# 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:

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

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 [11] 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 $\text{rank}(\boldsymbol{u}) - |\boldsymbol{x}|$ 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 $\text{rank}(\boldsymbol{u}) - |\boldsymbol{x}|$ 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 |
| $\mu_1$ | 1.0 | 1.0 | 10.0 |
| $\mu_2$ | 1.0 | 1.0 | 0.1 |
| $\mu_3$ | 0.05 | 0.1 | 1.0 |

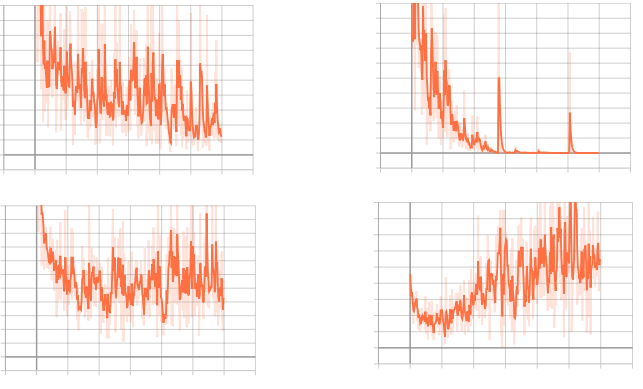
# E Evaluations of the experiments

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.

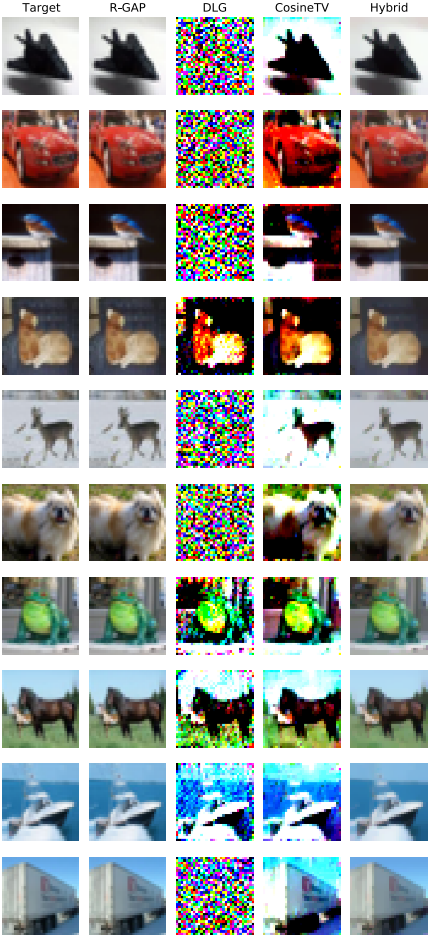
**Activation functions** In all convolutional networks used in the experiment, we have assumed the activation function in a convolutional layer to be Tanh. We believe that our framework will also apply to other activations as long as they are smooth and invertible. We noticed that if we use LeakyReLU 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.

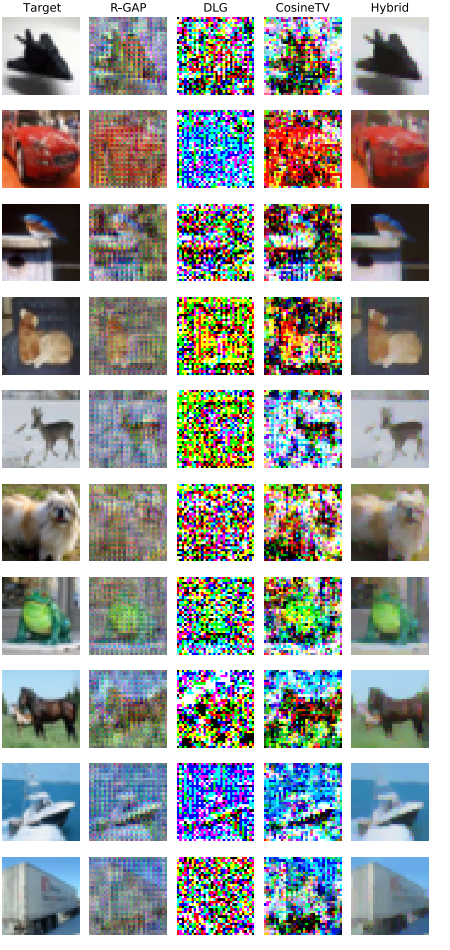
**Reducing gradient leakage** Another future avenue of work is investigating strategies to reduce gradient leakage with theoretical guarantee. A promising direction is considering training methods that provides Differential Privacy. One insight from our analysis leading to the metric $c(\mathcal{M})$ 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.

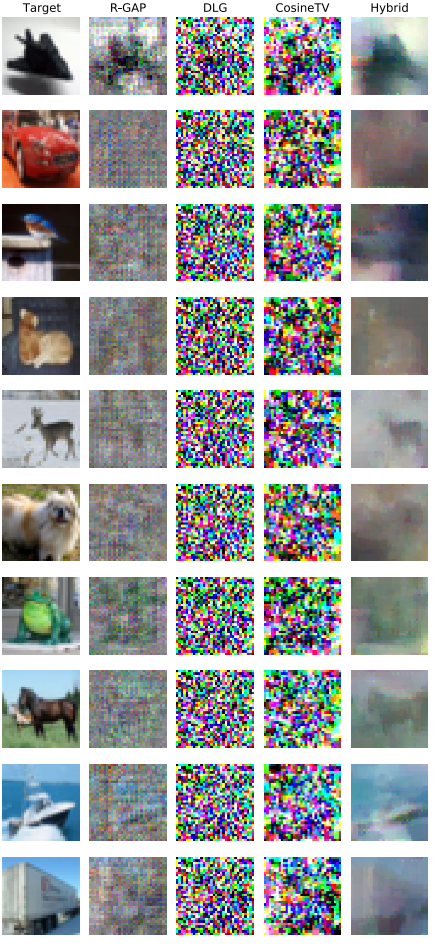



(a) CNN2 Variant 1

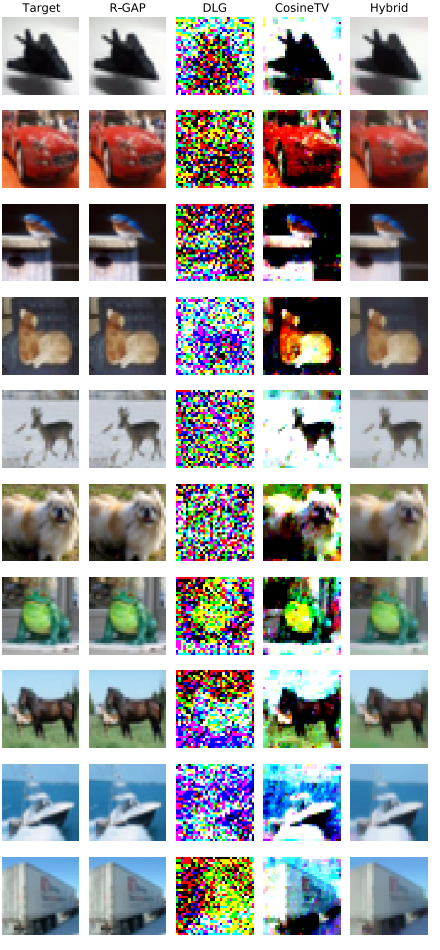



(b) CNN2 Variant 2

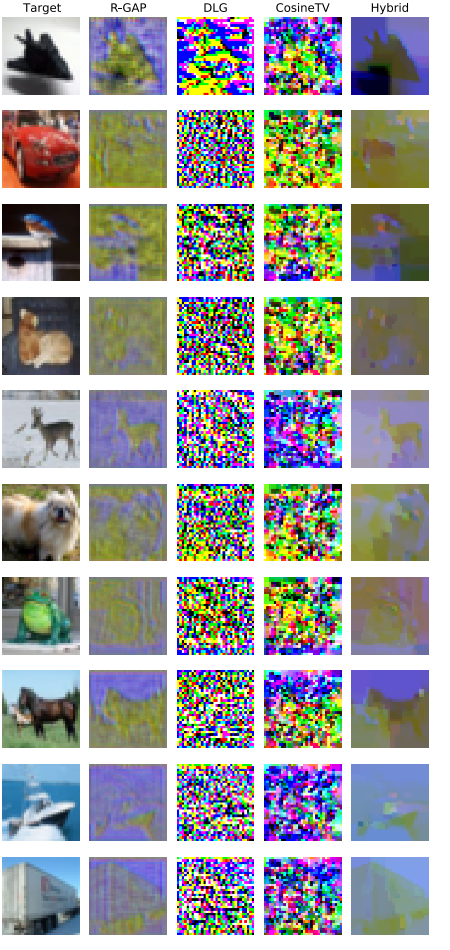
(a) CNN3 Variant 1




(b) CNN3 Variant 2

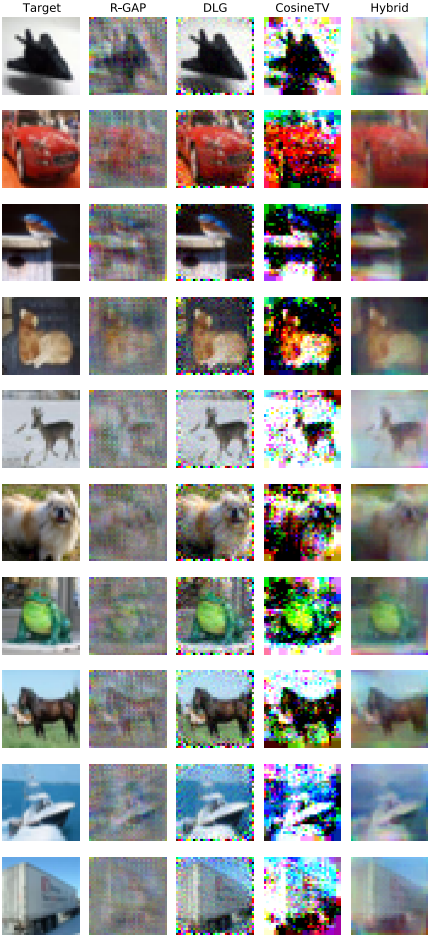



(a) CNN3 Variant 3




(b) CNN3 Variant 4

(a) CNN4 Variant 1




(b) CNN4 Variant 2