# Investigating Deep Leakage from Gradients (DLG)

Kelvin Kan, Chang Meng, David Wang

May 13, 2022

## 1 Motivation and Contributions

Deep Leakage from Gradients (DLG) is a method of data privacy attack where private training data can be obtained from shared gradients. In cases where people assume gradients are safe to share, studies have shown that gradients reveal properties of training data [8]. In the case of DLG, it is motivated based on the idea of completely stealing the training data from publicly available gradients (e.g., from centralized parameter server or neighboring clients). The implementation of this task is through an optimization problem given by

$$\min_{\mathbf{x},\mathbf{c}} ||\nabla\tilde{\mathbf{W}}(\mathbf{x},\mathbf{c}) - \nabla\mathbf{W}||_2^2, \tag{1}$$

where $\mathbf{x}$ and $\mathbf{c}$ are dummy input and output to optimize, $\tilde{W}$ is the corresponding dummy gradient. DLG performs minimization on (1). In particular, it initialize a random dummy input and output, and then apply gradient descent algorithm to (1). Since the truth data and label are the minimizer of (1), by solving the problem (1) the attack can recover them.

DLG is based on the assumptions that 1) the model is shared among individual clients and the central server (if any) so that both network architecture and model weights are known, and 2) gradients are shared between server and client or neighboring clients. Under these assumptions, DLG can easily recover training data and labels, including pixel-wise accurate images, and token-wise matching texts. We are impressed by the results presented in [8], but at the same time we have two major questions which are not discussed in the paper:

- Is a successful recovery guaranteed? If not, what is the success rate of DLG?
- How does the performance of DLG differ for different models (i.e., different model architectures/complexity?)

In this project, we seek to answer the above questions by evaluating DLG on different models to recover training data. We focus on the image classification task, and run DLG with different training images and model architectures. For each case, we would like to know how easy it is for DLG to recover the original training data. We do so by evaluating the DLG success rate (defined later in Section 3), and by comparing the recovered images to the ground truths.

Through experiments, we have discovered that although DLG is a simple algorithm, it actually poses a difficult optimization problem (more discussion in Section 5). While DLG is able to successfully recover training images when a simple model is used (LeNet), it does not produce consistent results with a more complex model (ResNet-18). It is in fact generally infeasible to recover training images with ResNet-18, as shown in Section 4. We further look into improved Deep Leakage from Gradients (iDLG) [7] (more on this in Section 2) to test out the possibility of improving the quality of recovered images. The iDLG implementation is similar to that of the DLG except that it assumes the true label is known and the gradients with respect to every training batch is available.

## 2 Related Work

In this section, we review some literature on privacy leakage attacks, with a focus on leakage from gradients.

In [6], the authors focus on the methods of evaluating gradient leakage attacks. To do so, the study presented a framework for evaluation and comparison of different privacy leakage attacks. They show analysis on methods of reconstructing private local training data, determine how hyperparameter configurations matter to attack algorithm for effectiveness and cost, and highlight importance of systematic attack evaluation framework to privacy leakage threats.

When it comes to gradient leakage, [4] shows a "shallow" approach to gradient leakage in a method of feature leakage. Specifically, it is shown that model updates (i.e. gradients) already leak certain levels of information about participants' training data known as unintended information. To do so, the authors show that adversaries can predict exact data points in training data such as by membership inference, then they show that properties only hold for specific subsets of training data. The results show the dangers of leakage of unintended features whereas defenses such as selective gradient sharing, reducing dimensionality, and dropout are not effective. Nevertheless, this approach of only considering unintended features is "shallow" as the leakage is unordered (e.g unordered words cannot form a complete sentence) and fail to extend to convolutional layers due to size of feature being larger than size of weights.

In Deep Leakage from Gradients (DLG) [8], the authors introduce a new concept of "deep" attack to gradient leakage, and demonstrates the possibility to obtain private training data from publicly shared gradients. This algorithm is further expanded upon by showing how it only requires gradients to reveal pixel-wise accurate images/token-wise matching texts, and how to protect against DLG attack via several defense strategies.

However, it is noted in [7] that DLG has difficulty in convergence and discovering the ground-truth labels consistently. Hence, the paper proposes iDLG, a simple method of extracting ground-truth labels through a gradient sign/label rule in shared gradients. This extraction is performed before DLG to get one hundred percent accuracy of extracted labels. However, this algorithm may only be applied if gradients with respect to all samples in training batch is provided.

# 3 Approach and Methodology

To reiterate some points of emphasis of the experimentation, [8] have shown that shared gradients may be attacked to obtain private training data. They have proposed the methodology called Deep Leakage of Gradients (DLG) in order to perform such an attack. This problem not only demonstrates a vital issue but the need for defense against attacks such as the DLG. Nevertheless, prior to making such assumptions and tests to defend against DLG, it is important to note and evaluate DLG as an attack. As such, the approach of the experimentations in this study would be to find how easy is it for DLG to recover training data through evaluating their success rate, which will be expanded on later. Furthermore, the success rate will be evaluated when the architecture becomes more complicated.

The experiments were set up by focusing on the image classification task. LeNet and ResNet, which are two different convolutional neural network (CNN) model architectures, were utilized. Both LeNet and ResNet are state-of-the-art image classification methodologies to recognize patterns with extreme variability. LeNet, often referred to as LeNet-5, was one of the earliest CNNs that was simple and made up of seven layers. The layers were broken down into three convolution layers, two subsampling/pooling layer that outputs feature graphs, and two fully connected layers to the convolution layers [2]. This CNN architecture was commonly applied early on in handwritten digit recognition to which the CNN was trained by backpropagation algorithms to read handwritten numbers and apply it to identify handwritten zip code numbers given by the US Postal Service [2]. ResNet, on the other hand, is a modern CNN method that can be utilized for up to more than 1000 layers, where the architecture would be denoted as ResNet-18 for 18 layers, ResNet-32 for 32 layers and so on. In these layers, two are unique convolution and average pool layers while the rest are all convolution layers [5]. Given the ResNet-18 as an example, it is made up of 16 convolution layers, one max pool layer, and one average pool layer. Along with the utilization of these different layers, the ResNet architecture also have skip connections to form shortcuts in jumping over certain layers for alternate paths in the network and batch normalization to recenter and rescale the layer inputs for speed and stability [5].

Alongside the proposed CNN architectures implemented to test DLG attacks, DLG algorithm was implemented with the limited-memory BFGS (L-BFGS), a Quasi-Newton method, as the gradient descent optimizer with history size 100, max iterations of 20 to solve the large scale DLG optimization with the two network architectures. The optimizer ran for 1200 iterations for the image task. L-BFGS

works by using a small amount of memory to store approximate Hessian values of the DLG function [3]. Furthermore, randomly initialized weights were utilized akin to that of the DLG paper. By doing so, it simulates the situations of the DLG attack at any time during the training process. Based on the DLG algorithm, the success rate was calculated based on the percentage of training images recovered after the convergence criterion has been met, where the convergence criterion was set to occur when the gradient loss is less than a value of 0.0001. Mathematically, the success rate is the percentage of seeds such that both DLG loss and relative errors converge. Note that the convergence of DLG loss alone does not guarantee the convergence of the training image we want to recover. As we will observe in Section 4, for ResNet-18, the MSE for the training image actually diverges as the loss of gradients converge. However, for LeNet, we have observed that both loss and MSE converge simultaneously.

The goal of the overall experiments was to recover 20 randomly selected training images from the CIFAR-100 dataset. The CIFAR-100 dataset consists of 60000 32x32 color images in 100 classes, with 600 images per class (500 for training and 100 for testing) [1].

Based on the different architectures and experimental set up, the DLG algorithm was implemented. For LeNet, each of the 20 randomly selected training images were tested with 50 randomly selected seeds to calculate the success rate of DLG, i.e., we compute the percentage of seeds that allow us to successfully reconstruct the training image, and average the numbers for the 20 randomly selected images. For ResNet, each of the 20 randomly selected training images were trained with only 20 randomly selected seeds due to the increased model complexity hence runtime. Adding on, different ResNet architectures with differing number of layers were utilized including that of ResNet-18. From implementations, the DLG success rate and the mean iteration number after reaching the convergence criterion were recorded.

We are also interested in learning how iDLG (improved DLG) improves over DLG. Recall that iDLG is based on the assumption that ground truth labels can be extracted by mathematical formula, which is only true when gradients with respect to all training batches are known. Hence, in Section 4, we also show side by side comparison between iDLG and DLG iterates on the same training image, and illustrate through convergence plots how these two methods compare.

# 4 Evaluation and Results

We begin the discussion with the results for the LeNet architecture. Depending on the target images, the DLG success rate ranges from 0.58 to 0.78. Furthermore, the lowest mean iteration number to achieve convergence ranges from 77 to 126. When these metrics were averaged for all 20 randomly selected image, it produced a DLG success rate of 0.68 and a mean iteration number at convergence of roughly 94 as seen in Table 1. The success rate measured the difficulty to recover the training data. Based on the results to recover image data from LeNet, it showed the relative ease to recover training data from the simpler architecture of LeNet. During evaluation, it is also necessary to note that it is not essential for the DLG success rate to be perfect at 100%. The evaluation decision being made is if DLG works to attack shared gradients where as long as if a decent amount of (or even at least one) randomly selected seeds were successfully attacked by DLG, the training data was leaked and the DLG algorithm proposed by the paper works on the architecture.

| DLG Success Rate | Mean Iter Number at Convergence |
|---|---|
| 0.68 | 94.1379 |

Table 1: Average success rate of deep leakage on LeNet CNN architecture. The experiment is performed with 20 randomly selected images, each with 20 random DLG recovery trials.

For ResNet-18, the performance of DLG was not optimal. First, in Figure 1, it shows a side-by-side comparison for ResNet-18 and LeNet and image with index 20319. This image was selected to be the model in this paper because it actually had the best performance when compared to the other randomly selected images with ResNet-18. As such, when examining the figure, we note that LeNet was generally able to reproduce/recover the original training image after convergence in around 90 iterations. ResNet-18, on the other hand, still had plenty of noise by iteration 90, and the DLG loss

has already converged, see 2(b). Moreover, it was unable to produce any DLG success rate to be reported. In fact, DLG was not successful even after the convergence criterion was reached where the gradient loss is less than a value of 0.0001. Nevertheless, when comparing the loss values of the two different architectures in Figure 2(a), ResNet-18 still produced a lower DLG loss in every iteration compared to that of LeNet. This discrepancy in DLG performance and loss could be attributed to the factor that a more complicated architecture has more different data which produce the same gradient, see Section 5 for more discussion.

In Figures 1 and 2 we have also included results involving iDLG.. To be specific, For each DLG reconstruction with respect to LeNet and ResNet, we also show reconstruction of iDLG for comparison. Recall that the iDLG implementation is the same as that of the DLG, except that it assumes the true labels are known, so iDLG only focuses on recovering the input image. For each convergence plot (loss or MSE), we also add corresponding curves for iDLG. We observe that although iDLG has the property of speeding up convergence by a little bit (as seen for LeNet iteration 30, where iDLG gives us a clearer image), it doesn't really improve the reconstruction accuracy towards the end. For ResNet-18, the upgrade to iDLG doesn't alter the convergence path of DLG to that a global minimum is reached, but instead converges to the same local minimum as DLG.

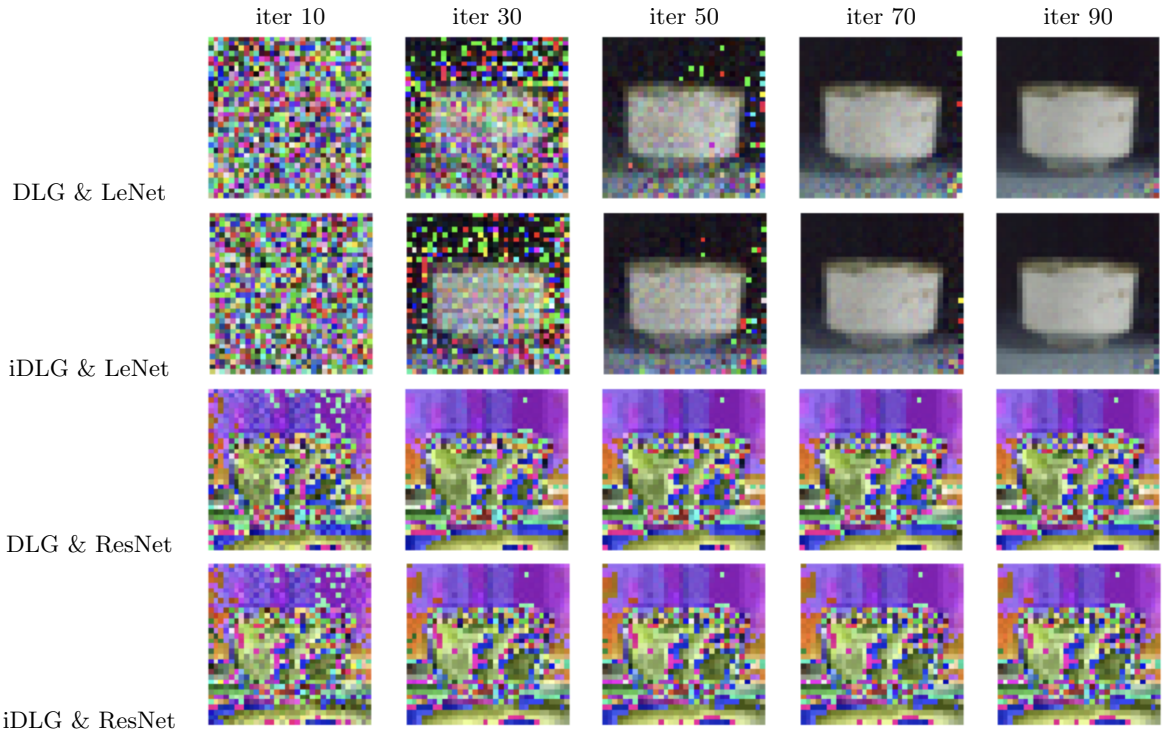
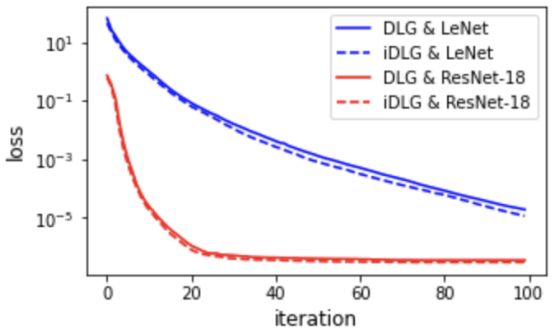
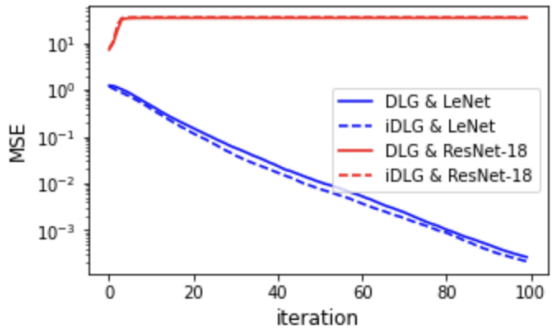



Figure 1: Juxtaposed ResNet-18 performance with LeNet on an image (image index = 20319) in the CIFAR-100 dataset. Both DLG and iDLG recoveries at different iterations are shown.




(a) DLG Loss




(b) MSE between true and recovered images

Figure 2: DLG Loss and MSE of ResNet-18 compared with those of LeNet on an image (image index = 20319) in the CIFAR-100 dataset.

# 5 Discussion

The successful experimental results reported in the DLG paper [8] disprove the traditional common conception that is it impossible to recover the original data and labels only from their gradients. However, the difficulty and implmentational challenge of recovering the data and labels using DLG remain uninvestigated in the literature. Our experiments on testing the success rate of original data recovery reveals that a successful recovery from DLG is not guaranteed. In particular, for a simple LeNet architecture, the success recovery rate is about 70%, and for a more complicated ResNet architecture, DLG was unable to recover a single image that visually resembles the original data.

In this section, we analyze the reason of the unsuccessful recovery in two angles: from an inverse problem and an optimization perspectives.

Firstly, we argue that there is no one-to-one correspondence between data and their corresponding gradients. To analyze this, we consider a simple linear learning problem given by

$$\min_{\mathbf{A}\in\mathbb{R}^{\mathbf{n_c}\times\mathbf{n_f}}} L(\mathbf{x},\mathbf{b}) = \frac{1}{2}||\mathbf{A}\mathbf{x}-\mathbf{b}||_2^2, \tag{2}$$

where $\mathbf{A}$ can be thought of as a simple linear "network", $\mathbf{x}\in\mathbb{R}^{n_f}$ is the input data and $\mathbf{b}\in\mathbb{R}^{n_c}$ is the target label, $n_f$ is the number of input features (e.g. number of pixels in each image), and $n_c$ is the number of output features (e.g. number of classes). In the learning problem, the goal is to compute $\mathbf{A}$ such that it represents the input-output relationship between $\mathbf{x}$ and $\mathbf{b}$. The gradient of $L$ with respect to $\mathbf{A}$ is

$$\nabla_{\mathbf{A}} L(\mathbf{x},\mathbf{b}) = \mathbf{A}\odot[(\mathbf{A}\mathbf{x}-\mathbf{b})\mathbf{e}_{n_f}^\top], \tag{3}$$

here $\odot$ denotes the Hadamard (element-wise) product, and $\mathbf{e}_{n_f}\in\mathbb{R}^{n_f}$ represents a vector of all ones. However, if $\boldsymbol{\epsilon}_{\mathbf{x}}$, the perturbation to the input data $\mathbf{x}$, belongs to the null space of $\mathbf{A}$, then we have

$$\nabla_{\mathbf{A}} L(\mathbf{x},\mathbf{b}) = \nabla_{\mathbf{A}} L(\mathbf{x}+\boldsymbol{\epsilon}_{\mathbf{x}},\mathbf{b}). \tag{4}$$

Note that in general the null($\mathbf{A}$) has a dimension of $n_f - n_c$, and $n_f$ is usually much greater than $n_c$. For instance, the CIFAR10 dataset has $n_f = 32\times 32$ and $n_c = 10$. Therefore, a successful DLG attempt to the learning problem (2) could generate any results $\mathbf{x}+\boldsymbol{\epsilon}_{\mathbf{x}}$, where $\boldsymbol{\epsilon}_{\mathbf{x}}\in$ null($\mathbf{A}$). And some of the generated results might not make sense visually, like the ones reported in Figure 1.

Secondly, the DLG optimization problem considers minimizing the Euclidean distance between the true gradient $\nabla\mathbf{W}$ and a "dummy" gradient $\nabla\mathbf{W}'$ of a randomly generated "dummy" data, that is

$$\min ||\nabla\mathbf{W}' - \nabla\mathbf{W}||_2^2, \tag{5}$$

This optimization might appear to be quadratic and hence has good optimization properties at first glance. However, note that the "dummy" gradient is actually the gradient of the underlying neural network with respect to its weights. In real practice, the networks usually have multiple layers and hence have complicated structure. Therefore, the complex architecture combined with the gradient operator

applied to the network render the DLG problem (5) highly nonconvex, and sometimes even nonsmooth. Therefore, it is likely that the optimization will end up at some sub-optimal local minimum, at which the recovered data can be very different from the true target data.

# 6 Conclusion and Future Work

In this project, we have investigated DLG for the task of revealing private training image from publicly shared gradients. We have observed that while DLG is possible to fully reconstruct training images, it doesn't always work, and its performance depends to some extent on randomness. As seen in Section 4, for the better-performing case of LeNet, the success rate is slightly below 70%. When it comes to the task of recovering training images for ResNet-18, a more complex network compared to LeNet, the performance of DLG is much worse – we are not able to fully reconstruct any training image. The DLG algorithm poses a difficult optimization problem, as discussed in Section 5, and the difficulty increases as the complexity of the underlying model increases. For both LeNet and ResNet, we are subject to the possibility of obtaining local minimums rather than global minimums. We have also experimented with improved DLG (iDLG), which only works in a simplified scenario where gradients with respect to all training batches are available. However, while iDLG improves in convergence slightly over DLG, it actually converges to the same image with DLG (i.e., when DLG converges to a local minimum, iDLG also does), hence not improving the quality of the reconstructed image.

We think an important future direction for making DLG more stable and robust is to develop problem-specific optimization solvers tailored to the DLG algorithm and specific model architectures. This would allow us to better understand the problem and perhaps derive more reliable defense strategies against gradient leakage attacks.

# References


[1] A. Krizhevsky, *The cifar-10 dataset*.

[2] Y. LeCun, L. D. Jackel, L. Bottou*, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Mulle, E. Sackinger, P. Simard, and V. Vapnik, *Learning algorithms for classification: A comparison on handwritten digit recognition*, (2000).

[3] D. Liu and J. Nocedal, *On the limited memory bfgs method for large scale optimization*, (1989).

[4] L. Melis, C. Song, E. D. Cristafaro, and V. Shmatikov, *Exploiting unintended feature leakage in collaborative learning*, (2018).

[5] S. Targ, D. Almeida, and K. Lyman, *Resnet in resnet: generalizing residual architectures*, (2016).

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

[7] B. Zhao, K. R. Mopuri, and H. Bilen, *idlg: Improved deep leakage from gradients*, (2020).

[8] L. Zhu, Z. Liu, and S. Han, *Deep leakage from gradients*, in Advances in Neural Information Processing Systems, 2019.