Analysing Training-Data Leakage from Gradients through Linear Systems and Gradient Matching

Cangxiong Chen[1]
cc2458@bath.ac.uk
Neill D. F. Campbell[2]
n.campbell@bath.ac.uk

[1] Institute for Mathematical Innovation, University of Bath, UK
[2] Department of Computer Science, University of Bath, UK

Abstract

Recent works have demonstrated that it is possible to reconstruct training images and their labels from gradients of an image-classification model when its architecture is known. Unfortunately, there is still an incomplete theoretical understanding of the efficacy and failure of these gradient-leakage attacks. In this paper, we propose a novel framework to analyse training-data leakage from gradients that draws insights from both analytic and optimisation-based gradient-leakage attacks. We formulate the reconstruction problem as solving a linear system from each layer iteratively, accompanied by corrections using gradient matching. Under this framework, we claim that the solubility of the reconstruction problem is primarily determined by that of the linear system at each layer. As a result, we are able to partially attribute the leakage of the training data in a deep network to its architecture. We also propose a metric to measure the level of security of a deep learning model against gradient-based attacks on the training data.

1 Introduction

For a neural network performing image classification, can we reconstruct the training image given its gradients, its label and the model architecture? More precisely, suppose we know the model $f(\boldsymbol{x};\boldsymbol{w})$ with parameters $\boldsymbol{w}$ that are given by either random initialisations or pre-training. Let $\boldsymbol{x}^*$ be the training image we wish to reconstruct and assume its label $\boldsymbol{y}^*$ is known. Furthermore, let $\mathcal{L}$ be the loss function for our classification problem. In mathematical terms, we are interested in inverting the mapping from $\boldsymbol{x}^*$ to its gradients $\nabla_{\boldsymbol{w}}\mathcal{L}(f(\boldsymbol{x}^*;\boldsymbol{w}),\boldsymbol{y}^*)$. This mapping is far from being injective in general, which makes this problem challenging. Besides being an interesting inverse problem, this problem has important implications in understanding the privacy risk in Federated Learning. For example, in the scenario where the local participants have sensitive data that are not to be shared with other participants or the central server, how can we guarantee that their data cannot be reconstructed from the gradients that are shared during collaborative training?

A number of works have tried to solve this reconstruction problem, which we refer to as 'gradient-leakage attacks'. Although each of these works can perform well in specific

situations, it remains unclear why they performed well in those situations and why they failed in other ones. Our goal in this paper is to provide a novel framework to understand the efficacy and failure of existing gradient-leakage attacks, so that we can get more insight into the nature of gradient leakage in deep learning. The main insight from our work is that the analytic method in [16] and the optimisation method in [17] can be unified under one framework (i.e. our hybrid framework) to understand gradient leakage attacks and shed light on the issue of gradient leakage itself. More precisely, we think the R-GAP method in [16] is essentially solving a least square problem by picking a particular representative in the family of solutions obtained from the Singular Value Decomposition (SVD). We claim that the idea of gradient matching proposed in [17] can be used to choose a better representative from the family of solutions than R-GAP. In this way, these two seemingly distinct methods can be viewed as two consecutive steps in solving the same least square problem. We hope that our framework to analyse gradient leakage can lead to further theoretical works on for example, proofs of the likelihood of a successful gradient-leakage attack for a given deep neural network.

Our contribution: Following the work in [16], we formulate the reconstruction problem as solving one linear system at each layer, iterating backward over the entire network. The linear system at each layer is defined by the forward and backward propagation of the target image during training. The linear system can only be solved approximately in general due to the size and rank of the coefficients. To reduce the approximation error of the solution, we propose and solve an optimisation problem using the idea of gradient matching inspired by the work in [17]: optimising the approximated solution so that the gradients of the loss function are close to those of the loss function evaluated at the target image according to the cosine distance function. By our formulation, the reconstruction is primarily determined by linear systems, which enables us to estimate the vulnerability of a deep network against gradient leakage attacks aiming at reconstructing the training image. We quantify this estimate by a novel metric, which is defined as a sum of rank-deficiency of the linear system at each layer, weighted by its position in the network. We apply our framework to convolutional networks with the number of layers ranging from two to four, which include either randomly initialised or pre-trained weights. The results have shown noticeable improvements over previous works.

2 Related Work

In [17], it is shown that one can reconstruct the training image fully for a single image in an untrained model performing classification tasks. In order to reconstruct a target image-label pair $\boldsymbol{x}^*, \boldsymbol{y}^*$ for an untrained model with randomly initialised weights $\boldsymbol{w}$, their method (which was referred to by the authors as 'Deep Leakage from Gradients' or DLG) proceeds as follows. Start with a randomly initialised image-label pair $(\boldsymbol{x}, \boldsymbol{y})$, minimise the $L2$ distance between the gradients from $(\boldsymbol{x}, \boldsymbol{y})$ and those from the target pair $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ by changing $(\boldsymbol{x}, \boldsymbol{y})$. The solution to this optimisation will be the desired reconstruction of the target $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. Although this method is able to reconstruct the target image and its label in high accuracy in some cases, it is prone to failure when we consider different target pairs or model architectures. Also, it is unable to handle pre-trained models. The work [15] provides an improvement to [17] by showing that one can always reconstruct the true label of the target image first so that the optimisation only needs to run with respect to the dummy input $\boldsymbol{x}$. This has improved the robustness of DLG. A further improvement is provided by [3], which

replaces the $L2$ distance in DLG by a combination of cosine similarity and a Total Variation $L1$ regulariser. Recently, [10] improves upon previous work by making use of GAN that has been trained on public datasets to provide image priors to guide the optimisation.

In a different line of thinking from the above formulation as optimisation problems, [1] showed one can invert a fully connected layer if the bias term is nonzero. The input to the layer can be expressed in closed form using gradients of the weights and of the bias. We make use of this idea and show that it is possible to extend this solution to the case when the bias is zero. Unfortunately the same close-form solution cannot be extended directly to the case of a convolutional layer, because the convolutional operation involves weight sharing in general. Fortunately, we can represent the convolution operation in a circulant representation described in [5]. In this way, convolution can be expressed as a single matrix multiplication, like in a fully-connected layer. Note that the circulant representation does not change the nature of weight sharing in a convolutional layer, so we cannot use the same method from the case of a fully-connected layer. This is in contrast to the view in [2], who treated both cases in a uniform manner.

Based on the circulant representation of convolution, [16] formulated a linear system of the input to each layer using the forward and backward propagation. By formulation, the training input must satisfy this linear system. Then by solving this linear system at each layer iteratively using pseudo inverse, they showed that we can ultimately solve the reconstruction problem. Furthermore, they formulated a so-called 'virtual constraint' to capture additional constraints that are not captured by the linear system at each layer. Unfortunately, they were not able to incorporate the 'virtual constraint' into their solution. We think that the 'virtual constraint' becomes redundant if we solve the linear systems jointly across the entire network. Empirically, we also found that there has been very little improvement over the reconstructions by incorporating the 'virtual constraint' into the objective function. In our work, we combine the linear system formulation from [16] with the idea of gradient matching from [17] and its improvement from [3]. On the one hand, our approach can correct errors inherent in solving the least square problem defined by the linear system in [16]. On the other hand, our approach improves gradient matching by introducing strong constraints defined by the linear systems. Unlike [10], we do not assume additional prior knowledge of the image we want to reconstruct. Our formulation of the reconstruction problem provides a framework for us to analyse the vulnerability of a deep network against gradient-leakage attacks through its architecture, which is not available through purely optimisation-based methods. Based on our framework, we propose a quantifying metric (12) to measure the likelihood of a successful reconstruction. Notice that [16] proposed a similar metric 'RA-i'. Our metric differs from 'RA-i' in three ways: 1. We do not incorporate the 'virtual constraint' based on our observation above; 2. Our metric is defined based on the rank of the linear system instead of the number of constraints; 3. We quantify the position of the layer by taking a weighted average of the rank-deficiency of the linear system at each layer.

3 A hybrid framework

Overall assumptions: We only consider the problem of reconstructing a single training image when the batch size is one. The target model for image-classification is assumed to consist of consecutive convolutional layers with the last layer being fully connected. We have considered 2, 3 and 4 layer CNNs with variations which represent different cases of changes in intermediate feature spaces. Our framework and analysis should apply to deeper

networks composed of concatenated shallow building blocks of CNNs, as we have specified an exhaustive set of those building blocks for CNNs. The quality of reconstruction can be estimated by the average of the rank deficiency from each layer weighted by the position of the layer in the network.

We assume the ground-truth label of the target image is given. This will not lose generality, because otherwise we can reconstruct the true label via observing the signs of the gradients of the weights in the fully connected layer according to [15]. We assume that the activation function for each layer is piecewise invertible and piecewise differentiable. For simplicity we do not include pooling in the design of the network. Notice that average pooling can be regarded as convolution. Also notice that CNN without pooling was considered in [13].

Notations We introduce notations used throughout the paper.

$\boldsymbol{w}^{(i)}$: weight from layer i of size m by n, where $0 \leq i \leq d$ and d is the total number of layers. For simplicity of notation, we omit i in m and n when possible, but readers should be aware that the size of the weight need not be the same for different layers. For a convolutional layer, it denotes the circulant representation of the kernel following [5].

$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$: Cross entropy loss function of the network with input image $\boldsymbol{x}$, label $\boldsymbol{y}$ and weight $\boldsymbol{w}$. We use $\mathcal{L}^{(i)}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$ for the shorthand notation denoting the loss with the truncated model starting from layer i with corresponding intermediate input $\boldsymbol{x}^{(i)}$, weight from the i-th layer onward. We will omit the label $\boldsymbol{y}$ where possible.

$\boldsymbol{z}^{(i)}$: the linear output of the layer i before activation given by $\boldsymbol{w}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{b}^{(i)}$ with input $\boldsymbol{x}^{(i)}$ and bias $\boldsymbol{b}^{(i)}$. This also expresses the convolutional operation following the circulant form of $\boldsymbol{w}^{(i)}$.

$\boldsymbol{\alpha}^{(i)}(\cdot)$: activation function after linearity in vector form. We use the unbold letter $\alpha^{(i)}$ to denote its component.

$|.|$: when applied to a matrix, the absolute value sign $|.|$ denotes the number of elements.

w, x, z, b: we use unbold letters with subscripts to denote the component at specified indices of the corresponding matrix in bold letters.

To reconstruct the input image, we adopt an iterative approach similar to [16]: starting from the label, we reconstruct the input to the last layer and repeat this procedure layer by layer, each time making use of the reconstructed input to the succeeding layer. We will treat the cases of a fully-connected layer and a convolutional layer separately. First we treat the forward and backward pass in training a neural network as imposing two linear constraints on the input to each layer.

Weight and gradient constraints: At a given layer i, the forward and backward propagation gives rise to the following equations:

$$\boldsymbol{w}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{b}^{(i)} = \boldsymbol{z}^{(i)}, \tag{1a}$$

$$\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L} \cdot \boldsymbol{x}^{(i)} = \nabla_{\boldsymbol{w}^{(i)}}\mathcal{L}. \tag{1b}$$

We note that this represents both the fully-connected and the convolutional cases, using the circulant representation for the weight $\boldsymbol{w}^{(i)}$ and the gradient $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}$. Both $\boldsymbol{z}^{(i)}$ and $\nabla_{\boldsymbol{w}^{(i)}}\mathcal{L}$ are written as vectors. From the reconstruction point of view, we treat $\boldsymbol{x}^{(i)}$ as the unknown and regard the above equations as weight and gradient constraints imposed on the unknown. The term $\boldsymbol{z}^{(i)}$ is computed from inverting the reconstruction from the subsequent layer $\boldsymbol{x}^{(i+1)}$ using inverse of the activation function:

$$\boldsymbol{z}^{(i)} = (\boldsymbol{\alpha}^{(i)})^{-1}(\boldsymbol{x}^{(i+1)}). \tag{2}$$

The term $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}$ can be computed by using the following relations deduced from backpropagation:

$$\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L} = \nabla_{\boldsymbol{x}^{(i+1)}}\mathcal{L} \cdot \nabla_{\boldsymbol{z}^{(i)}}\boldsymbol{\alpha}^{(i)}, \tag{3a}$$

$$\nabla_{\boldsymbol{x}^{(i)}}\mathcal{L} = \nabla_{\boldsymbol{x}^{(i+1)}}\mathcal{L} \cdot \nabla_{\boldsymbol{z}^{(i)}}\boldsymbol{\alpha}^{(i)} \cdot \boldsymbol{w}^{(i)}. \tag{3b}$$

We notice that the circulant representation of the gradient $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}$ is determined by the circulant form of the weight $\boldsymbol{w}^{(i)}$ from backpropagating through the weight constraint (1a).

3.1 Fully connected layer

For a fully connected layer, the input can be solved uniquely in closed form. We summarise the solution in the following lemma. The case of non-zero bias is due to [1] and we show that it can be extended to the general case. Please see Supplementary A for the proof.

Lemma 3.1. *If a fully connected layer has non-zero bias $\boldsymbol{b}^{(i)} = (b_1^{(i)}, \ldots, b_m^{(i)}) \in \mathbb{R}^m$, then the input $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)}) \in \mathbb{R}^n$ is uniquely determined from the gradient constraint (1b). Suppose $\exists k, 1 \leq k \leq m$, such that $b_k^{(i)} \neq 0$ and $\frac{\partial \mathcal{L}}{\partial b_k^{(i)}} \neq 0$. Then $\boldsymbol{x}^{(i)}$ is given by:*

$$x_l^{(i)} = \frac{\partial \mathcal{L}}{\partial w_{kl}^{(i)}} \left(\frac{\partial \mathcal{L}}{\partial b_k^{(i)}} \right)^{-1}, \quad 1 \leq l \leq n. \tag{4}$$

More generally, assuming both $\frac{\partial \mathcal{L}}{\partial x_k^{(i+1)}}$ and $\frac{\partial \alpha_k^{(i)}}{\partial z_k^{(i)}}$ are nonzero, we have

$$x_l^{(i)} = \frac{\partial \mathcal{L}}{\partial w_{kl}^{(i)}} \left(\frac{\partial \mathcal{L}}{\partial x_k^{(i+1)}} \right)^{-1} \left(\frac{\partial \alpha_k^{(i)}}{\partial z_k^{(i)}} \right)^{-1}. \tag{5}$$

Remark 3.2. The reason that we can solve for $\boldsymbol{x}$ in closed form described above is essentially because there is no weight sharing in a fully-connected layer. This implies that the circulant form of $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}$ consists of blocks of diagonal matrices with the same element along the diagonal in each block, which allows (1b) to be solved exactly.

3.2 Convolutional layer

For a convolutional layer, we can no longer uniquely determine the input in general because of weight sharing. We first build on the work in [16] and formulate a linear system by combining the weight and gradient constraints from (1a) and (1b) into a single linear system of the input $\boldsymbol{x}$. Define $\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}$ to denote the following block matrices

$$\boldsymbol{u}^{(i)} := \begin{bmatrix} \boldsymbol{w}^{(i)} \\ \nabla_{\boldsymbol{z}^{(i)}}\mathcal{L} \end{bmatrix}, \boldsymbol{v}^{(i)} := \begin{bmatrix} (\boldsymbol{\alpha}^{(i)})^{-1}(\boldsymbol{x}^{(i+1)}) \\ \nabla_{\boldsymbol{w}^{(i)}}\mathcal{L} \end{bmatrix}. \tag{6}$$

Here both the term $(\boldsymbol{\alpha}^{(i)})^{-1}(\boldsymbol{x}^{(i+1)})$ and the term $\nabla_{\boldsymbol{w}^{(i)}}\mathcal{L}$ are written as vectors. The term $\nabla_{\boldsymbol{w}^{(i)}}\mathcal{L}$ has the same dimension as $|\boldsymbol{w}^{(i)}|$, i.e. the number of elements of the weight in its non-circulant form as a 4-dimensional array.

Notice that we can absorb the bias term into the product $\boldsymbol{w}^{(i)}\boldsymbol{x}^{(i)}$ by replacing the the weight matrix with its augmentation by $\boldsymbol{b}^{(i)}$ and $\boldsymbol{x}^{(i)}$ by its augmentation by one. Adopting this notation, (1a) and (1b) can be written as:

$$\boldsymbol{u}^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{v}^{(i)} = 0. \tag{7}$$

Recall that under our assumptions and (2), we can get $\boldsymbol{z}^{(i)}$ that defines $\boldsymbol{v}^{(i)}$ by inverting the solution to the reconstruction problem for the following layer and other coefficients in $\boldsymbol{u}^{(i)}$ and $\boldsymbol{v}^{(i)}$ are given from training. Based on [16], we view (7) as a linear system for the input $\boldsymbol{x}^{(i)}$ to the current layer i. In general, $\boldsymbol{u}^{(i)}$ may not be square and it can be rank-deficient, which means we can only get an estimated solution using the pseudo inverse of $\boldsymbol{u}^{(i)}$. Proceeding layer-by-layer in this manner, we can obtain a reconstruction of the input to the network. This is the R-GAP approach introduced in [16]. We think there are several sources of error in the solution using this approach alone:

1. The linear system (7) is defined using estimated value of $\boldsymbol{z}$ by inverting the solution from the succeeding layer. So the error from reconstructing the input to the succeeding layer will carry over to the current layer.
2. Pseudo inverse to a linear system is not unique in general and a minimum norm solution might not be the best one in our reconstruction problem.
3. Bad conditioning of $\boldsymbol{u}^{(i)}$ can contribute to the error of the estimated solution.

To tackle these issues, we propose a correction procedure based on the idea of gradient matching.

Correcting the approximated solution: From the theory of linear systems, we know that we can get an approximated solution to a linear system such as (7) by solving a least square problem:

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} ||\boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)}||^2. \tag{8}$$

The solution is not unique without requiring the norm of the solution to be minimal: the sum of a given solution with another vector $\boldsymbol{x_0}$ such that $\boldsymbol{u}^{(i)}\boldsymbol{x_0} = 0$ is still a solution. On the other hand, the solution with the minimum norm may not be the most accurate one for our reconstruction problem. Assuming $\boldsymbol{x}_{LS}$ is the minimum norm solution to the least square problem (8) obtained by using the Singular Value Decomposition (e.g. by Theorem 5.5.1 in [9]), we propose to correct it using the idea of gradient matching from [17]. More precisely, after we have obtained a solution $\boldsymbol{x}_{LS}^{(i)}$ at a layer, we formulate and solve the following optimisation problem:

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\, \mathcal{D}\left[\nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}, \nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x}_{true};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}\right], \quad \text{subject to} \quad \boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)} = 0. \tag{9}$$

where $\boldsymbol{x}_{true}$ is the target image, $\boldsymbol{w}^*$ are given weights and $\mathcal{D}[\cdot,\cdot]$ is a chosen distance function. Instead of taking $\mathcal{D}[\cdot,\cdot]$ to be the $L2$-norm as in [17], we adopt the cosine distance function proposed in [3] because we believe it is less sensitive to the stage of training. More precisely, we define $\mathcal{D}[\cdot,\cdot]$ to be

$$\mathcal{D}[\boldsymbol{x}_1,\boldsymbol{x}_2] := 1 - \frac{\langle \boldsymbol{x}_1,\boldsymbol{x}_2\rangle}{||\boldsymbol{x}_1||\cdot||\boldsymbol{x}_2||}, \tag{10}$$

for n-dimensional vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, where $\langle\cdot,\cdot\rangle$ and $||\cdot||$ are the Euclidean inner product and norm respectively.

In numerical experiments, we find it helpful in terms of reconstruction quality to add total variation to the objective function (9). The optimisation problem described by (9) will now become:

Algorithm 1 Hybrid method.

Input: Number of layers d of the network; True label $\boldsymbol{y}$ of the target image $\boldsymbol{x}_{true}$; Initial weights $\boldsymbol{w}^*$; Gradients $\nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}$ at each layer $i, 0 \leq i \leq d-1$; Number of iterations $N^{(i)}$ at each layer i.
Initialise $\overline{\boldsymbol{x}^{(d)}} = \boldsymbol{y}$.
for $i = d-1$ **to** 0 {iterate backward from the last layer of the network} **do**
  Compute the gradient $\nabla_{\boldsymbol{x}^{(i+1)}}\mathcal{L}(\boldsymbol{x}_{true};\boldsymbol{w}^*)\big|_{\boldsymbol{x}^{(i+1)}=\overline{\boldsymbol{x}^{(i+1)}}}$ using (3b) and $\overline{\boldsymbol{x}^{(i+1)}}$.
  Compute $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}(\boldsymbol{x}_{true};\boldsymbol{w}^*)\big|_{\boldsymbol{z}^{(i)}=(\boldsymbol{\alpha}^{(i)})^{-1}(\overline{\boldsymbol{x}^{(i+1)}})}$ from $\nabla_{\boldsymbol{x}^{(i+1)}}\mathcal{L}$ using (3a).
  if the current layer is fully connected **then**
    solve for $\overline{\boldsymbol{x}^{(i)}}$ using (5).
  else if the current layer is convolutional **then**
    Define $\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}$ from (6) using $(\boldsymbol{\alpha}^{(i)})^{-1}(\overline{\boldsymbol{x}^{(i+1)}})$ and gradients of $\mathcal{L}$ computed above.
    Get an estimate $\boldsymbol{x}_{LS}^{(i)}$ of the input to layer i by solving the linear system $\boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)} = 0$.
    Get a corrected estimate $\overline{\boldsymbol{x}^{(i)}}$ based on $\boldsymbol{x}_{LS}^{(i)}$ by solving the optimisation problem (11) with initialisation $\boldsymbol{x}_{LS}^{(i)}$ for $N^{(i)}$ iterations. {only compute and use gradients from the current layer to the last one}
  end if
end for
Output: Reconstruction $\overline{\boldsymbol{x}^{(0)}}$ of the target $\boldsymbol{x}$.

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \mu_1 \mathcal{D}\left[\nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}, \nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x}_{true};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}\right] + \mu_2 \mathrm{TV}(\boldsymbol{x}) \right\}, \quad \text{subject to} \quad \boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)} = 0. \tag{11}$$

where $\mu_1, \mu_2 \in \mathbb{R}$ are some given weights. Performing the correction described in (11) at each convolutional layer, we have a hybrid method to reconstruct the input presented in Algorithm 1. We observe that if we turn the hard constraint in (11) into a soft one, the algorithm will converge more quickly. More details are provided in section B.

A security measure: In light of the hybrid framework given for a convolutional layer, the problem of reconstructing a training image can be viewed as consisting of two parts : **i.** An iterative procedure starting from the output of the network ; **ii.** at each layer, solving a linear system with corrections using gradient matching when the layer is convolutional. Based on this insight, we define a metric that measures the efficacy of the hybrid method given by Algorithm 1, which depends partially on the architecture of the target model.

Definition 3.3. Suppose the model $\mathcal{M}$ has d convolutional layers indexed by $1, \ldots, d$, followed by a fully-connected layer. We define the following metric:

$$c(\mathcal{M}) := \sum_{i=1}^{d} \frac{d-(i-1)}{d} \cdot \left(\operatorname{rank}(\boldsymbol{u}^{(i)}) - n_i\right), \tag{12}$$

where $\boldsymbol{u}^{(i)}$ is defined in (6) and n_i is the dimension of the input for the i-th layer as a vector.

Because $\operatorname{rank}(\boldsymbol{u}^{(i)}) \leq n_i$ for each convolutional layer, $c(\mathcal{M})$ will be non-positive. The larger the value of the metric is, the less secure the model tends to be and the more likely it is to create better reconstructions. The metric is better interpreted as an estimate of the security of the model against the hybrid method. Our experiments have shown that it is possible to fully reconstruct the input to a model $\mathcal{M}$ using our method when $c(\mathcal{M}) = 0$. For more details on the thinking behind Definition 3.3, please refer to Section C in the Supplementary.

Recommendations on architectural design: Based on the proposed metric and analysis, we can see that a network tends to be less secure against our method if it has wider convolutional layers that greatly increase the dimensions of the feature spaces at the beginning of the network and only shrinks the feature spaces towards the final layer, compared to other

designs. Overall, we would recommend designing a model with a small value of $c(\mathcal{M})$ if defendability against gradient-leakage attacks is the main concern.

4 Experiments

We demonstrate the performance of our hybrid method and how our proposed index $c(\mathcal{M})$ in (12) can be an indicator of model security in practice. We consider a series of shallow architectures performing classification on CIFAR-10 provided by [8] and [12]. These architectures consist of two, three and four-layer convolutional networks. For simplicity, we assume the convolutional layer to be bias-free and the fully-connected layer to have non-zero bias. However, these assumptions on bias are not necessary for our method to work. In order to illustrate typical architecture designs for the network, we consider several variations of the network, each representing a different case of changes in dimensions of intermediate feature spaces. For each network, weights are initialised randomly from a uniform distribution.

Table 1: Model architecture for all variants of the models, rank deficiency 'rd' for each layer and values of metric $c(\mathcal{M})$. The numbers in columns 'layer x' refer to kernel width, channels, strides and padding in order. The numbers in the column 'fully connected' refer to the input dimension of that fully-connected layer, whereas the output dimension is always 10. The values of $c(\mathcal{M})$ are the same between two images used in the experiment.

	Layer 1	Layer 2	Layer 3	Fully Connected	rd1	rd2	rd3	$c(\mathcal{M})$
CNN2 Variant 1	3,6,1,0	/	/	5400	0	/	/	0
CNN2 Variant 2	4,6,2,0	/	/	1350	-1470	/	/	-1470
CNN3 Variant 1	3,6,1,0	4,3,2,0	/	588	0	-4533	/	-2266
CNN3 Variant 2	4,6,2,0	3,3,2,0	/	147	-1470	-1050	/	-1995
CNN3 Variant 3	3,6,1,0	3,9,1,0	/	7056	0	0	/	0
CNN3 Variant 4	3,1,1,0	3,6,1,0	/	4704	-2146	0	/	-2146
CNN4 Variant 1	3,6,1,0	4,5,2,0	4,3,1,0	363	0	-3965	-386	-2772
CNN4 Variant 2	5,16,1,0	5,6,2,0	5,32,1,2	4608	0	-9316	0	-6211

Figure 1: Comparisons of reconstructions among all approaches for CNN2 and CNN4. Two examples are presented for each architecture. Observe that DLG is unable to reconstruct in all variants in CNN2 and in CNN4 Variant 1, but is able to produce good reconstruction with artefact for CNN4 Variant 2. CosineTV is more stable than DLG while R-GAP performs even more consistently. Our hybrid method improves the results from R-GAP and visually reduces its checkerboard effect and produces results with better overall quality.

The target image will go through one forward and backward pass to generate the gradients. For each convolutional layer, we assume Tanh activation and for each fully-connected layer we assume identity activation. We adopt the unconstrained strategy (16) from Section B in Algorithm 1, using ADAM optimiser from [7] with default learning rate of 0.001. The architecture of the models in the experiments are shown in Table 1, together with the values

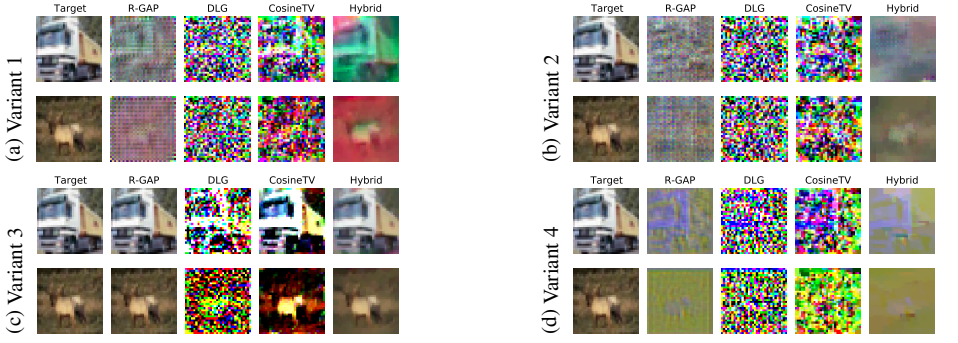


Figure 2: Comparisons of reconstructions among all approaches for CNN3. Two examples are presented for each architecture. Similar to Figure 1, we notice that DLG and CosineTV show similar performance although CosineTV provides improvement overall. Our hybrid method is consistent with R-GAP but produces smoother results. Also notice that all methods generally produce best results in Figure 2c and worst in Figure 2d, which is mostly consistent with the value of the metric $c(\mathcal{M})$ given in Table 1.

of the metric $c(\mathcal{M})$. For each variant of the model, we compare our method with that of DLG from [17], R-GAP from [16], and [3] (which we name 'CosineTV' for short). For details of the number of iterations for all the methods, and other hyperparameter settings in the experiment, please refer to Section D in the Supplementary.

We present sample outputs in Figure 1, Figure 2 with their MSE and PSNR scores in Table 3 in the Supplementary. Comparing the reconstruction qualities of results in Figure 1, Figure 2 with the values of $c(\mathcal{M})$ in Table 1, we notice that it is more likely for all methods to have reconstructions with better visual quality on architectures with a bigger value of $c(\mathcal{M})$, despite that the relation is not strictly monotonic.

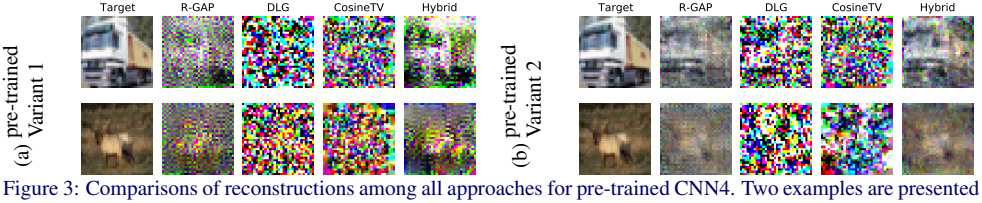


Figure 3: Comparisons of reconstructions among all approaches for pre-trained CNN4. Two examples are presented for each architecture. Compared to Figure 1c and 1d, we notice that reconstructions from all methods have worsened, and DLG and CosineTV are no longer producing visually recognisable results. On the other hand, R-GAP and our hybrid method are still showing more recognisable results.

It is worth noticing that in CNN2 Variant 1 and CNN3 Variant 3, all of their convolutional layers have positive index given by the summand $\operatorname{rank}(\boldsymbol{u}^{(i)}) - n_i$ in (12), which explains the most information leakage about the training image compared to other variants. On the other hand, we noticed that the value of $\operatorname{rank}(\boldsymbol{u}^{(i)}) - n_i$ is negative in the first layer in CNN3 Variant 4. Although it becomes positive in layer 2, it seems that this cannot make up for the loss of information occurred in layer 1. We provide the value of $\operatorname{rank}(\boldsymbol{u}^{(i)}) - n_i$ at each layer in all the models in Table 1.

We also apply our algorithm against CNN4 that are pre-trained, see Figure 3. The quality of reconstructions has degraded for all methods, although our method is still showing more recognisable results. Details of these experiments are provided in Section F. We provide more reconstructed examples from CIFAR-10 in Section H. All results are consistent with

our analysis.

In summary, we have shown that reconstructions from our hybrid method improve over those from R-GAP while minimising the instability that is inherent in DLG and CosineTV.

5 Conclusion

In this paper, we try to further our understanding of existing gradient-leakage attacks by developing a hybrid framework which combines solving a linear system at each layer accompanied by gradient matching for corrections. Our framework provides a connected viewpoint between the existing analytic and optimisation-based methods. It also partially attributes the vulnerability of a deep network against gradient-leakage attacks to its architecture. The metric we propose can provide us with a guideline in designing a deep network more securely.

Limitations and Future Work There are a few important questions that we haven't addressed in this work, e.g. how does one apply our framework when the batch size is greater than one? We provide detailed discussions on limitations and future work in Section G. The code for this work will be posted on https://github.com/CangxiongChen/training-data-leakage.

Acknowledgements We would like to acknowledge funding support from the EPSRC CAMERA Research Centre (EP/M023281/1 and EP/T022523/1), the Innovate UK SmartRoto project (04642027) and the Royal Society.

References

[1] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving Deep Learning via Additively Homomorphic Encryption. IEEE Transactions on Information Forensics and Security, 13(5):1333–1345, 2017.

[2] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, Chang Liu, Chee Seng Chan, and Qiang Yang. Rethinking Privacy Preserving Deep Learning: How to Evaluate and Thwart Privacy Attacks. In Qiang Yang, Lixin Fan, and Han Yu, editors, Federated Learning: Privacy and Incentive, pages 32–50. Springer International Publishing, Cham, 2020. ISBN 978-3-030-63076-8. doi: 10.1007/978-3-030-63076-8_3. URL https://doi.org/10.1007/978-3-030-63076-8_3.

[3] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients – how easy is it to break privacy in federated learning? In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[4] G.H. Golub and C.F. Van Loan. Matrix Computations. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. ISBN 9781421408590. URL https://books.google.co.uk/books?id=5U-l8U3P-VUC.

[5] G.H. Golub and C.F. Van Loan. Circulant Systems. In Matrix Computations, Johns Hopkins Studies in the Mathematical Sciences, chapter 4.8.2, pages 220–222. Johns Hopkins University Press, 2013. ISBN 9781421408590. URL https://books.google.co.uk/books?id=5U-l8U3P-VUC.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

[8] Alex Krizhevsky et al. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[9] Marucha Lalee, Jorge Nocedal, and Todd Plantenga. On the Implementation of an Algorithm for Large-Scale Equality Constrained Optimization. SIAM Journal on Optimization, 8(3):682–706, 1998. doi: 10.1137/S1052623493262993. URL https://doi.org/10.1137/S1052623493262993.

[10] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10122–10132, 2022. doi: 10.1109/CVPR52688.2022.00989.

[11] Jorge Nocedal and Stephen Wright. Numerical Optimization. Springer Science & Business Media, 2006.

[12] Valay Shah. CIFAR-10. https://peltarion.com/knowledge-center/documentation/terms/dataset-licenses/cifar-10, 2013. [accessed 27-Jan-2022].

[13] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for Simplicity: The All Convolutional Net. In ICLR (workshop track), 2015. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a.

[14] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through Gradients: Image Batch Recovery via Gradinversion. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 16332–16341, 2021. doi: 10.1109/CVPR46437.2021.01607.

[15] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved Deep Leakage from Gradients. In Advances in Neural Information Processing Systems (NeurIPS), 2019.

[16] Junyi Zhu and Matthew Blaschko. R-GAP: Recursive Gradient Attack on Privacy. In International Conference on Learning Representations - (ICLR), 2021.

[17] Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32, pages 14774–14784. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf.

Analysing Training-Data Leakage from Gradients through Linear Systems and Gradient Matching: Supplementary Material

A Proof of Lemma 3.1

Proof. We will prove the general case first. By the construction of a fully-connected layer, we can take the j-th column of the gradient constraint (1b) which gives:

$$(\frac{\partial \mathcal{L}}{\partial w_{1j}^{(i)}},...,\frac{\partial \mathcal{L}}{\partial w_{nj}^{(i)}})^T = x_j^{(i)}(\frac{\partial \mathcal{L}}{\partial z_1^{(i)}},...,\frac{\partial \mathcal{L}}{\partial z_n^{(i)}})^T. \tag{13}$$

This implies that if $\frac{\partial \mathcal{L}}{\partial z_k^{(i)}} \neq 0$ for some $k, 1 \leq k \leq n$, then $\boldsymbol{x}^{(i)}$ can be uniquely determined:

$$x_j^{(i)} = \frac{\partial \mathcal{L}}{\partial w_{kj}^{(i)}}(\frac{\partial \mathcal{L}}{\partial z_k^{(i)}})^{-1}. \tag{14}$$

In the special case when $b_k^{(i)} \neq 0$, we can see from the weight constraint (1a) that:

$$\frac{\partial \mathcal{L}}{\partial z_k^{(i)}} = \frac{\partial \mathcal{L}}{\partial b_k^{(i)}}, \tag{15}$$

which was observed in [1] and subsequently also in [2]. In general, since the activation functions are assumed to be piecewise invertible and piecewise differentiable, and since $\frac{\partial \mathcal{L}}{\partial x_k^{(i+1)}}$ is assumed to be nonzero, we can compute $x_j^{(i)}$ using (3), which gives (5). □

B Convergence of the algorithm

At a convolutional layer, we have formulated the optimisation problem where the linear system (7) defines a hard constraint, so that it is satisfied throughout the optimisation. Although this formulation makes it clear that the difference $\overline{\boldsymbol{x}^{(i)}} - \boldsymbol{x}_{LS}^{(i)}$ is inside the null space of $\boldsymbol{u}^{(i)}$, we find that in practice it does not always lead to the convergence of the optimisation within a reasonable amount of run time when we use trust-region methods such as [10] and [9] to solve the constrained optimisation problem given in (11). On the other hand, allowing constraint violation by defining (7) as a soft constraint can often speed up the convergence. More precisely, instead of the problem (11), we consider an unconstrained optimisation problem:

$$\underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \mu_1 \mathcal{D}\left[\nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}, \nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x}_{true};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}\right] + \mu_2 \mathrm{TV}(\boldsymbol{x}) + \mu_3 ||\boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)}||^2 \right\}, \tag{16}$$

where $\mu_1, \mu_2, \mu_3 \in \mathbb{R}$ are some given weights. We have observed in our experiments that with using the unconstrained optimisation, Algorithm 1 will be able to converge much faster whereas it can take much longer for the original version to converge to the same result.

C Justification for the security measure

In light of the hybrid framework given for a convolutional layer, the problem of reconstructing a training image can be viewed as consisting of two parts:

1. An iterative procedure starting from the output of the network.
2. At each layer, we first solve a linear system defined by the forward and backward pass of the target image-label pair, then we correct the solution by gradient matching with the target image if the layer is convolutional. If the layer is fully connected, the correction is not necessary.

For a fully connected layer, we have shown in Lemma 3.1 that we can always reconstruct the input in full. This can be regarded as no level of security and we omit it from our definition of the metric. For a convolutional layer, since the basic criterion for measuring the solubility of a linear system is given by comparing the rank of the coefficient matrix with the number of unknowns, and that the corrected solution still satisfies the linear system, we consider $\operatorname{rank}(\boldsymbol{u}) - |\boldsymbol{x}|$ as an index to measure the efficacy of the hybrid method. The larger this number is, the less rank-deficient the linear system (7) is and so more likely to have a full reconstruction for this layer. We also notice that the position where the rank-deficiency happens also matters. The closer it is to the first layer, the bigger impact it has on the reconstruction. This is consistent with our intuition that if the representation of the input data loses information at the first layer, it will be unlikely to substitute that loss in latter layers. To accommodate for this effect, we discount the index $\operatorname{rank}(\boldsymbol{u}) - |\boldsymbol{x}|$ by the position of the layer in the network.

D Details of the implementation

For the re-implementations of DLG and CosineTV, we follow the number of iterations used by the authors of the corresponding work, i.e. 300 for DLG and 4800 for CosineTV. In the implementation for our hybrid method, we adopt the following setting of hyperparameters:

Table 2: Hyperparameters for our implementation of the hybrid method. The weights are placed according to the objective function given in (16).

	layer 1	layer 2	other layers
Iterations	10000	8000	1000
μ_1	1.0	1.0	10.0
μ_2	1.0	1.0	0.1
μ_3	0.05	0.1	1.0

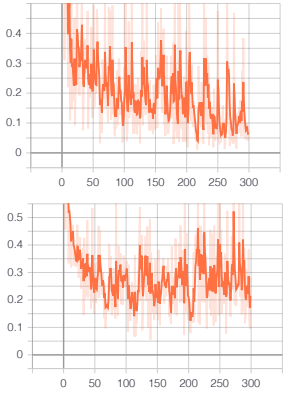
E Evaluations of the experiments

Table 3: MSE and PSNR (as in the first and second component in the pair) of the reconstructions in each variant, averaged over the two images; they are consistent with the visual qualities in Figures 1, 2 and 3.

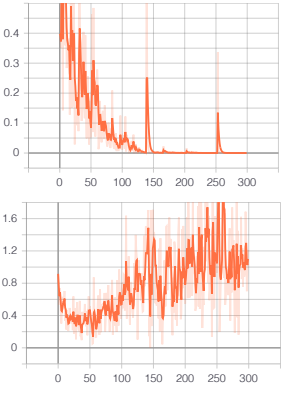
	R-GAP	DLG	CosineTV	Hybrid
CNN2 Variant 1	0.0000, 197.00	2.0181, 45.08	0.2290, 54.54	0.0008, 79.82
CNN2 Variant 2	0.0346, 62.75	2.15E+08, -9.32	0.3257, 53.01	0.0051, 71.69
CNN3 Variant 1	0.0531, 60.90	0.9279, 48.46	0.4086, 52.03	0.0478, 61.48
CNN3 Variant 2	0.0518, 60.99	0.9900, 48.17	0.4739, 51.37	0.0322, 63.78
CNN3 Variant 3	0.0000, 181.26	3.75E+17, -59.49	0.2302, 54.51	0.0020, 75.79
CNN3 Variant 4	0.0429, 61.83	5.11E+13, -29.17	0.5082, 51.07	0.0417, 61.96
CNN4 Variant 1	0.0547, 60.81	0.8585, 48.79	0.4255, 51.86	0.0610, 60.28
CNN4 Variant 2	0.0406, 62.05	0.0951, 58.35	0.2177, 54.82	0.0139, 67.72
CNN4 Variant 1 (pre-trained)	0.2174, 54.90	7.58E+08, -3.07	0.8048, 49.25	0.3449, 53.33
CNN4 Variant 2 (pre-trained)	0.0341, 62.81	406.2, 26.98	0.7353, 49.71	0.0288, 63.63

F Details on the pre-training of CNN4

We pre-train CNN4 on images from only two classes from CIFAR-10, namely 'automobiles' and 'birds'. The target images used for reconstructions have not been seen by the models during pre-training. Both variants of CNN4 have been trained on 10000 images and tested on 1000 images, with batch size 64. We have used ADAM optimiser with initial learning rate of 0.001 and the models were trained for 300 epochs. Figure 4 shows losses during training and testing. We expect that reconstructions from all methods to deteriorate, because



(a) CNN4 Variant 1. Top: training loss; Bottom: testing loss



(b) CNN4 Variant 2. Top: training loss; Bottom: testing loss

Figure 4: Plots of the losses during training and testing. Notice that variant 2 has noticeable overfitting.

a pre-trained model is likely to produce gradients with smaller magnitude and variance when it is retrained on an unseen image compared to an untrained model. Results shown in Figure 3 seems to have confirmed this guess. However, there might be exceptions when the model is overfitted and then retrained on an unseen image. We notice that across untrained and trained cases and for both images, the metric $c(\mathcal{M})$ and the layerwise rank deficiency $\text{rank}(\boldsymbol{u}^{(i)}) - n_i$ have the same value in each variant respectively.

G Limitations and future work

Batch size In this work, we are only considering the problem to reconstruct one single training image. It is natural to wonder how to apply our framework to the case when we are given the gradients from a batch of training images. When a batch of images are used, the gradients used to define the gradient constraint (1b) will be an average of those from each image in the batch. It is unclear how to decompose the gradient constraint without introducing further assumptions. Without those assumptions, a straightforward application of our hybrid framework in this case will give a reconstruction that is difficult to interpret. Although there exist works that try to tackle this problem (for example [14] and [3]), we have not found any approach that offers theoretical insight nor guarantees to this problem. For the interest of obtaining theoretical guarantees of the reconstructions and the security of the architecture, we would leave this as future work.

Scope of the security measure We think that the security measure (12) captures the solubility of the linear system (7) by computing its rank deficiency which depends on the input and output dimensions of the layer and the values of weights and gradients. The security measure does not take into account other factors such as the condition number of the the system (7), which although affects the stability of the solution rather than solubility, can also affect the quality of the reconstructions. It would be interesting to extend the security measure to include the condition number of the system to give a more accurate measure of the security of the architecture under our hybrid framework.

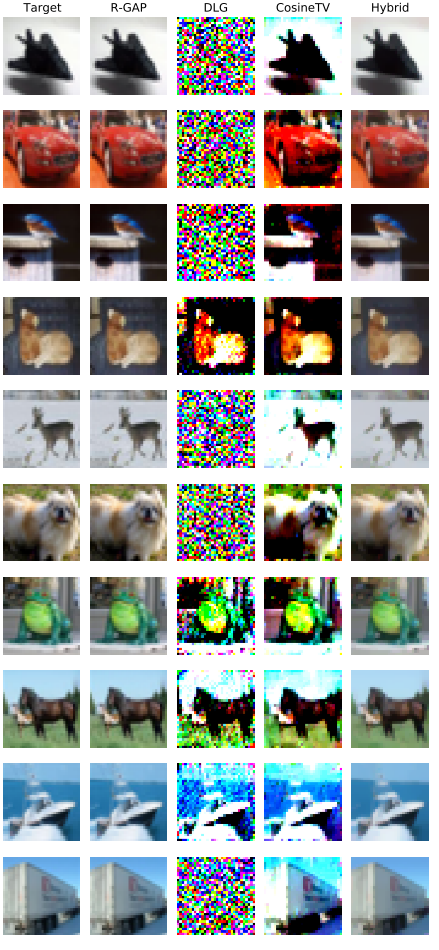
Activation functions In all convolutional networks used in the experiment, we have assumed the activation function in a convolutional layer to be Tanh. We believe that our framework will also apply to other activations as long as they are smooth and invertible. We noticed that if we use LeakyReLU and solve the optimisation problem in (11) using constrained optimisation such as trust-region methods from [11] and [9], it will be difficult for it to converge to the correct optimum within a reasonable amount of running time. Although this is more of a limitation with the optimisation than our framework, we will be looking for strategies for optimisation that can better deal with non-smooth functions in the future.

Architectures of the target model Although we have not considered other popular architectures in image classifications such as Residual networks [6], we believe our framework can be adapted to these networks if we can define the corresponding linear system for a residual block. For example, we can define a similar linear system as (1a) by approximating the ResNet block defined in [6] using Taylor expansion for the non-linear terms given by activations inside the block. More details will be given in future work.

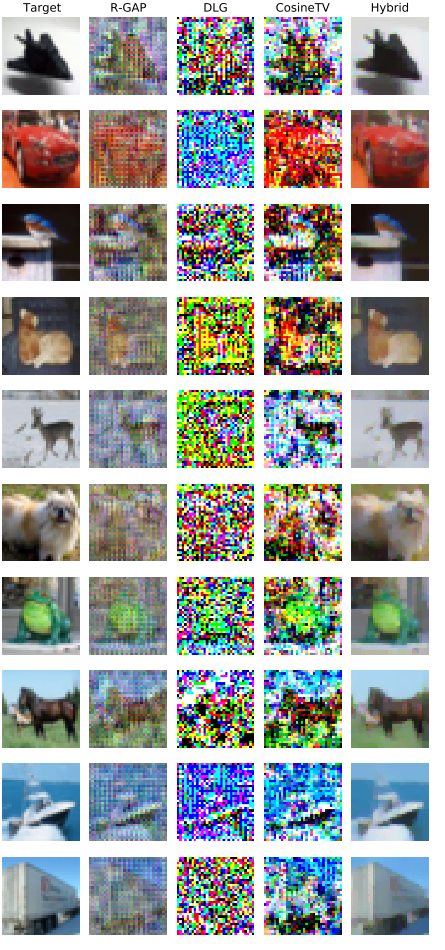
Reducing gradient leakage Another future avenue of work is investigating strategies to reduce gradient leakage with theoretical guarantee. A promising direction is considering training methods that provides Differential Privacy. One insight from our analysis leading to the metric $c(\mathcal{M})$ is that to reduce gradient leakage, we can add noise to the gradients so that the value of $c(\mathcal{M})$ can be reduced sufficiently.

H More examples of reconstructions

We provide more examples from CIFAR-10 for comparing all the methods discussed in the Experiment section. One image from each of the 10 classes is chosen.

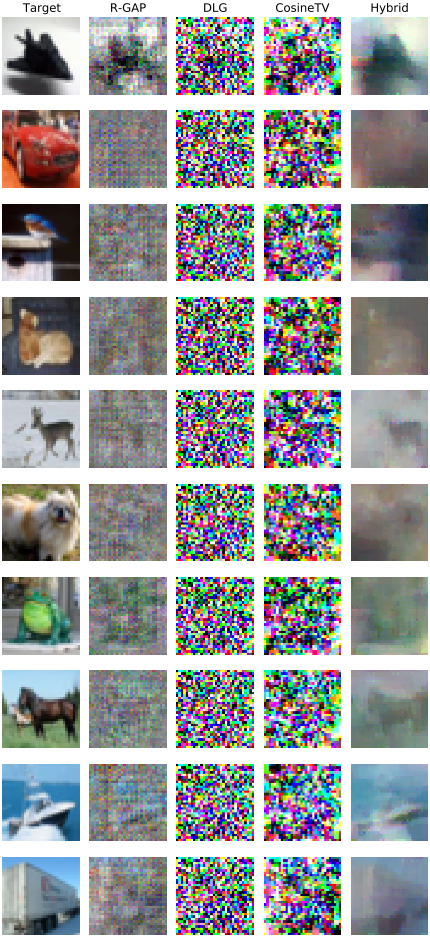


(a) CNN2 Variant 1

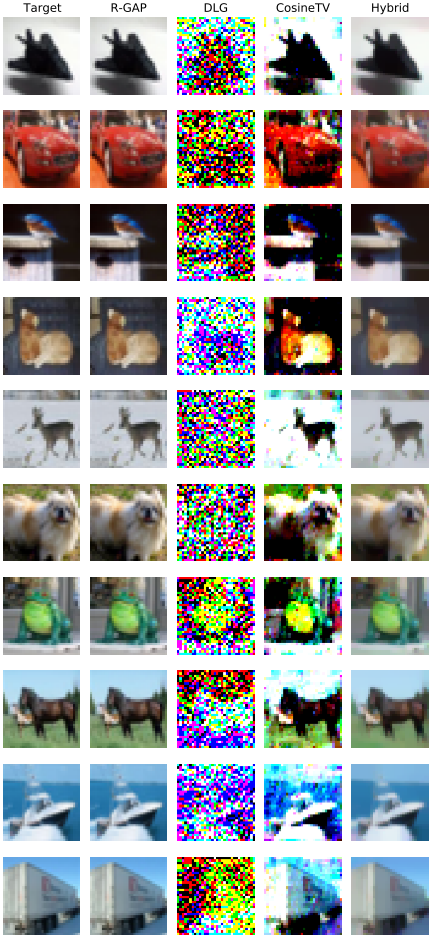


(b) CNN2 Variant 2

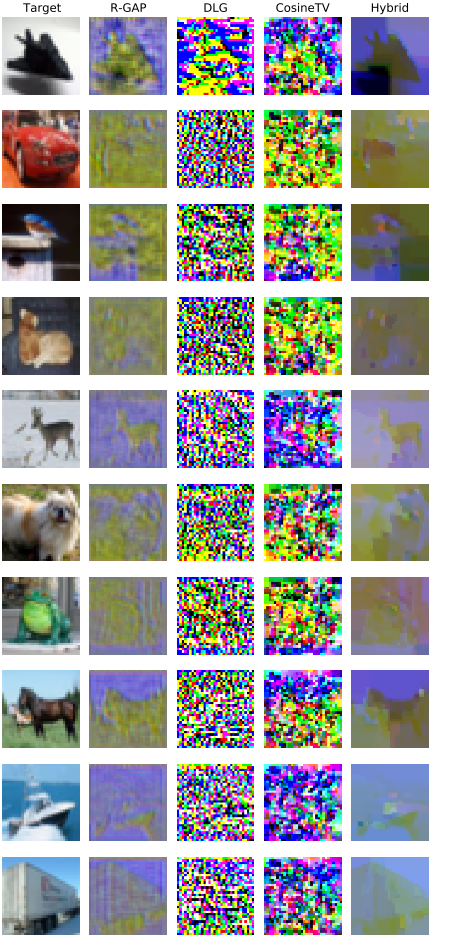
(a) CNN3 Variant 1



(b) CNN3 Variant 2

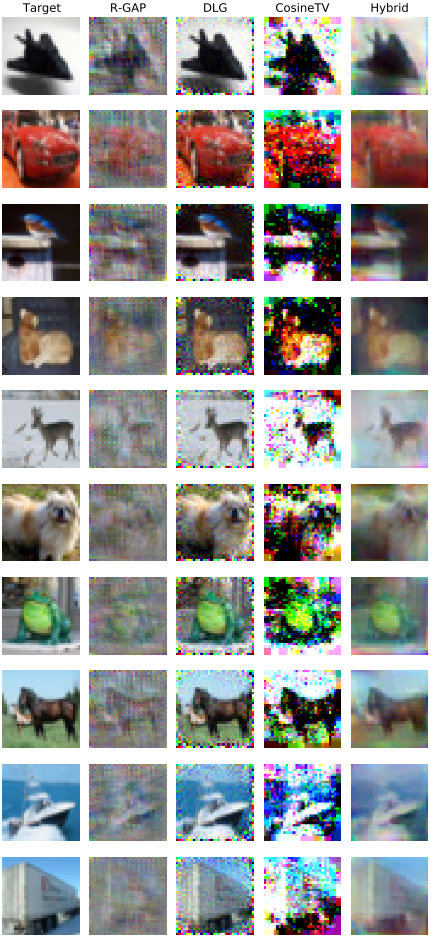


(a) CNN3 Variant 3



(b) CNN3 Variant 4

(a) CNN4 Variant 1



(b) CNN4 Variant 2