the content reconstruction attack and label reconstruction attack during the CPL attack iterations, as done in CPL (Algorithm 1), is another factor contributing to high success rate and low cost (fewer attack iterations) compared to optimizing the content attack and label attack simultaneously [45] (recall Table 2 and Table 3).

**Variation Study: batch size and iterations.** Motivated by the batch size visualization in Figure 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 the CPL attack. Table 6a 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 6b 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 inspires us to add hyperparameter optimization as one of the mitigation strategies for evaluation in Section 4.3.

| 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 7: Effect of CPL attack under communication-efficient FL protocols

**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 7 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 7(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 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($\infty$) 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

Motivated by our comprehensive analysis of CPL attacks of different forms, we next evaluate two attack mitigation strategies: gradient perturbation and gradient squeezing.

**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 8 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), which causes small SSIM and large MSE, showing poor quality of reconstruction attack. Visualization of two examples from each dataset is given in Figure 12.

| | 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 |

Table 8: Mitigation with Gaussian noise and Laplace noise

**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. We perform CPL attack after each local iteration. Table 9 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%,
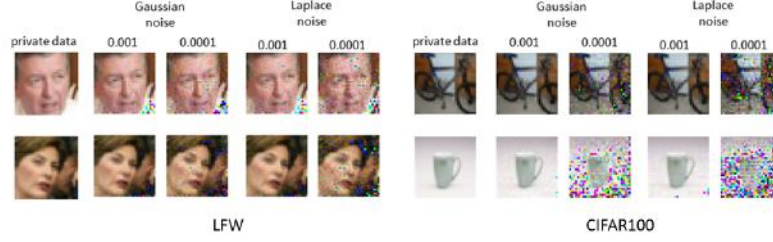
Fig. 12: Effect of additive noise on CPL-patterned attacks

| | 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 |

Table 9: Mitigation with controlled local training iterations

97%, 78% and 7% for $M = 3$, 5, 8 and 10 respectively for LFW. This preliminary mitigation study shows that clients in federated learning may adopt some attack resilient optimizations when configuring their local training hyperparameters. An example of gradient squeezing with controlled local training iterations is provided in Figure 13.

## 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 [10,45,43], are one type of privacy exploits in the training phase. In addition, Aono et al [2,1] 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 [14] 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
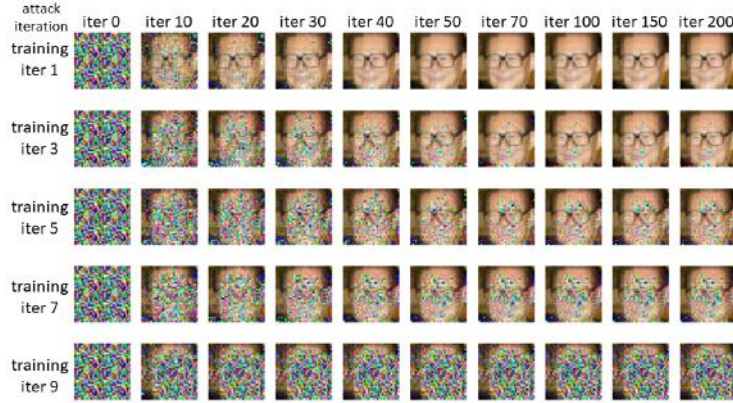
Fig. 13: Effect of local training on CPL-patterned attacks using LFW

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 [13,14,38]. Fredrikson et al [8] proposed the model inversion attack to exploit confidence values revealed along with predictions. Ganju et al [9] infers the global properties of the training data from a trained white-box fully connected neural network. Membership attacks [34,29,36] exploit the statistical differences between the model prediction on its training set and the prediction on unseen data to infer the membership of training data with high confidence.

## 6    Conclusion

We have presented a principled framework for evaluating and comparing different forms of client privacy leakage attacks. We show how adversaries can reconstruct the private local training data by simply analyzing the shared parameter update (e.g., local gradient or weight update vector). We identify and analyze the impact of different attack configurations and different hyperparameter settings of federated learning on client privacy leakage attacks. Extensive experiments on four benchmark datasets highlight 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.

# References

1. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning: Revisited and enhanced. In: International Conference on Applications and Techniques in Information Security. pp. 100–110. Springer (2017)
2. Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security **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 computation **4**(2), 141–166 (1992)
5. Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H.B., et al.: Towards federated learning at scale: System design. In: Proceedings of the 2nd SysML Conference. pp. 619–633 (2018)
6. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research **12**(Jul), 2121–2159 (2011)
7. Fletcher, R.: Practical methods of optimization. John Wiley & Sons (2013)
8. 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)
9. 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)
10. 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)
11. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017)
12. Gower, R.M.: Convergence theorem of gradient descent. Lecture notes for Statistical Optimization (2018)
13. 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)
14. 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)
15. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In: Technical report (2008)
16. Janocha, K., Czarnecki, W.M.: On loss functions for deep neural networks in classification. arXiv preprint arXiv:1702.05659 (2017)
17. Kamp, M., Adilova, L., Sicking, J., Hüger, F., Schlicht, P., Wirtz, T., Wrobel, S.: Efficient decentralized deep learning by dynamic model averaging. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 393–409. Springer (2018)

18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)
19. 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)
20. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. In: Technical report (2009)
21. LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits, 1998. URL http://yann. lecun. com/exdb/mnist **10**,  34 (1998)
22. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., Ahmed, A., Josifovski, V., Long, J., Shekita, E.J., Su, B.Y.: Scaling distributed machine learning with the parameter server. In: 11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14). pp. 583–598 (2014)
23. 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)
24. Liu, W., Chen, L., Chen, Y., Zhang, W.: Accelerating federated learning via momentum gradient descent. IEEE Transactions on Parallel and Distributed Systems (2020)
25. Ma, C., Smith, V., Jaggi, M., Jordan, M.I., Richtárik, P., Takáč, M.: Adding vs. averaging in distributed primal-dual optimization. In: Proceedings of the 32nd International Conference on Machine Learning-Volume 37. pp. 1973–1982 (2015)
26. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282 (2017)
27. McMahan, B., Ramage, D.: Federated learning: Collaborative machine learning without centralized training data. Google Research Blog **3** (2017)
28. McMahan, H.B., Moore, E., Ramage, D., y Arcas, B.A.: Federated learning of deep networks using model averaging. corr abs/1602.05629 (2016). arXiv preprint arXiv:1602.05629 (2016)
29. 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)
30. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural networks **12**(1), 145–151 (1999)
31. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
32. 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'94). vol. 1, pp. 546–550. IEEE (1994)
33. Roy, A.G., Siddiqui, S., Pölsterl, S., Navab, N., Wachinger, C.: Braintorrent: A peer-to-peer environment for decentralized federated learning. arXiv preprint arXiv:1905.06731 (2019)
34. 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)
35. Truex, S., Liu, L., Gursoy, M.E., Yu, L., Wei, W.: Towards demystifying membership inference attacks. arXiv preprint arXiv:1807.09173 (2018)

36. Turex, S., Liu, L., Chow, K.H., Gursoy, M.E., Wei, W.: Ldp-fed: Federated learning with local differential privacy. In: The 3rd International Workshop on Edge Systems, Analytics and Networking. ACM (2020)
37. Vanhaesebrouck, P., Bellet, A., Tommasi, M.: Decentralized collaborative learning of personalized models over networks. In: Artificial Intelligence and Statistics (2017)
38. 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)
39. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
40. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
41. 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)
42. Zhang, Q., Benveniste, A.: Wavelet networks. IEEE transactions on Neural Networks **3**(6), 889–898 (1992)
43. Zhao, B., Mopuri, K.R., Bilen, H.: idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610 (2020)
44. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
45. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems. pp. 14747–14756 (2019)

# 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 $f(x)$, we have:*

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

*where $L$ is called Lipschitz constant.*

**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 equation 5 and equation 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 equation 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

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

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

$$\leq ||x^t - x^*||_2^2 - \frac{1}{L^2}||\nabla f(x^t)||_2^2 \tag{9}$$

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

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

By applying convexity,

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

Then we insert equation 11 into equation 10:

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

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

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

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

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 Equation 13 utilizes $f(x^{t+1}) - f(x^*) \le f(x^t) - f(x^*)$. Then, following by induction over $t = 0, 1, 2, ..T - 1$ and telescopic cancellation, we have

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

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

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

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

Thus complete the proof.