

# More than Enough is Too Much: Adaptive Defenses against Gradient Leakage in Production Federated Learning

Fei Wang, Ethan Hugh, Baochun Li  
Department of Electrical and Computer Engineering  
University of Toronto

**Abstract**—With increasing concerns on privacy leakage from gradients, a variety of attack mechanisms emerged to recover private data from gradients at an honest-but-curious server, which challenged the primary advantage of privacy protection in federated learning. However, we cast doubt upon the real impact of these gradient attacks on production federated learning systems. By taking away several impractical assumptions that the literature has made, we find that gradient attacks pose a limited degree of threat to the privacy of raw data.

Through a comprehensive evaluation on existing gradient attacks in a federated learning system with practical assumptions, we have systematically analyzed their effectiveness under a wide range of configurations. We present key priors required to make the attack possible or stronger, such as a narrow distribution of initial model weights, as well as inversion at early stages of training. We then propose a new lightweight defense mechanism that provides *sufficient* and *self-adaptive* protection against time-varying levels of the privacy leakage risk throughout the federated learning process. As a variation of gradient perturbation method, our proposed defense, called OUTPOST, selectively adds Gaussian noise to gradients at each update iteration according to the Fisher information matrix, where the level of noise is determined by the privacy leakage risk quantified by the spread of model weights at each layer. To limit the computation overhead and training performance degradation, OUTPOST only performs perturbation with iteration-based decay. Our experimental results demonstrate that OUTPOST can achieve a much better tradeoff than the state-of-the-art with respect to convergence performance, computational overhead, and protection against gradient attacks.

## I. INTRODUCTION

As an emerging distributed machine learning paradigm, federated learning (FL) allows clients to train machine learning models collaboratively with private data, without transmitting them to the server. Though federated learning is celebrated as a privacy-preserving paradigm of training machine learning models, it was pointed out in the recent literature [1] that sharing gradients with the server may lead to the potential reconstruction of raw private data, such as images and texts, used in the training process. The discovery of this new attack, known as *Deep Leakage from Gradients* (DLG), has stimulated a new line of research to improve the attack efficiency [2]–[4] and to provide stronger defenses against known DLG-family attacks [5], [6] as well.

As existing studies have made significant efforts to indicate that federated learning is vulnerable to gradient attacks from a malicious participant or eavesdropper, a lightweight defense

that provides adequate privacy protection with guaranteed training accuracy is sought by recent work to prevent this attack [6]. However, before designing for even more efficient and effective defense mechanisms, we begin to have second thoughts on how severe the threat is in practice, even without any defense mechanisms in place. Existing works focused on reconstructing raw data from known gradients or model weights in ideal settings, rather than considering practical settings in production federated learning.

As its name suggests, DLG proved that sharing gradients has the potential of leaking private data. However, when this attack was first proposed and later improved, most works in the literature considered sharing model updates as equivalent to sharing gradients. In production FL, however, multiple epochs are used routinely, and gradients are only accessible locally in a single step of gradient descent. No gradient — in its strict, original connotation — is transmitted to the server at all. Instead, only model updates — the *delta* between local models and the server’s global model in the preceding round — are transmitted from clients to the server. Yet, to the best of our knowledge, very little is known on the effectiveness of gradient attacks in practical contexts in production federated learning. It was shown [7] that gradients can be calculated from model updates with a known learning rate; but with multiple epochs, we find that this calculation is far from accurate. Only [4], [6] considered the possibility of reconstructing raw data with model updates directly, without estimating the gradients.

Even with the assumption of direct gradient sharing, existing works have mainly validated the efficiency when reconstructing one or multiple images (using a larger batch size) in full gradient descent, i.e., merely one local step of Stochastic Gradient Descent (SGD). None of them has shown convincing evidence that reconstructed images are recognizable by humans under the standard settings of production federated learning, where clients perform more local computation and less communication (i.e., multiple update steps on a local model) [8]. Moreover, existing attacks neglect the change of model status through FL training and tend to use untrained neural networks that are explicitly initialized with weights of a wide distribution for deriving gradients, which we have found makes the model and shared gradients fundamentally more vulnerable. Our empirical results disclose very limited privacy leakage even when gradients are shared, not to mention the

privacy leakage with model updates, *delta*, only.

Existing defenses in the literature were designed explicitly for gradient inversion attacks [5], [6], and evaluated their performance with respect to privacy metrics (such as the peak signal-to-noise ratio) and utility metrics (such as the validation accuracy of the global model). However, they failed to show the feasibility of data reconstruction by the DLG attack in the first place before any defense is even applied, especially in the context of production FL [9] with federated averaging (FedAvg). Given that gradient leakage attacks are much weaker as we will discover in this paper, we argue that a new defense mechanism against such attacks can be much more lightweight, without introducing extra overhead or incurring the risk of sacrificing the utility of the global model after training completes.

Inspired by our empirical observations that models with weights from a narrower distribution and more local SGD update steps will effectively make potential attacks weaker, we propose a new defense mechanism, called OUTPOST, that provides *sufficient* and *self-adaptive* protection throughout the federated learning process against time-varying levels of privacy leakage risks. As its highlight, OUTPOST is designed to apply *biased perturbation* to gradients based on how spread out and how informative model weights are at different local update steps. Specifically, OUTPOST uses a probability threshold to decide whether we perform a perturbation to the gradients at the current step or not; and to limit the computation overhead, such a threshold decays as local update steps progress over time. When performing the perturbation, OUTPOST first evaluates privacy leakage risks of the current local model by the variance statistics of the model weights at each layer, and then adds Gaussian noise to each layer of the gradients of the current step based on the Fisher information matrix, whose range is decided by the quantified privacy leakage risks.

We have evaluated OUTPOST and four state-of-the-art defense mechanisms in the literature, against two gradient leakage attacks under both production FL settings and a simplistic, yet unrealistic, FL scenario where the attack is the most efficient<sup>1</sup>. We seek to evaluate both *utility metrics* — regarding both the wall-clock time needed to converge and the converged accuracy — and *privacy metrics*, which shows the effectiveness of the defenses against gradient leakage attacks in the worst case. With two image classification datasets and the same LeNet neural network architecture used in the literature [1], our experimental results have demonstrated convincing evidence that OUTPOST can achieve a much better accuracy compared with the state-of-the-art, incurs a much smaller amount of computational overhead, while effectively providing a sufficient level of protection against DLG attacks when evaluated using common privacy metrics in the literature.

<sup>1</sup>Our implementation is available as an open-source git repository at <https://github.com/TL-System/plato/tree/main/examples/dlg>.

## II. PRELIMINARIES

Recent research has discovered that, by simply exchanging gradients of neural network models rather than raw data among multiple clients, privacy leakage still cannot be prevented in distributed machine learning. An honest-but-curious server can recover the private data from the obtained gradients through an optimization process [1], [2].

Essentially, DLG and other existing gradient inversion attacks tried to solve an optimization problem. With the given target gradient  $\nabla^* : \nabla_t^k = \frac{\partial \mathcal{L}_{\text{grad}}(F_w(\mathbf{x}^*), \mathbf{y}^*)}{\partial w}$  received from a participant  $k$  at a certain step  $t$ , the attacker can steal the inputs with labels  $(\mathbf{x}_t^k, \mathbf{y}_t^k)$  such as an image in pixels or a sentence in tokens in this training step. To do so, the attacker first generates random (dummy) inputs with labels  $(\mathbf{x}'_0, \mathbf{y}'_0)$  of the same size as the target data. The attacker then (1) derives dummy gradient  $\nabla'_0 = \frac{\partial \mathcal{L}_{\text{grad}}(F_w(\mathbf{x}'_0), \mathbf{y}'_0)}{\partial w}$  w.r.t. the model weights from the dummy data by feeding the dummy data into the machine learning model  $F_w$  shared between the participants; (2) updates the dummy data  $(\mathbf{x}'_1, \mathbf{y}'_1)$  by gradient-based methods, in which the loss function is the distance between the dummy gradient and the given gradient  $\nabla^*$ . By iterating these steps, the dummy data  $(\mathbf{x}'_i, \mathbf{y}'_i)$  will move closer to the target training data as the dummy gradient matches the given gradient based on the following objective:

$$\mathbf{x}'^*, \mathbf{y}'^* = \arg \min_{\mathbf{x}', \mathbf{y}'} \mathbb{D}(\nabla', \nabla^*). \quad (1)$$

The distance  $\mathbb{D}(\nabla', \nabla^*)$  is a function differentiable w.r.t. the dummy data, which can be the L2 distance  $\|\nabla' - \nabla^*\|^2$  [1] or the cosine distance  $1 - \frac{\langle \nabla', \nabla^* \rangle}{\|\nabla'\| \|\nabla^*\|}$  [2]. We illustrate the process that DLG attacks use in Fig. 1. More comprehensive objective functions incorporating regularization terms have been designed by some existing works to make the reconstructed images less noisy [2].

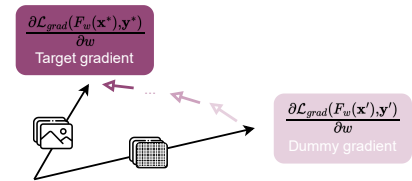


Fig. 1: Matching the dummy gradient with the target gradient.

Existing gradient inversion attacks, including [1]–[4], [7], [10], explored the space of this attack in a wider set of circumstances, such as reconstructing multiple images and allowing multiple training steps. However, none of them considered these settings in the context of production FL. We now proceed to systematically evaluate the effectiveness of these attacks, considering both the required assumptions and applicable coverage, in the context of production FL.

## III. RE-EVALUATING GRADIENT LEAKAGE ATTACKS IN PRODUCTION FEDERATED LEARNING

In this section, we first show that existing gradient inversion attacks do not work effectively in production FL, as some of their implicit assumptions fail to hold.



TABLE I: Reconstructing a single image using the DLG attack, with different model initialization methods or training stages.

Ground truth	Untrained network with explicit initialization			Untrained network with default PyTorch initialization			Trained network pre-trained with the same data		
	RS1	RS2	RS3	RS1	RS2	RS3	RS1	RS2	RS3

TABLE II: Assumptions and settings: production FL vs. what was used in existing gradient leakage attacks.

Parameters/Settings	What is practical in production FL	Used in previous attacks			
Network model state	Model in an arbitrary communication round	Untrained model with explicit initialization			
		$\xrightarrow{E:\text{stronger}}$	$\xrightarrow{n:\text{stronger}}$	$\xrightarrow{B:\text{stronger}}$	Examples
		$1 \geq 1$	$1 \geq 1$	$= n \leq n$	
The number of epochs $E$ , the number of data samples $n$ , and batch size $B$	Multiple local steps with $E \geq 1, n \geq 1, B \leq n$	✓	✓	✓	DLG [1]
		✓	✓	✓	iDLG [3]
		✓	✓	✓	csDLG [2]
		✓	✓	✓	GradInversion [4]
Data heterogeneity	non-i.i.d. distribution	Not fully investigated			
Gradient descent optimizer	Incorporating learning rate, momentum, weight decay, learning rate schedule	Fixed learning rate only			
Other prior knowledge	Not accessible for the server	Batch normalization statistics [2], [4], private labels [2], etc.			

labels as a result of such a non-i.i.d. distribution. Under these circumstances, the accuracy of label estimation may be significantly affected.

### B. More Sophisticated DLG Attacks

**Approximating gradients from updates.** As we have argued, instead of gradients, *model updates* are transmitted from clients to the server in production FL. When training a neural network in federated learning, each client tries to optimize the network parameters  $\theta$  using a loss function  $\mathcal{L}_\theta$  on local training data. With a model  $F_{w_t^k}$ , the gradient  $\nabla_t^k$  at a local training step can be evaluated by  $\nabla \mathcal{L}_\theta(F_{w_t^k})$  and the new model weights can be updated with a learning rate  $\eta$  as  $w_{t+1}^k = w_t^k - \eta \nabla_t^k$ . With  $\tau^k$  steps of local training completed, the model weights at the end of this communication round can be expressed as  $w^k(t+1) : w_{t+\tau^k}^k$ . After multiple local steps of training in a communication, the server updates the global model by the local model updates (delta) as  $w(t+1) = w(t) + \sum_k \frac{n_k}{n} \Delta^k(t, t+1)$ , where  $\Delta^k(t, t+1) = w^k(t+1) - w^k(t)$ .

Existing work in the literature [7], [12] considered gradients and model updates as mathematically equivalent, and used Eq. (2) to convert the delta  $\Delta^k(t, t+1)$  to gradients before performing gradient matching. Note that, for such an approximation to be accurate, the fixed learning rate used at the victim client (or shared between all the clients) must be known by the attacker.

$$\nabla^k(t, t+1) = -\frac{\Delta^k(t, t+1)}{\eta} \quad (2)$$

Even with a known learning rate, such an approximation only works effectively in the simplest case where only the learning rate is used in the local gradient descent steps. In production FL, however, there are a number of other factors, such as momentum, weight decay, and learning rate schedules, that are used routinely as best practices when training neural networks. Gradient approximation from model updates will fail in these more realistic settings as the computation is not fully reversible for the attacker [13].

**Matching deltas from updates.** As an alternative to approximating gradients from updates, Geiping *et al.* [2] directly conducted the matching process over model updates (or updated weights), with a similar matching mechanism to conventional gradient matching. The attacker directly matches its dummy weights or weight delta with the absolute weights or model updates.

For delta matching to be effective, the server requires a series of prior knowledge to realize the same gradient descent process using the dummy data instead, including the number of images to be reconstructed  $n$ , the batch size  $B$ , the learning rate  $\eta$ , and other gradient descent factors such as weight decay and momentum. What's more, the effectiveness of the delta matching process relies heavily on the assumption that attackers know the data labels [2]. Without known labels as prior, data reconstruction will become harder. But as we have pointed out earlier in this section, label recovery often fails in production FL, where client data distributions are usually non-i.i.d.

**Summary.** Based on our analysis so far, the most feasible



way to perform this kind of attack in federated learning is by matching deltas from updates. Therefore, when we devise our new defense and conduct experiments, we only consider matching deltas from updates, which does not imply that our defense is ineffective against attacks that approximate gradients from updates. We show in Table II the stark differences between assumptions in the gradient leakage literature and practical settings in production FL.

#### IV. OUTPOST: OUR LIGHTWEIGHT DEFENSE

Thanks to the inherent resilience against gradient leakage attacks in production FL, such as multiple local iterations and more complex gradient descent optimizers, it becomes feasible to design a simple, lightweight, yet effective gradient protection mechanism as a proactive defense, without sacrificing the training performance in FL sessions. In this section, we propose a new defense mechanism, called OUTPOST, to achieve these objectives.

##### A. OUTPOST: Mechanism Design

**Privacy leakage risk.** The scale of the initial distribution has a significant effect on both the outcome of the optimization procedure and on the ability of the neural network model to generalize [14]. To elaborate what we have found in Section III, the default model initialization in PyTorch uses the Kaiming Uniform method [15] for both linear and convolutional layers. The weights (and biases) of each layer are values drawn randomly from a uniform distribution of  $\mathcal{U}(-\sqrt{\frac{1}{\text{in\_features}}}, \sqrt{\frac{1}{\text{in\_features}}})$ , where `in_features` is the size of the previous layer. However, the explicit model weight initialization in those existing attacks for generating the gradients uses a distribution of  $\mathcal{U}(-0.5, 0.5)$ , which is of a much larger scale. This phenomenon gives us insights that the magnitude of neural network weights can intuitively reveal the privacy leakage risks of the corresponding gradients. Therefore, we employ the variance statistics of the neural network weights as a way to quantify the privacy leakage risks per layer. We denote the scalar variance of all weights at layer  $d$  as  $\text{Var}[w_d]$ . Then, the privacy leakage risk at the  $d$ -th layer of the client  $k$ 's local model at communication round  $t$  iteration  $i$  can be expressed as  $r_{t+i_d}^k = \text{Var}[w_{t+i_d}^k]$ .

**Selective perturbation.** Existing defenses based on gradient compression or pruning techniques are designed for pruning gradients with small magnitudes to zeros. As these gradients have insignificant effects when weights are updated, the accuracy and convergence speed are both preserved with these techniques in place. However, we doubt the effectiveness of this method in the context of gradient leakage attacks, as the remaining gradients still carry the primary information for data reconstruction. With this insight, we suspect that perturbing those insignificant gradients cannot sufficiently protect privacy information from recovering. Therefore, we tend to also perturb gradients of which the model parameters contain more important information.

We choose the diagonal of the Fisher information matrix (FIM) to estimate how important a certain parameter is.

FIM [16] is strongly related to the Hessian Matrix, indicating the curvature of the loss function for efficient optimization. The Fisher information of the model can be expressed as:

$$I_\theta = E_{p(x|\theta)} [\nabla_\theta \log p(x|\theta) \nabla_\theta \log p(x|\theta)^T], \quad (3)$$

where  $\log p(x|\theta)$  is a the log-likelihood function given by the model with parameter  $\theta$ .

However, it is infeasible to directly use FIM or Hessian information in the context of deep learning as the likelihood is intractable. We use the empirical Fisher information matrix instead, which crudely approximates the FIM, and is often used to make computation easier [17]. We express the empirical Fisher as

$$\hat{\mathbf{F}} = \frac{1}{n/B} \sum_{i=1}^{n/B} \nabla \mathcal{L}_{\theta_i} \cdot \nabla \mathcal{L}_{\theta_i}^\top, \quad (4)$$

where  $\nabla \mathcal{L}_{\theta_i}$  denotes the gradient w.r.t. the  $i$ -th samples at the given point, and  $\cdot$  is the outer product of individual gradients.

In the given iteration, we can compute the empirical Fisher for each model parameter based on average gradients over this batch of data. When performing the perturbation, we only choose gradients with the  $\varphi\%$  highest values of empirical Fisher in each layer to add noise, at a level determined by the privacy leakage risk.

**Perturbation with iteration-based decay.** As we have found that model updates shared after multiple local training steps have a decreased risk of leaking information about the raw training data, we propose to devise a schedule in OUTPOST to adaptively perturb gradients in different local training steps. Unlike existing defenses that treat gradients in each local step equally, we introduce a bias on the probability of perturbing gradients according to the number of local iterations. Specifically, in each local training iteration  $i$ , the probability of a client applying perturbation on its gradients averaged over a mini-batch at the current iteration is determined by  $\text{Pr} = 1/(1 + \beta \cdot i)$ , where  $\beta$  is the hyperparameter to control how fast the probability decay is as the number of iterations grows.

The layer-based perturbation consists of two steps: (1) compressing gradients by their magnitudes:  $\rho\%$  of the smallest gradients in each layer  $l$  are pruned to zeros; (2) adding noise to gradients by their privacy leakage risks: noise is added to each layer  $l$  of the gradients following the Gaussian distribution  $\mathcal{N}(0, (\lambda r_{t+i_d}^k)^2)$ , where  $\lambda$  controls the range of variance. With such a design, OUTPOST is a simple and *self-adaptive* defense mechanism against time-varying levels of privacy leakage risks, yet without introducing a substantial amount of computation overhead. The overall random perturbation mechanism in OUTPOST is shown as Algorithm 1.

##### B. Convergence Guarantee with the FedAvg Algorithm

Our convergence analysis of FedAvg on non-i.i.d. data with the OUTPOST defense mechanism is based on the following assumptions. We use the same assumptions (Assumptions 1 to 5) as [8] on local objective functions  $F_1, \dots, F_N$  and partial device participation.

**Algorithm 1** FedAvg local training at client  $k$  with OUTPOST

**Input:** Broadcast the global model  $F_{wt}$  with weights  $w(t)$  of the current communication round  $t$ ; learning rate  $\eta$ ; the number of local data samples  $n$ ; the number of epochs  $E$ ; batch size  $B$ ; Initial iteration number as 0

**Output:** Local model update  $\Delta^k(t, t+1)$

- 1: Set local model weights same as the global one  $w^k(t) = w(t)$
- 2: **for** each epoch 1 to  $E$  **do**
- 3:   **for** each batch 1 to  $n/B$  **do**
- 4:     Iteration  $i = i + 1$
- 5:     Compute loss and derive gradient at the current iteration  $\nabla_{t+i}^k = \nabla \mathcal{L}_\theta(F_{w_{t+i}^k})$
- 6:     **if** a random value  $(\in [0, 1)) \leq \text{Pr} = 1/(1 + \beta \cdot i)$  or  $i = 1$  **then**
- 7:       Evaluate privacy leakage risks based on current model per layer by  $r_{t+i,d}^k = \text{Var}[w_{t+i,d}^k]$
- 8:       Perform pruning with threshold  $\rho\%$  and update perturbed gradient to  $\tilde{\nabla}_{t+i}^k$
- 9:       Add noise to the selected  $\varphi\%$  gradient at each layer and update the perturbed gradient as  $\tilde{\nabla}_{t+i}^k = \tilde{\nabla}_{t+i}^k + m$  where  $m \in \mathcal{N}(0, (\lambda r_{t+i,d}^k)^2)$
- 10:     **else**
- 11:       Update perturbed gradient as  $\tilde{\nabla}_{t+i}^k = \nabla_{t+i}^k$
- 12:     **end if**
- 13:     Update local model weights with the perturbed gradient and learning rate as  $w^{t+i} = w^{t+i-1} - \eta \tilde{\nabla}_{t+i}^k$
- 14:   **end for**
- 15: **end for**
- 16: Send the model update  $\Delta^k(t, t+1) = w^k(t+1) - w^k(t)$  to the server

**Assumption 1.**  $F_1, \dots, F_N$  are all  $L$ -smooth: for all  $\mathbf{v}$  and  $\mathbf{w}$ ,  $F_k(\mathbf{v}) \leq F_k(\mathbf{w}) + (\mathbf{v} - \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{L}{2} \|\mathbf{v} - \mathbf{w}\|_2^2$ .

**Assumption 2.**  $F_1, \dots, F_N$  are all  $\mu$ -strongly convex: for all  $\mathbf{v}$  and  $\mathbf{w}$ ,  $F_k(\mathbf{v}) \geq F_k(\mathbf{w}) + (\mathbf{v} - \mathbf{w})^T \nabla F_k(\mathbf{w}) + \frac{\mu}{2} \|\mathbf{v} - \mathbf{w}\|_2^2$

**Assumption 3.** Let  $\xi_t^k$  be sampled from the  $k$ -th device's local data uniformly at random. The variance of stochastic gradients in each device is bounded:  $\mathbb{E} \|\nabla F_k(\mathbf{w}_t^k, \xi_t^k) - \nabla F_k(\mathbf{w}_t^k)\|^2 \leq \sigma_k^2$  for  $k = 1, \dots, N$ .

**Assumption 4.** The expected squared norm of stochastic gradients is uniformly bounded, i.e.,  $\mathbb{E} \|\nabla F_k(\mathbf{w}_t^k, \xi_t^k)\|^2 \leq G^2$  for all  $k = 1, \dots, N$  and  $t = 1, \dots, T-1$

**Assumption 5.** Assume  $\mathcal{S}_t$  contains a subset of  $K$  indices uniformly sampled from  $[N]$  without replacement. Assume the data is balanced in the sense that  $p_1 = \dots = p_N = \frac{1}{N}$ . The aggregation step of FedAvg performs  $\mathbf{w}_t \leftarrow \frac{N}{K} \sum_{k \in \mathcal{S}_t} p_k \mathbf{w}_t^k$ .

We now derive new bounds for Assumption 3 with our defense. The expected norm of the distance between the perturbed gradients  $\nabla F'_k(\mathbf{W}_t^k, \xi_t^k)$  and the original gradients  $\nabla F_k(\mathbf{W}_t^k, \xi_t^k)$  of the whole model is the sum of that of every

layer. Thus,

$$\begin{aligned} & \mathbb{E} \left\| \nabla F'_k(\mathbf{W}_t^k, \xi_t^k) - \nabla F_k(\mathbf{W}_t^k, \xi_t^k) \right\|^2 \\ &= \sum_{d=1}^D \mathbb{E} \left\| \nabla F'_k(\mathbf{w}_{dt}^k, \xi_t^k) - \nabla F_k(\mathbf{w}_{dt}^k, \xi_t^k) \right\|^2 \quad (5) \\ &\leq (\lambda r_d^k)^2 \cdot D, \end{aligned}$$

according to our perturbation noise distribution  $\mathcal{N}(0, (\lambda r_d^k)^2)$ , where  $r_d^k \in [0, 1]$  is the privacy risk of the  $d$ -th layer, and  $D$  is the total number of layers of the stochastic gradients.

By using the norm triangle inequality, we can bound the variance of stochastic gradients after perturbation in each device as

$$\begin{aligned} & \mathbb{E} \left\| \nabla F'_k(\mathbf{W}_t^k, \xi_t^k) - \nabla F_k(\mathbf{W}_t^k) \right\|^2 \\ &\leq \mathbb{E} \left\| \nabla F'_k(\mathbf{W}_t^k, \xi_t^k) - \nabla F_k(\mathbf{W}_t^k, \xi_t^k) \right\|^2 \quad (6) \\ &\quad + \mathbb{E} \left\| \nabla F_k(\mathbf{W}_t^k, \xi_t^k) - \nabla F_k(\mathbf{W}_t^k) \right\|^2 \\ &\leq (\lambda r_d^k)^2 \cdot D + \sigma_k^2, \end{aligned}$$

in which we use Assumption 3 and Eq. (5).

Similarly, we can derive new bounds for Assumption 4 of the expected squared norm of stochastic gradients after perturbation in each device as  $\mathbb{E} \|\nabla F'_k(\mathbf{w}_t^k, \xi_t^k)\|^2 \leq (\lambda r_d^k)^2 \cdot D + G^2$  for all  $k = 1, \dots, N$  and  $t = 1, \dots, T-1$ .

Similar to [8], let  $F^*$  and  $F_k^*$  be the minimum values of  $F$  and  $F_k$ , respectively. The term  $\Gamma = F^* - \sum_{k=1}^N p_k F_k^* > 0$  can be used for quantifying the degree of non-i.i.d. We then have the following convergence guarantee with FedAvg on non-i.i.d. data, and with the OUTPOST defense mechanism.

**Theorem 1.** Let Assumptions 1 to 5 hold and  $L, \mu, \sigma_k, G$  be defined therein. Choose  $\kappa = \frac{L}{\mu}$ ,  $\gamma = \max\{8\kappa, E\}$  and the learning rate  $\eta_t = \frac{2}{\mu(\gamma+t)}$ . Then

$$\begin{aligned} & \mathbb{E}[F(\mathbf{w}_T)] - F^* \leq \\ & \frac{\kappa}{\gamma + T - 1} \left( \frac{2(B+C)}{\mu} + \frac{\mu\gamma}{2} \mathbb{E} \|\mathbf{w}_1 - \mathbf{w}^*\|^2 \right), \end{aligned}$$

where

$$\begin{aligned} B &= \sum_{k=1}^N p_k^2 (\sigma_k^2 + (\lambda r_d^k)^2 \cdot D) + 6L\Gamma + 8(E-1)^2 ((\lambda r_d^k)^2 \cdot D + G^2), \\ C &= \frac{N-K}{N-1} \frac{4}{K} E^2 ((\lambda r_d^k)^2 \cdot D + G^2). \end{aligned}$$

## V. EXPERIMENTAL RESULTS

We are now ready to compare OUTPOST with state-of-the-art defense mechanisms in the literature, against different gradient leakage attacks and under a variety of experimental settings. All our experiments are conducted on a server with two Intel Xeon Silver 4210R CPUs and one NVIDIA RTX A4500 GPU with 20 GB CUDA memory.

**Defense and attack baselines.** We compare OUTPOST with two state-of-the-art defense mechanisms: Soteria [5]

and GradDefense (GD) [6], along with two commonly used defenses for general attacks in federated learning — gradient compression (GC) [1], which prunes small values in gradients; and differential privacy (DP) [18], which adds noise to gradients. We evaluate these defense mechanisms against two gradient leakage attacks: DLG [1] and csDLG [2]. When applying these attacks, we use the mechanism of matching deltas from updates, which is the most practical choice in production FL.

**Datasets and models.** We evaluate both gradient leakage attacks and defenses in PLATO,<sup>2</sup> an open-source research framework for federated learning. We show data reconstruction results over two image classification datasets: EMNIST and CIFAR-10. We use the same LeNet model evaluated in [1], [2], which consists of 4 convolutional layers and 1 fully-connected layer.

**Evaluation metrics.** To evaluate the effectiveness of the defense mechanisms, we use mean-square-error (MSE), structural similarity index measure (SSIM), and learned perceptual image patch similarity (LPIPS) as our metrics to measure the distance between the reconstructed image and raw image. To evaluate the impact of the defenses on the convergence performance of FL training, we show the accuracy of the global model on the validation dataset over multiple communication rounds using the FedAvg algorithm. To evaluate the computation overhead introduced by the defenses, we measure the wall-clock time averaged across multiple communication rounds.

**Hyperparameter configurations.** We evaluate OUTPOST with respect to two aspects: (1) *training performance*, where we examine how our defense affects the FL training time and the validation accuracy of the converged model in production FL settings; and (2) *defense effectiveness*, where we evaluate how effective OUTPOST is under settings where the gradient leakage attacks are as threatening as possible. For alternative defense mechanisms, we have the following configurations. For GC, we set the pruning rate of gradients to 80%; for DP, we use Laplacian noise and set the noise variance to 0.1; for Soteria, we set the pruning rate of the fully connected layer’s gradients to 50%; for GradDefense, we turn on the local clipping operation and use the same settings in their source code with 0.01 as the Gaussian noise variance; and for OUTPOST, we set our hyperparameters as  $\lambda = 0.8, \varphi = 40, \beta = 0.1, \rho = 80$ .

#### A. Evaluating the Training Performance

With respect to the training performance, for both datasets, data is non-i.i.d. distributed across 100 clients, each holding 1% of the total training samples (i.e., 1128 for EMNIST and 500 for CIFAR-10). Regarding the local epoch  $E$  and batch size  $B$ , we set them as  $E = 5, B = 32$  for both datasets. We apply the SGD optimizer for local training and set the learning rate  $\eta$  to 0.01. In each communication round, the server randomly selects 10 clients out of 100 and aggregates

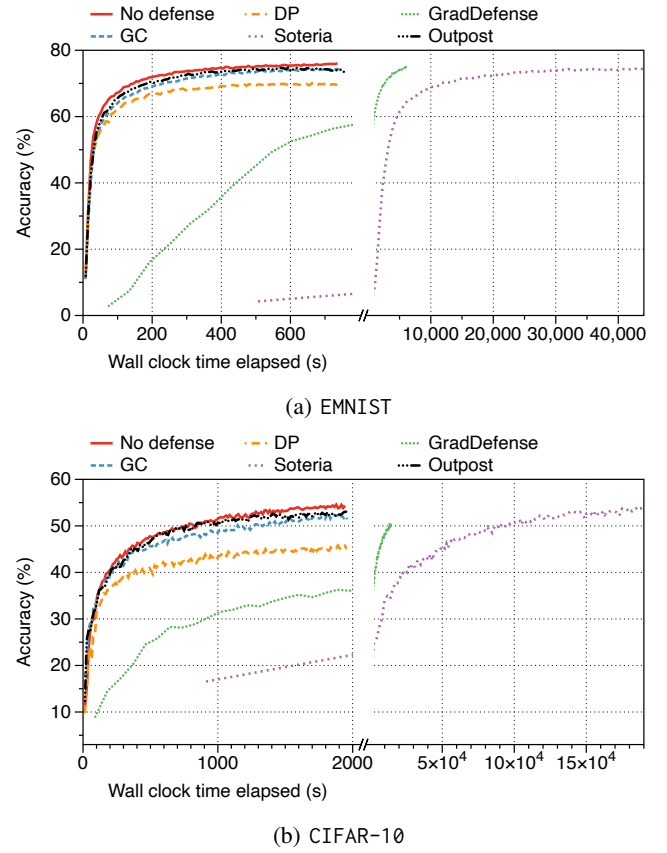


Fig. 3: The impact of defenses on FL training.


















































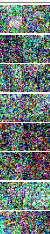
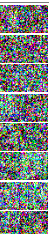
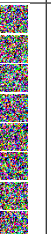
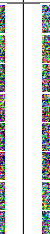
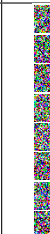


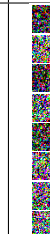
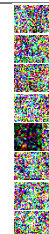


their model updates. We terminate FedAvg training sessions with various defenses when the number of communication rounds reaches 100 on EMNIST and 150 CIFAR-10, where the global models converge.

With respect to the validation accuracy of the converged model and the elapsed wall-clock time, our results over EMNIST and CIFAR-10 have been shown in Fig. 3. If we examine the ultimate global model accuracy at the end of each training session, it can be observed that all the defenses incurred a certain amount of losses. DP affected the convergence performance the most, and only reached 91.5% and 83.1% of the validation accuracy without any defense, over EMNIST and CIFAR-10 datasets, respectively. This observation is consistent with our expectations based on these defense mechanisms. GC and Soteria both prune gradients with small magnitudes to zeros, which have insignificant effects on weights to be updated. Though GradDefense and OUTPOST also add noise to gradients, the level of noise is controlled with model sensitivity and model status, respectively. Overall, OUTPOST induces only 3.28% and 2.19% accuracy loss with 3.54% and 1.47% delay, over EMNIST and CIFAR-10 datasets, respectively.

When we examine the wall-clock time elapsed in the same number of communication rounds, it is apparent from Fig. 3 that GC, DP and OUTPOST as defenses did not introduce any significant delays due to the computation overhead, while

<sup>2</sup>Available online at <https://github.com/TL-System/plato>.

TABLE III: The effectiveness of various defense mechanisms against different attack baselines, datasets, hyperparameter configurations.

	DLG						csDLG					
	[Scenario 1] EMNIST: $E = 1, n = 1, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE $\uparrow$	6.6e-3	13.96	113.63	95.19	32.57	77.05	2.6e-7	199.08	297.84	296.76	360.98	294.678
LPIPS $\uparrow$	7.1e-2	0.55	0.60	0.63	0.64	0.58	5.8e-7	0.60	0.66	0.63	0.64	0.68
SSIM $\downarrow$	0.99	0.30	0.19	3.3e-2	1.0e-2	0.13	1.00	0.32	6.5e-2	4.1e-2	1.7e-2	1.6e-2
												
	[Scenario 2] EMNIST: $E = 1, n = 2, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE $\uparrow$	5.8e-2	69.87	319.30	16.07	25.50	75.92	0.14	336.05	1231.82	782.93	668.79	277.73
LPIPS $\uparrow$	0.31	0.61	0.66	0.70	0.65	0.67	0.40	0.60	0.65	0.59	0.63	0.66
SSIM $\downarrow$	0.77	3.7e-2	2.4e-2	4.7e-2	2.7e-2	6.4e-2	0.70	3.3e-2	1.4e-2	1.6e-2	3.4e-2	3.8e-2
 	 	 	 	 	 	 	 	 	 	 	 	 
	[Scenario 3] CIFAR-10: $E = 1, n = 1, B = 1$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE $\uparrow$	5.1e-2	7.83	34.08	25.91	11.46	13.10	5.9e-5	27.50	34.51	25.91	56.66	35.24
LPIPS $\uparrow$	0.53	0.77	0.77	0.76	0.74	0.77	1.8e-3	0.76	0.78	0.76	0.77	0.77
SSIM $\downarrow$	0.57	4.4e-2	3.6e-2	5.5e-2	2.2e-2	2.1e-2	0.99	2.6e-2	2.9e-2	5.5e-2	1.7e-2	3.4e-2
												
	[Scenario 4] CIFAR-10: $E = 1, n = 16, B = 16$											
	No defense	GC	DP	Soteria	GD	OUTPOST	No defense	GC	DP	Soteria	GD	OUTPOST
MSE $\uparrow$	1.37	4.82	19.92	2.85	4.04	15.85	6.2e-2	6.85	13.68	14.87	2.1e10	13.03
LPIPS $\uparrow$	0.67	0.69	0.71	0.70	0.69	0.70	0.51	0.69	0.71	0.69	0.71	0.70
SSIM $\downarrow$	2.9e-2	1.9e-2	1.5e-2	2.2e-2	2.2e-2	1.7e-2	0.30	4.5e-2	1.6e-2	2.7e-2	3.8e-2	3.1e-2
												

GradDefense and Soteria extended the training session an order of magnitude longer than FedAvg without any defenses. This is because Soteria has to learn the perturbed data representation in each local SGD iteration based on data representation in the FC layer of the model trained after that iteration. Similarly, GradDefense needs to compute model sensitivity in each iteration based on which it adds Gaussian noise to selected slices of gradients. In contrast, OUTPOST is an order of magnitude less time-consuming than these state-of-the-art alternatives, thanks to its fast evaluation of the privacy leakage risk based on model status and its perturbation with iteration-based decay.

One may wonder why Soteria and GradDefense are not applied to model updates at the end of each communication round instead, since the notions of gradients and model updates are not clearly differentiated in their papers. This is due to the fact that, to compute the mask for pruning, Soteria

needs detailed information such as the gradient of the loss with respect to the input batch data, which is only accessible in each SGD iteration. The slicing and perturbation methods in GradDefense are also gradient specific.

#### B. How Effective are the Defenses?

For our defense evaluation, we make the attack as strong as possible by having only one client participate in the training, with the attack performed in the first round of training. The model is explicitly initialized with a wider distribution, and no momentum, weight decay, or learning rate scheduler is applied to the SGD optimizer. Although we've shown in Section III that this setting is completely unrealistic, we only use it to demonstrate the effectiveness of the defenses, and exaggerate the difference between various defense mechanisms in this extreme case. For both DLG and csDLG attacks, we applied the L-BFGS optimizer with 3000 iterations of reconstruction to



guarantee convergence, and presented the worst defense result in 10 trials with different dummy data.

Table III shows the results from our experiments, evaluating a variety of metrics for each defense. The up and down arrows indicate the direction of better defense performance. We also put the reconstructed images along with ground truth images in the last row of each experiment scenario to show if they are recognizable by human eyes. Note that the order of reconstructed images may be inconsistent with the raw ones in Scenario 4, which is an inherent issue of the attacks.

Our previous argument — that gradient leakage attacks are not practical when there are multiple update steps in local training — can be supported by our results in the columns of attacks without defense in Table III, even though csDLG demonstrates a better data reconstruction capability in general. As shown in Scenario 2, the performance of attacks is heavily affected with merely two update steps each having a batch of one data sample as the color difference in reconstructed images is less clear compared to Scenario 1. Performing the attack on a larger batch size not only makes the attacks significantly slower, but also worse, with only one image out of 16 being reconstructed using csDLG without defense. Realistic FL settings use batch sizes that are many magnitudes greater than 16, meaning that attacks are unlikely to ever converge; and even if they were to converge, a successful reconstruction would be nearly impossible.

We notice that MSE is the most inconsistent metric and a higher MSE does not correlate to a reconstructed image containing fewer key features of ground truth. Most related works use this metric as a basis for their conclusions, which means we may need to revisit some of their claims. Compared to CIFAR-10 images with  $32 \times 32$  pixels and three channels, images in EMNIST have  $28 \times 28$  pixels in resolution with only one channel, which makes it easier, not only to leak privacy information in gradients but also to recognize visually from the reconstructed images, even with defenses such as GC. Our OUTPOST does not achieve the highest LPIPS and lowest SSIM in every scenario with both attack methods; however, it can still provide solid and sufficient protection resulting in highly noisy images.

## VI. RELATED WORK

Starting from the pioneering work in [1], researchers have made efforts to improve the capability and efficiency of gradient leakage attacks. One of the highlights of such efforts, iDLG [3], further discovered that ground-truth labels can be directly extracted from the given gradients, which simplified the gradient matching process since it only needed to recover the inputs  $\mathbf{x}'$  in Eq. (1). Different from using the Euclidean distance as the objective function in DLG, Geiping *et al.* [2] utilized cosine similarity between the target gradient and the dummy gradient to optimize the reconstructed data during local updates. GradInversion [4] demonstrated even higher fidelity and better localization as compared with [1], [2]. GIAS [10] revealed more severe privacy leakage of raw data from gradients with prior knowledge about the pre-trained

generative model. These variants have made up the landscape of gradient leakage attacks to date.

To preserve the privacy from the gradients, researchers have applied several categories of general-purpose defenses to the gradient leakage attack, such as gradient compression and local differential privacy. But as [6] argued, these mechanisms were not custom tailored to this specific attack, and may incur either unacceptable computational overhead or significant degradation of performance, with respect to the converged accuracy after the FL training process completes. A particular defense, Soteria [5], was proposed specifically for the gradient leakage attack. In order to reduce the quality of the reconstructed data, and based on their finding that privacy leakage mainly comes from data representations embedded in the fully connected (FC) layer, it proposed a delicate design of perturbation on the data representation in the FC layer of shared gradients. Yet, GradDefense [6] showed that the raw data can still be recovered from the remaining gradients by muting the perturbed layer, and thus proposed a stronger defense by perturbing all the layers of the shared gradients by their measured sensitivity. Setting aside its statements contradictory to [5], GradDefense [6] demanded a substantial amount of computation for sensitivity measurements and perturbation by all the layers. Based on our own empirical observations and evaluations of the threat of gradient leakage attacks in the context of production FL, we are motivated to devise OUTPOST, a defense mechanism that is simple but good enough to defend against the small attack surface in production FL. OUTPOST is designed to optimize the trade-off between computation overhead, accuracy guarantee, and privacy preservation.

## VII. CONCLUDING REMARKS

In this paper, we have thoroughly investigated gradient leakage attacks, a popular category of attacks in federated learning. Our original objective was to conduct an in-depth study of these attacks in the context of production federated learning systems, and to design a practical, simple, and lightweight defense mechanism that can be used to defend against real-world threats. Along our journey to achieve this goal, we discovered that the effectiveness and efficiency of existing gradient leakage attacks are weakened by a substantial margin in standard federated learning settings, where clients send model updates rather than gradients, perform multiple local training iterations over local data with a non-i.i.d. distribution, and initialize model weights normally. With weakened attacks, we proposed OUTPOST, a new defense mechanism that can provide sufficient protection on shared model updates without sacrificing accuracy and convergence speed, and can adapt to time-varying levels of the privacy leakage risk throughout the federated learning process. We showed convincing results from a wide array of experiments that OUTPOST incurs much less computational overhead, achieves better accuracy, and converges much faster than its state-of-the-art alternatives in the literature. The authors have provided public access to their code at [OUTPOST](#).

## REFERENCES

- [1] L. Zhu, Z. Liu, and S. Han, “Deep Leakage from Gradients,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, “Inverting Gradients — How Easy Is It to Break Privacy in Federated Learning?” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 16 937–16 947, 2020.
- [3] B. Zhao, K. R. Mopuri, and H. Bilen, “iDLG: Improved Deep Leakage from Gradients,” *arXiv preprint arXiv:2001.02610*, 2020.
- [4] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, “See through Gradients: Image Batch Recovery via GradInversion,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 337–16 346.
- [5] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, “Soteria: Provable Defense against Privacy Leakage in Federated Learning from Representation Perspective,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 9311–9319.
- [6] J. Wang, S. Guo, X. Xie, and H. Qi, “Protect Privacy from Gradient Leakage Attack in Federated Learning,” in *Proc. IEEE INFOCOM*, 2022.
- [7] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, “A Framework for Evaluating Client Privacy Leakages in Federated Learning,” in *Proc. European Symposium on Research in Computer Security*. Springer, 2020.
- [8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the Convergence of FedAvg on Non-IID Data,” in *Proc. International Conference on Learning Representations (ICLR)*, 2020.
- [9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.
- [10] J. Jeon, K. Lee, S. Oh, J. Ok *et al.*, “Gradient Inversion with Generative Image Prior,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 29 898–29 908, 2021.
- [11] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, “Evaluating Gradient Inversion Attacks and Defenses in Federated Learning,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.
- [12] H. Wu and P. Wang, “Fast-Convergent Federated Learning with Adaptive Weighting,” *IEEE Trans. on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.
- [13] D. Maclaurin, D. Duvenaud, and R. Adams, “Gradient-Based Hyperparameter Optimization through Reversible Learning,” in *Proc. International Conference on Machine Learning (ICML)*, 2015, pp. 2113–2122.
- [14] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, <http://www.deeplearningbook.org>.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.
- [16] S.-I. Amari, “Natural Gradient Works Efficiently in Learning,” *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.
- [17] J. Martens, “Deep Learning via Hessian-Free Optimization,” in *Proc. International Conference on Machine Learning (ICML)*, vol. 27, 2010, pp. 735–742.
- [18] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning Differentially Private Recurrent Language Models,” in *Proc. International Conference on Learning Representations (ICLR)*, 2018.