



Deep Leakage from Gradients

Ligeng Zhu^(✉) and Song Han

Massachusetts Institute of Technology, Cambridge, USA
{ligeng,songhan}@mit.edu

Abstract. Exchanging model updates is a widely used method in the modern federated learning system. For a long time, people believed that gradients are safe to share: *i.e.*, the gradients are less informative than the training data. However, there is information hidden in the gradients. Moreover, it is even possible to reconstruct the private training data from the publicly shared gradients. This chapter discusses techniques that reveal information hidden in gradients and validate the effectiveness on common deep learning tasks. It is important to raise people’s awareness to rethink the gradient’s safety. Several possible defense strategies have also been discussed to prevent such privacy leakage.

Keywords: Federated Learning · Privacy leakage · Gradients’ safety

1 Introduction

Federated Learning (FL), has gained increasing attention as both data requirements and privacy concerns continue to rise [1–3]. In the Federated Learning system, the user data is not shared across the network and only model updates/gradients are transmitted. Therefore, such kind of distributed learning has been used in real-world applications where user privacy is crucial, e.g., hospital data [16] and text predictions on mobile devices [3]. Ideally, any such approach is considered as safe, as the gradients are thought less informative than the original data. Shown in Fig. 1 below, it is hard to infer from a list of numerical tensor values that the original image is a cat.



[1, 0, 0 ... 0]
cat

```
[[ 0.75,  1.26,  0.56,  ..., -0.19],  
 [-0.99, -0.37, -0.93,  ...,  2.54],  
 [ 0.06, -0.96,  0.78,  ..., -0.85],  
 ...,  
 [-0.55, -0.55, -1.31,  ...,  0.32]]
```

(a) Training data and label

(b) Corresponding gradients

Fig. 1. It is easy to compute gradients from the training, but not intuitive to perform vice versa. As shown in the Figure, human cannot read the cat (either image or label) from raw numerical values.

FL provides default participant privacy because only the gradients are shared across while the sensitive training never leaves the local device. However, is the protocol really safe? Do the gradients contain zero information about training data? If not, then how much information can we recover from them? In this section, we will explore the hidden information in the gradients and rethink the safety of gradients sharing scheme.

2 Information Leakage from Gradients

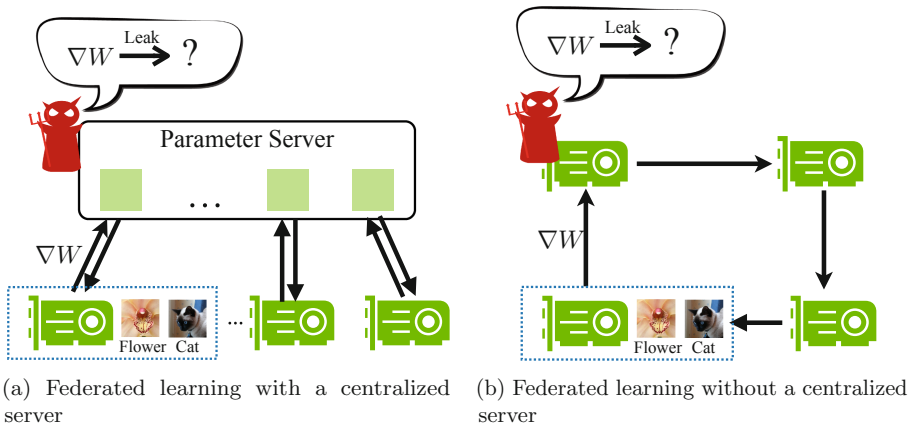


Fig. 2. The information leakage in two types of federated learning. The little red demon appears in the location where the leakage might happen.

From a privacy perspective, we are interested in possible leaks against an *honest-but-curious* server: It faithfully aggregates the updates from participants and delivers the updated model back, but it may be curious about the participant information and attempt to reveal it from the received updates. To study the question, we consider a key question: **What can be inferred about a participant’s training dataset from the gradients shared during federated learning?**

Given that gradients and model updates are mathematically equivalent, the local model updates can easily be obtained with the gradients and the learning rate. In the following discussion, gradient-based aggregation is used without loss of generality. We care what kind of information can be inferred from the gradients. Such an attack happens in the parameter server for centralized federated learning (Fig. 2a), or any neighbours in decentralized federated learning (Fig. 2b). We focus on centralized federated learning because it is more widely used [4, 5]. In this setting, several studies have been made to show that it is actually possible to infer some hidden information from the gradients.

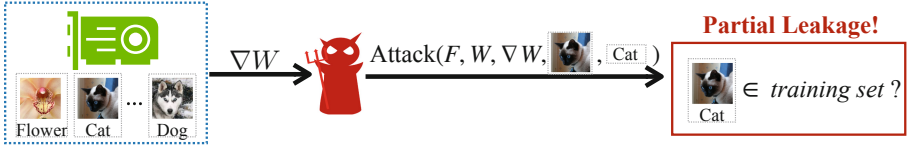


Fig. 3. Membership inference [6] in the federated setting. It uses the predicted results and ground truth labels to infer whether the record was *in* or *out* of the victim training datasets.

Membership-Inference. [6] is the basic privacy violation in this setting: Given an exact data point and pre-trained model, determine whether the data point was used to train the model. In Federated Learning, the gradients are sent to the server every round and the server thus knows the trained model based on local data. With membership-inference, the server is able to infer whether a specific data point exists in the local training set or not. In some cases, it can directly lead to privacy breaches. For example, finding that a specific patient’s clinical record was used to train a model associated with a disease can reveal the fact that the patient has the disease. In practice, Melis *et al.* [7] shows that a malicious attacker can convincingly (precision 0.99) tell whether a specific location profile was used to train a gender classifier on the FourSquare location dataset [8] (Fig. 3).

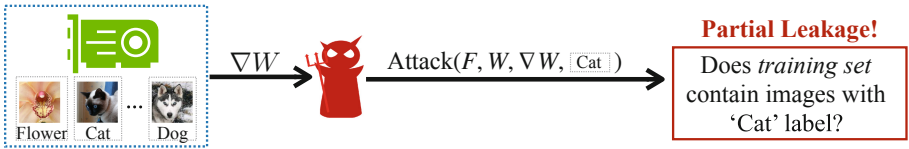


Fig. 4. Property inference [7] in the federated setting. It infers whether the victim’s training set contains a data point with certain property.

Property-Inference. [7] is a similar attack: Given a pre-trained model, determine whether the corresponding training set contains a data point with certain properties. It is worth noting that the property is not necessarily related to the main task. When a model is trained to recognize gender or race on the LFW dataset [9], the property-inference can not only reveal the people’s race and gender in the training set, but also tell they wear glasses or not. In practice, this also brings the potential risk of privacy leakage. It is easy to identify the patient if knowing his/her age, gender, race and glass-wearing, even the name and clinical record remain hidden.

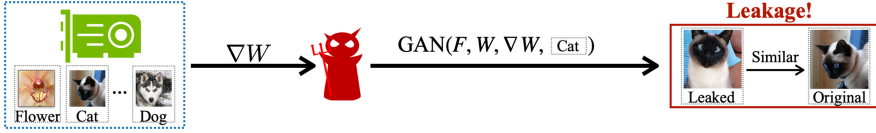


Fig. 5. Model inversion [10] in the federated setting. It first trains a GAN model from model updates and attacker’s own training data. Then it uses the GAN model to generate look alike images from the victim’s updates.

Model Inversion. [10] is another powerful attack that leaks the participant privacy. The attack exploits the real-time nature of the learning process and allows the adversary to train a Generative Adversarial Network (GAN) [11] to generate a prototypical sample of the targeted training set, which was meant to private. As shown in Fig. 5, the leaked images are almost identical as the original one, because the samples generated by the GAN are intended to come from the same distributions as the training data. This attack is powerful especially when all class members look alike (e.g., face recognition).

The three attack strategies above reveal a certain level of information hidden in the gradients, but they all have their own limitations. Membership inference requires an existing data record to perform the attack. This may be hard to get when the input data is not text (e.g., image, voice). Property inference relaxes the constraints and only require a label to execute the leakage. However, the attack results only reduce the range and cannot guarantee to find a record-level identity. As for model inversion, though it can directly generate synthetic images from the statistical distribution of training data, the results are look-alike alternatives (not the original data) and only works when all class members are similar. Here we consider a more challenging question: Without prior about the training data, can we **completely** steal the training data from gradients? Conventional wisdom suggests that the answer is no, but we will show it is actually possible.

3 Training Data Leakage from Gradients

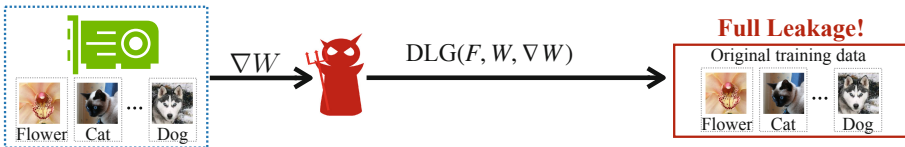


Fig. 6. Deep Leakage in federated settings. Given model updates/gradients received from victim, it aims to reserve the gradients and fully reconstructs the private training set.

While information leakage already intrudes participants’ privacy in Federated Learning, we are curious about the limit of the leakage – can we completely reconstruct the private training set from shared information? Federated Learning aims to train a robust model without sharing data, but now knowing the gradients is knowing the data. This question is critical as it raises a serve challenge to the fundamental privacy assumption. In this section, we will study the challenge and demonstrate the potential risks brought by such leakage (Fig. 6).

3.1 Partial Leakage in Specific Layers

To begin with, we start with several special layers. The first one is *Fully Connected* (FC) layers. FC layers are widely used in both Neural Networks (NN) and Convolutional Neural Networks (CNN). For a biased FC layer, we can show that the corresponding input can be computed from the gradients regardless where the layer’s position is and the types of preceding and succeeding layers.

Lemma 1. *Let a neural network contain a biased fully-connected layer, i.e. for the layer’s input $X \in \mathbb{R}^n$, its output $Y \in \mathbb{R}^m$ is calculated as*

$$Y = WX + B \tag{1}$$

with weight $W \in \mathbb{R}^{m \times n}$ and bias $B \in \mathbb{R}^m$. The input X can reconstructed from $\frac{dL}{dW}$ and $\frac{dL}{dB}$ if there exists index i s.t. $\frac{dL}{d(B_i)} \neq 0$.

Proof 1. It is know that $\frac{dL}{d(B_i)} = \frac{dL}{dY_i} \frac{d(Y_i)}{d(W_i)} = X^T$. Therefore

$$\frac{dL}{d(W_i)} = \frac{dL}{d(Y_i)} \cdot \frac{d(Y_i)}{d(W_i)} = \frac{dL}{d(B_i)} \cdot X^T \tag{2}$$

where the Y_i , W_i and B_i denote the i^{th} row of output Y , weights W and biases B . Thus X can be reconstructed as long as $\frac{dL}{d(B_i)} \neq 0$. \square

The knowledge of the derivative w.r.t. the bias $\frac{dL}{dB}$ is essential for reconstructing the layer’s input. To make the leakage more general, Geiping *et al.* [12] further proved that even without biases, the input could also be reconstructed as long as a proper activation follows (e.g., ReLU). The proof procedure is similar and no optimization is required to reconstruct the training set from a fully connected network.

Even without inverting the derivatives, the gradients from some layers already indicate certain information about the input data. For example, the *Embedding* layer in language tasks only produces gradients for words appeared in the data, which reveals what words have been used in other participant’s training set [7]. Another example is the *Cross Entropy* layers in classification tasks, which only generate negative gradients for class marked as corrected in the training set [13]. This property implies what the ground truth label is.

However, it is not trivial to extend to *Convolution* layers (CONV), where the dimension of features is far larger than the size of gradients. The analytical reconstruction like Lemma 1 is no longer practical. A more general attack algorithm is required for modern convolutional neural networks.

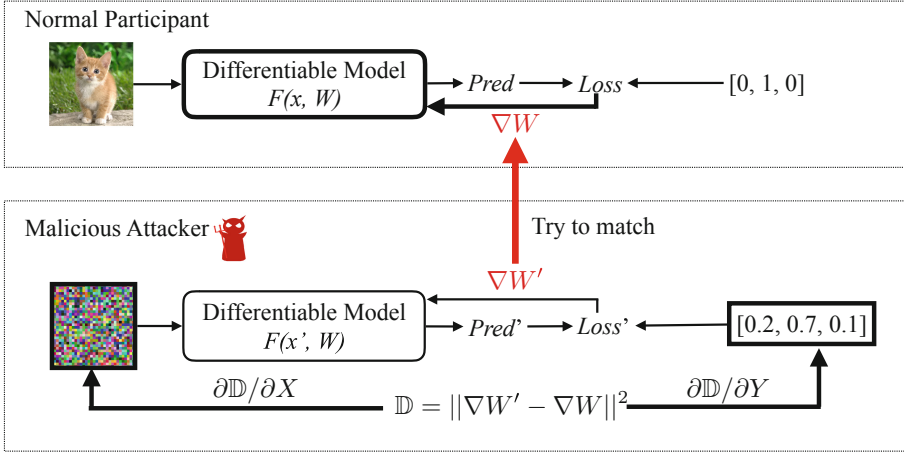


Fig. 7. The overview of DLG algorithm. Variables to be updated are marked with a bold border. While normal participants calculate ∇W to update parameter using its private training set, the malicious attacker updates its dummy inputs and labels to minimize the gradients distance. When the optimization finishes, the evil user is able to steal the training set from honest participants.

3.2 Complete Leakage from Gradients

To overcome the limitation, Zhu *et al.* [14] proposes an iterative method to **fully** reconstruct the training set, which was meant to be local and private, by only intercepting the corresponding gradients on the same neural network. The technique is named as *Deep Leakage from Gradients* (DLG), because of the “deep” threat it raises to user data’s privacy.

DLG is a gradient-based feature reconstruction attack. The attacker receives the gradient update $\nabla W_{t,k}$ from other participant k at round t , and aims to steal participant k ’s training set $(\mathbf{x}_{t,k}, \mathbf{y}_{t,k})$ from the shared the information. Figure 7 presents how it works for federated learning on images: The algorithm first initializes a dummy image with the same resolution as real one’s and a dummy label with probability representation followed by a softmax layer. DLG runs a test of this attack on the intermediate local model to compute “dummy” gradients. Note the model architecture $F()$ and weights W_t are shared by default for most federated learning applications.

Then a gradient distance loss between dummy gradients and real ones is calculated as the optimization objective. The key point of this reconstruction attack is to iteratively refine the dummy image and label so that the attacker’s dummy gradients will approximate the actual gradients. When the gradient construction loss is minimized, the dummy data will also convergence to the training data with high confidence (examples shown in Sect. 3.3).

$$\mathbf{x}'^*, \mathbf{y}'^* = \arg \min_{\mathbf{x}', \mathbf{y}'} \|\nabla W' - \nabla W\|^2 = \arg \min_{\mathbf{x}', \mathbf{y}'} \left\| \frac{\partial \ell(F(\mathbf{x}', W), \mathbf{y}')}{\partial W} - \nabla W \right\|^2 \quad (3)$$

Algorithm 1. Deep Leakage from Gradients for Masked Language Model

Input: $F(\cdot)$: Differentiable machine learning model; W : parameter weights; ∇W : gradients calculated by training data; η : learning rate used for DLG optimization.

Output: the original private training data \mathbf{x} and label \mathbf{y}

```

1: procedure DLG( $F, W, \nabla W$ )
2:    $\mathbf{x}'_1 \leftarrow \mathcal{N}(0, 1)$ ,  $\mathbf{y}'_1 \leftarrow \mathcal{N}(0, 1)$            ▷ Initialize dummy inputs and labels.
3:   for  $i \leftarrow 1$  to  $n$  do
4:      $\mathbf{L}'_i = \text{softmax}(\mathbf{y}'_i)$ 
5:      $\nabla W'_i \leftarrow \partial \ell(F(\mathbf{x}'_i, W), \mathbf{L}'_i) / \partial W_i$            ▷ Compute dummy gradients.
6:      $\mathbb{D}_i \leftarrow \|\nabla W'_i - \nabla W\|^2$ 
7:      $\mathbf{x}'_{i+1} \leftarrow \mathbf{x}'_i - \eta \nabla_{\mathbf{x}'_i} \mathbb{D}_i$            ▷ Update data to match gradients.
8:      $\mathbf{y}'_{i+1} \leftarrow \mathbf{y}'_i - \eta \nabla_{\mathbf{y}'_i} \mathbb{D}_i$            ▷ Update label to match gradients.
9:   end for
10:  return  $\mathbf{x}'_{n+1}, \mathbf{y}'_{n+1}$ 
11: end procedure

```

Note that the distance $\|\nabla W' - \nabla W\|^2$ is differentiable w.r.t dummy inputs \mathbf{x}' and labels \mathbf{y}' can thus be optimized using standard gradient-based methods. Therefore this optimization requires 2^{nd} order derivatives. A mild assumption that F is twice differentiable is made here. This holds for the majority of modern machine learning models (e.g., most neural networks) and tasks.

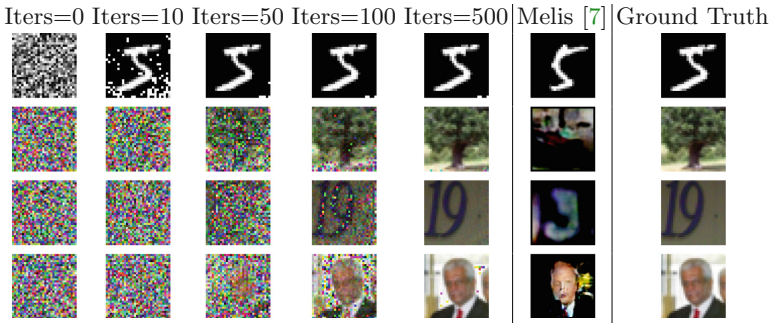


Fig. 8. The visualization showing the deep leakage on images from MNIST [15], CIFAR-100 [16], SVHN [17] and LFW [9] respectively. Our algorithm fully recovers the four images while previous work only succeeds on simple images with clean backgrounds.

3.3 DLG Attack on Image Classification

Given an image containing objects, images classification aims to determine the class of the item. The power of DLG attack is first evaluated on modern CNN architectures ResNet [18] and pictures from MNIST [15], CIFAR-100 [16], SVHN [17] and LFW [9]. Note that two changes have been made here: (1)

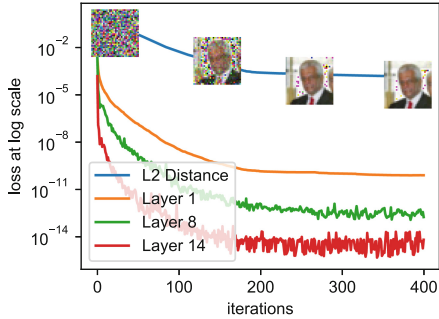


Fig. 9. Layer- i means MSE between real and dummy gradients of i^{th} layer. When the gradients’ distance gets smaller, the MSE between leaked image and the original image also gets smaller.

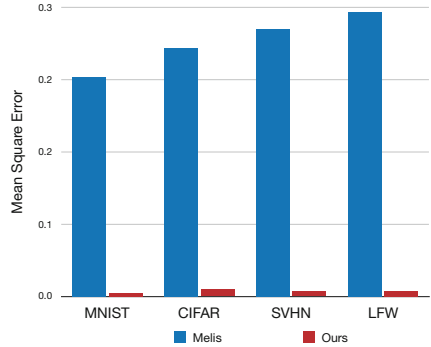


Fig. 10. Compassion of the MSE of images leaked by different algorithms and the ground truth. DLG method consistently outperforms previous approach by a large margin.

For model architecture, all ReLU operators are replaced with Sigmoid and the strides in CONV are removed, as our algorithm requires the model to be twice-differentiable (2) For image labels, instead of directly optimizing the discrete categorical values, we random initialize a vector with shape $N \times C$ where N is the batch size and C is the number of classes, and then take its softmax output as the classification label for optimization and DLG attack.

The leaking processes are visualized in Fig. 8. All DLG attacks start with random Gaussian noise (first column) and tried to match the gradients produced by the dummy data and real ones. As shown in Fig. 9, minimizing the distance between gradients also reduces the gap between data and makes the dummy data gradually converge to the original one. It is observed that monochrome images with a clean background (MNIST) are easiest to recover, while complex images like face (LFW) take more iterations to recover (Fig. 8). When the optimization finishes, the reconstructed results are almost identical to ground truth images, despite few negligible artifact pixels.

We compare the effectiveness of DLG attack with the GAN Inversion results [7] (discussed in Sect. 2) in Fig. 8 (visually) and Fig. 10 (numerically). The previous GAN based inversion requires the class label to be known and only works well on MNIST. On the 3^{rd} row and 6^{th} column of Fig. 8, though the revealed image on SVHN is still visually recognizable as digit “9”, it is far different from the original training image. The cases are even worse on LFW and totally collapse on CIFAR. Figure 10 shows a numerical comparison by performing leaking and measuring the mean square error (MSE) on all dataset images. Images are normalized to the range $[0, 1]$ and DLG appears much better results (<0.03 v.s. previous >0.2) on all four datasets.

3.4 DLG Attack on Masked Language Model

Table 1. The progress of deep leakage on language tasks.

	Example 1	Example 2	Example 3
Initial Sentence	tilting fill given **less word **itude fine **nton overheard living vegas **vac **vation *f forte **dis cerambycidae ellison **don yards marne **kali	toni **enting asbestos cutler km nail **oof **dation **ori righteous **xie lucan **hot **ery at **tle ordered pa **eit smashing proto	[MASK] **ry toppled **wled major relief dive displaced **lice [CLS] us apps - **face **bet
Iters = 10	tilting fill given **less full solicitor other ligue shrill living vegas rider treatment carry played sculptures lifelong ellison net yards marne **kali	toni **enting asbestos cutter km nail undefeated **dation hole righteous **xie lucan **hot **ery at **tle ordered pa **eit smashing proto	[MASK] **ry toppled identified major relief gin dive displaced **lice doll us apps _ **face space
Iters = 20	registration, volunteer applications, at student travel application open the; week of played; child care will be glare	we welcome proposals for tutor **ials on either core machine denver softly or topics of emerging importance for machine learning	one **ry toppled hold major ritual ' dive annual conference days 1924 apps novelist dude space
Iters = 30	registration, volunteer applications, and student travel application open the first week of september. Child care will be available	we welcome proposals for tutor **ials on either core machine learning topics or topics of emerging importance for machine learning	we invite submissions for the thirty - third annual conference on neural information processing systems
Original Text	Registration, volunteer applications, and student travel application open the first week of September. Child care will be available	We welcome proposals for tutorials on either core machine learning topics or topics of emerging importance for machine learning	We invite submissions for the Thirty-Third Annual Conference on Neural Information Processing Systems

For language tasks, DLG is evaluated on Masked Language Model (MLM) task. In each sequence, 15% of the words are replaced with a [MASK] token and MLM model attempts to predict the original value of the masked words from a given context. BERT [19] is chosen as the backbone and all hyperparameters are adopted from the official implementation¹.

Different from vision tasks where RGB inputs are continuous values, language models need to preprocess discrete words into embeddings. Therefore, on language model DLG attack is applied on embedding space and the gradients distance between dummy embeddings and real ones is minimized. After optimization finishes, the original words are derived by finding the closest entry in the embedding matrix reversely.

Table 1 exhibits the leaking history on three sentences selected from NeurIPS conference page. Similar to the vision task, DLG attack starts with randomly initialized embedding: The reconstructed results at iteration 0 is meaningless. During the optimization, the gradients produced by dummy embedding gradually match the original data’s gradients and the dummy embeddings also converges to the original data’s embeddings. In later iterations, part of the original sequence appears. In example 3, at iteration 20, “annual conference” shows up at iteration 30 and the leaked sentence is already close to the original one. When DLG finishes, though there are few mismatches caused by the ambiguity in tokenizing, the main content is already fully leaked.

3.5 Extensions to DLG Attack

In Algorithm 1, many factors can affect the leakage results such as the data initialization (line 2), the distance measurement between two gradients (line 6), and the optimization method (line 7 & 8). Besides these, hyper-parameters in Federated Learning like batch size and local steps also matters. In some cases, DLG may fail to reveal (e.g., with a bad initialization). To improve the stability of DLG, several approaches have been explored.

Leakage of Label Information on Classification Tasks. DLG attack is based on the belief that there is a one-to-one mapping between gradients and training data. Therefore if DLG does not discover the ground truth data, the attack will fail to converge. For tasks with cross entropy loss, Zhao *et al.* [13] proposes an analytical solution to extract the ground-truth labels from the shared gradients. When the differentiable model is trained with one-hot supervisions, the loss is computed as

$$L(X, c) = -\log \frac{e^{Y_c}}{\sum_j e^{Y_j}} \quad (4)$$

where the corresponding derivative is

$$g_i = \frac{\partial L(X, c)}{\partial Y_i} = \begin{cases} -1 + \frac{e^{Y_i}}{\sum_j e^{Y_j}}, & \text{if } i = c \\ \frac{e^{Y_i}}{\sum_j e^{Y_j}}, & \text{else} \end{cases} \quad (5)$$

¹ <https://github.com/google-research/bert>.

It is known that the softmax probability $e^{Y_c} / \sum_j e^{Y_j} \in (0, 1)$. Therefore, only the index with ground truth label yields negative gradients

$$g_i \in \begin{cases} (-1, 0) & \text{if } i = c \\ (0, 1) & \text{else} \end{cases} \quad (6)$$

Through the observation, the ground truth label can directly obtained and the leakage process becomes more stable and efficient with the extracted label.

Choices of Gradient Distances Loss. In the original DLG algorithm, the reconstruction optimizes the euclidean distances (also known as mean squared error) between two gradients via L-BFGS optimizer.

$$\arg \min_{x \in [0,1]^n} \|\nabla_w F(x, y; w) - \nabla_w F(x^*, y; w)\|_2^2 \quad (7)$$

where x^* indicates the original training input. Note that here the label y is assumed known via the trick introduced above. Geiping *et al.* [12] suggests that the magnitude appears not to be an important factor. Instead, the direction of gradients matters more during the leaking process. They propose to reconstruct based on cosine similarity $l(x, y) = \frac{\langle x, y \rangle}{\|x\| \|y\|}$ and the optimization objective becomes

$$\arg \min_{x \in [0,1]^n} 1 - \frac{\langle \nabla_w F(x, y; w), \nabla_w F(x^*, y; w) \rangle}{\|\nabla_w F(x, y; w)\| \|\nabla_w F(x^*, y; w)\|} + \alpha TV(x) \quad (8)$$

The term $TV(x)$ is a simple image prior *total variation* [20]. They include this as an extra regularization to ensure the leaked results is realistic. Figure 11 shows


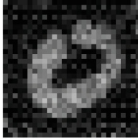





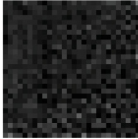


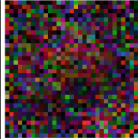

LeNet					
Input	Euclidean	Cosine	Input	Euclidean	Cosine
					
	0.82	66.57		37.82	29.85
ResNet20					
Input	Euclidean	Cosine	Input	Euclidean	Cosine
					
	-3.29	23.48		6.81	25.15

Fig. 11. Comparison between euclidean distance and cosine similarity on MNIST [15] (Left) and LFW [9] (Right) datasets. The number shown below the Figure is the PSNR (the larger the better).

the comparison between two losses. The proposed objective (Eq. 8) performs better especially complex CNN architectures.

Different Initialization. Wei *et al.* [21] analyzes the convergence of DLG on a single layer neural network, and proves that the convergence rate is $O(\frac{\|X_0 - X^*\|_2^2}{T})$, where T is the attack iterations. According to the results, the attack speed is closely related to the initialization of x_0 . The default way to initialize dummy data is to sample from a uniform distribution. Though such initialization works in most scenarios [12–14], it is not optimal and sometimes may fail to converge. To address the issue, they study various initialization. The ideal initialization is to use a natural image from the same classes as the private training set. Though this initialization requires the least iterations to converge, it needs extra prior about user data, which may not always be available. As an alternative, the geometric initialization [22] is a more general approach to boost up the attack.

4 Defense Strategies

4.1 Cryptology

Cryptology can be applied to prevent the leakage: Bonawitz *et al.* [23] designs a secure aggregation protocol and Phong *et al.* [24] proposes to encrypt the gradients before sending. Among all defenses, cryptology is the most secure one and can perfectly defend the leakage in theory. However, most cryptology-based defense strategies have their limitations. Secure aggregation [23] requires gradients to be integers thus not compatible with most CNNs, secure outsourcing computation [25] only supports limited operations, and homomorphic encryption [26] involves a large computation overhead and slows the whole pipeline. Therefore, in practice we are more interested in those lightweight defense strategies.

4.2 Noisy Gradients

One straightforward attempt to defend DLG is to add noise on gradients before sharing. To evaluate, we experiment Gaussian and Laplacian noise (widely used in differential privacy studies) distributions with variance range from 10^{-1} to 10^{-4} and central 0. From Fig. 12a and b, we observe that the defense effect mainly depends on the magnitude of distribution variance and less related to the noise types. When variance is at the scale of 10^{-4} , the noisy gradients do not prevent the leak. For noise with variance 10^{-3} , though with artifacts, the leakage can still be performed. Only when the variance is larger than 10^{-2} and the noise is starting affect the accuracy, DLG will fail to execute. We also notice that Laplacian tends to slightly a better defense when both at scale 10^{-3} .

Another common perturbation on gradients is half precision, which was initially designed to save memory footprints and widely used to reduce communication bandwidth. We test two popular half precision implementations IEEE float16 (*Single-precision floating-point format*) and bfloat16 (*Brain Floating Point* [27], a truncated version of 32 bit float). Shown in Fig. 12c, unfortunately, neither half precision format is able to protect the training data.

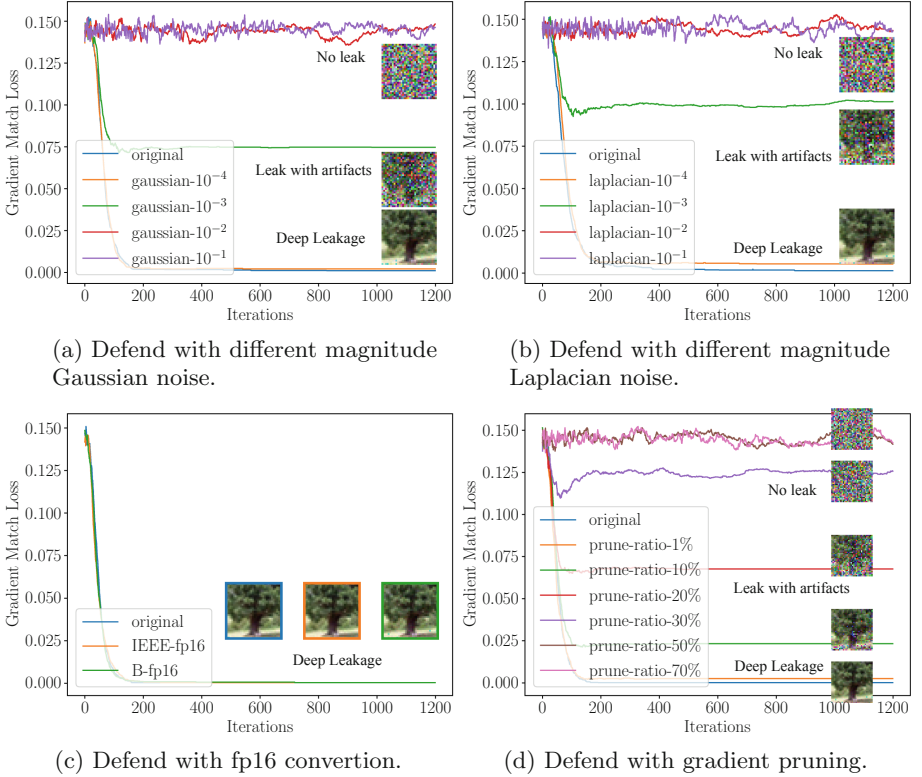


Fig. 12. The effectiveness of various defense strategies. The corresponding accuracy is attached in Table 2.

Table 2. The trade-off between accuracy and defendability. **G**: Gaussian noise, **L**: Laplacian noise, **FP**: Floating number, **Int**: Integer quantization. \checkmark means it successfully defends against DLG while \times means fails to defend (whether the results are visually recognizable). The accuracy is evaluated on CIFAR-100.

	Original	G - 10^{-4}	G - 10^{-3}	G - 10^{-2}	G - 10^{-1}	FP -16
Accuracy	76.3%	75.6%	73.3%	45.3%	$\leq 1\%$	76.1%
Defendability	–	\times	\times	\checkmark	\checkmark	\times
		L - 10^{-4}	L - 10^{-3}	L - 10^{-2}	L - 10^{-1}	Int -8
Accuracy	–	75.6%	73.4%	46.2%	$\leq 1\%$	53.7%
Defendability	–	\times	\times	\checkmark	\checkmark	\checkmark

4.3 Gradient Compression and Sparsification

We also experimented to defend by gradient compression [28, 29]. Gradient compression prunes small gradients to zero, therefore it’s more difficult for DLG to match the gradients since the optimization target also gets pruned. We evaluate

how different level of sparsities (range from 1% to 70%) defense the leakage. When sparsity is 1% to 10%, it has almost no effects against DLG. When prune ratio increases to 20%, as shown in Fig. 12d, there are obvious artifact pixels on the recover images. We notice that maximum tolerance of sparsity if around 20%. When pruning ratio is larger than 20%, the recovered images are no longer visually recognizable and thus gradient compression successfully prevents the leakage.

Previous work [28,29] show that gradients can be compressed by more than 300 *times* without losing accuracy. In which case, the sparsity is above 99% and already exceeds the maximum tolerance of DLG (which is around 20%). It suggests that compressing the gradients can be a good practical approach to avoid the leakage.

References

1. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data, arXiv preprint [arXiv:1602.05629](https://arxiv.org/abs/1602.05629) (2016)
2. Jochems, A., et al.: Developing and validating a survival prediction model for NSCLC patients through distributed learning across 3 countries. *Int. J. Radiat. Oncol. Biol. Phys.* **99**(2), 344–352 (2017)
3. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **10**(2), 1–19 (2019)
4. Konečný, J., McMahan, H.B., Yu, F.X., Richtarik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. In: *NIPS Workshop on Private Multi-Party Machine Learning* (2016). <https://arxiv.org/abs/1610.05492>
5. Bonawitz, K., et al.: Towards federated learning at scale: system design. *CoRR*, vol. abs/1902.01046 (2019). <http://arxiv.org/abs/1902.01046>
6. 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)
7. Melis, L., Song, C., Cristofaro, E.D., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. *CoRR*, vol. abs/1805.04049 (2018). <http://arxiv.org/abs/1805.04049>
8. Yang, D., Zhang, D., Yu, Z., Yu, Z.: Fine-grained preference-aware location search leveraging crowdsourced digital footprints from LBSNs. In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 479–488 (2013)
9. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. University of Massachusetts, Amherst, Technical Report 07-49, October 2007
10. 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. ACM (2015)
11. Goodfellow, I., et al.: Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, pp. 2672–2680 (2014)

12. 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](https://arxiv.org/abs/2003.14053) (2020)
13. Zhao, B., Mopuri, K.R., Bilen, H.: iDLG: improved deep leakage from gradients. arXiv preprint [arXiv:2001.02610](https://arxiv.org/abs/2001.02610) (2020)
14. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Annual Conference on Neural Information Processing Systems (NeurIPS) (2019)
15. LeCun, Y.: The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Citeseer, Technical report 2009
17. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, vol. abs/1810.04805 (2018). <http://arxiv.org/abs/1810.04805>
20. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D Nonlinear Phenom. **60**(1–4), 259–268 (1992)
21. Wei, W., et al.: A framework for evaluating gradient leakage attacks in federated learning. arXiv preprint [arXiv:2004.10397](https://arxiv.org/abs/2004.10397) (2020)
22. Rossi, F., Gégout, C.: Geometrical initialization, parametrization and control of multilayer perceptrons: application to function approximation. In: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN 1994), vol. 1, pp. 546–550. IEEE (1994)
23. Bonawitz, K., et al.: Practical secure aggregation for federated learning on user-held data. CoRR, vol. abs/1611.04482 (2016). <http://arxiv.org/abs/1611.04482>
24. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2018)
25. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_15
26. Armknecht, F., et al.: A guide to fully homomorphic encryption, Cryptology ePrint Archive, Report 2015/1192 (2015). <https://eprint.iacr.org/2015/1192>
27. Tagliavini, G., Mach, S., Rossi, D., Marongiu, A., Benini, L.: A transprecision floating-point platform for ultra-low power computing. CoRR, vol. abs/1711.10374 (2017). <http://arxiv.org/abs/1711.10374>
28. Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W.J.: Deep gradient compression: reducing the communication bandwidth for distributed training. arXiv preprint [arXiv:1712.01887](https://arxiv.org/abs/1712.01887) (2017)
29. Tsuzuku, Y., Imachi, H., Akiba, T.: Variance-based gradient compression for efficient distributed deep learning. arXiv preprint [arXiv:1802.06058](https://arxiv.org/abs/1802.06058) (2018)