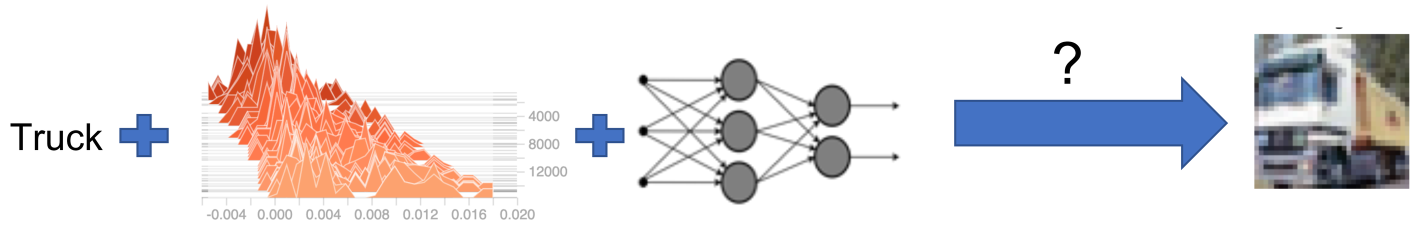
Analysing Training-Data Leakage from Gradients through Linear Systems and Gradient Matching

Cangxiong Chen and Neill D.F. Campbell

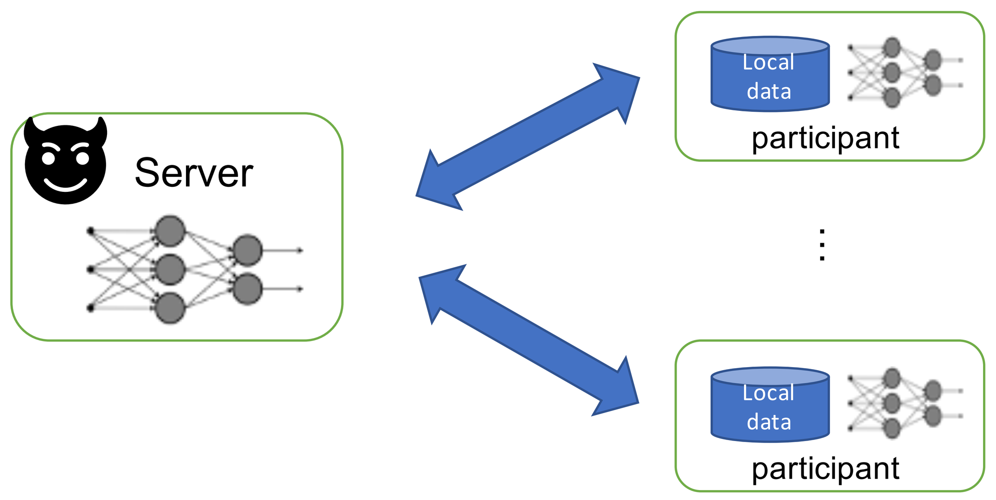
Introduction

- Given training gradients of the input image, its label, the model architecture, can we reconstruct the input image?



- Gradient-leakage-attack (GLA) methods such as DLG [1] and R-GAP [2] have demonstrated that this is possible for an image classification model.
- We can group existing GLA methods into optimisation-based or analytic ones.
- In this work, we provide a unified framework to understand existing GLA methods.

An important motivation is to understand how we can protect local training data from participants in federated learning, when there is a server-side adversary with access to the local training updates.



Notation

We provide a summary of notation used here. Superscript with parenthesis indicates the index of a layer in the network.

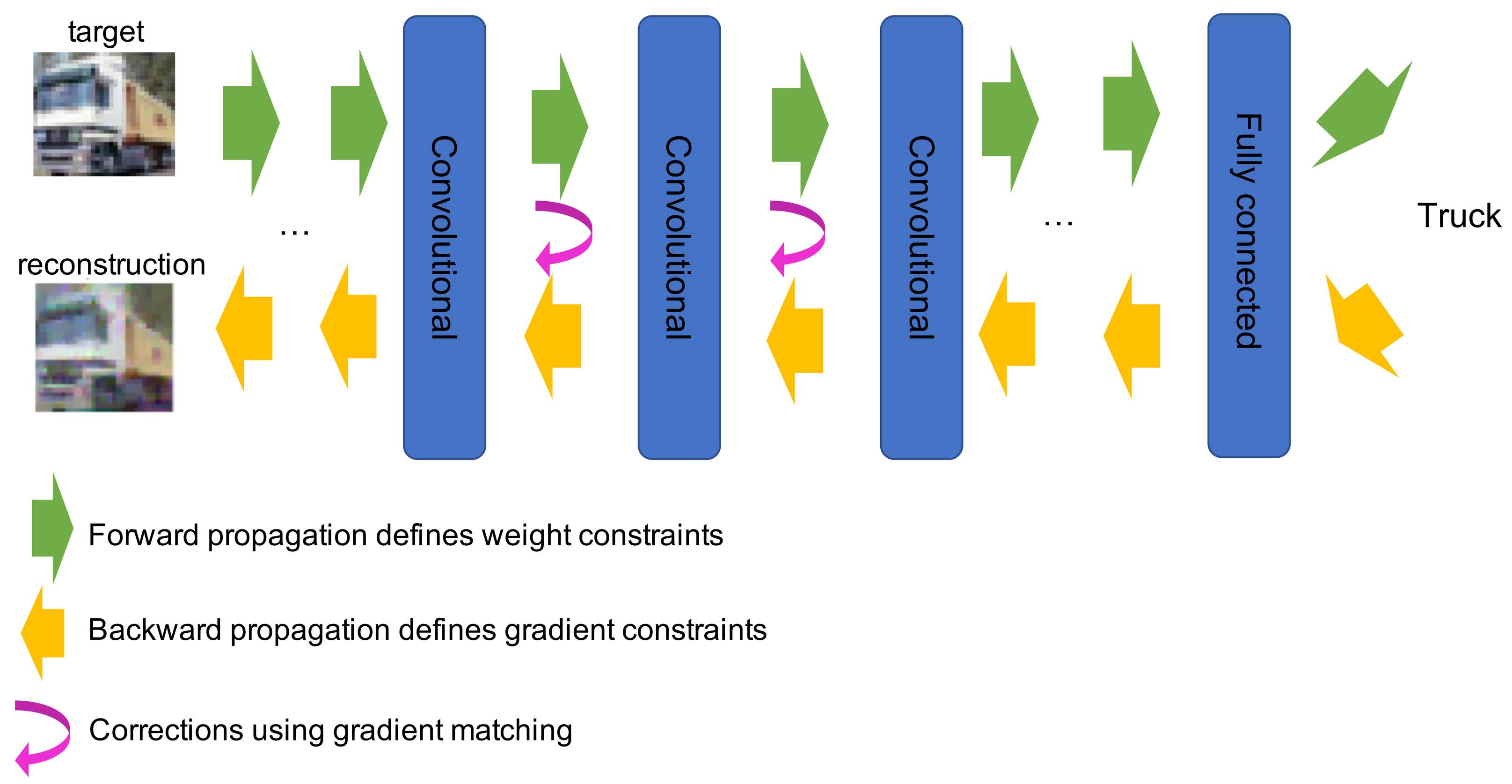
$\boldsymbol{w}^{(i)}$: weight from layer i of size m by n, where $0 \le i \le d$ and d is the total number of layers. For simplicity of notation, we omit i in m and n when possible, but readers should be aware that the size of the weight need not be the same for different layers. For a convolutional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

$\mathcal{L}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w})$: 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.

A Hybrid Framework for GLA



- We will focus on the case when the batch size in training is one.
- We adopt an iterative approach. The solution is approximate if the layer is convolutional, or is closed-form if it is fully connected.

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

$$\boldsymbol{x}_{LS}^{(i)} := \operatorname*{argmin}_{\boldsymbol{x}} ||\boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)}||^2.$$

$$\Downarrow$$

$$\overline{\boldsymbol{x}^{(i)}} := \operatorname*{argmin}_{\boldsymbol{x}} \left\{ \mu_1 \mathcal{D}\left[\nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}, \nabla_{\boldsymbol{w}}\mathcal{L}^{(i)}(\boldsymbol{x}_{true};\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*}\right] + \mu_2 \mathrm{TV}(\boldsymbol{x}) \right\},$$

subject to $\boldsymbol{u}^{(i)}\boldsymbol{x} - \boldsymbol{v}^{(i)} = 0$ and with initialisation $\boldsymbol{x}_{LS}^{(i)}$.

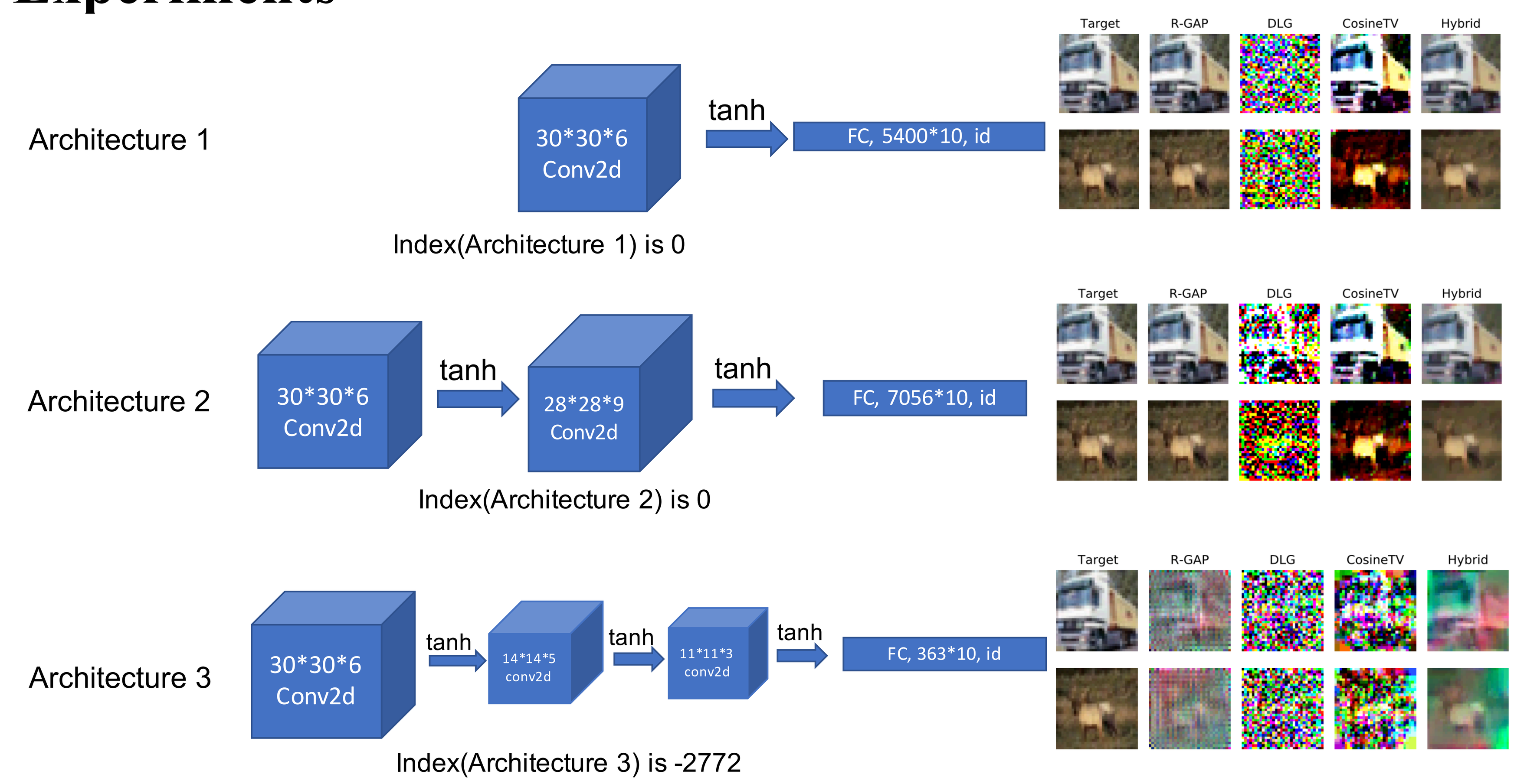
$$\mathcal{D}[\boldsymbol{x}_1,\boldsymbol{x}_2] := 1 - \frac{\langle \boldsymbol{x}_1,\boldsymbol{x}_2 \rangle}{||\boldsymbol{x}_1|| \cdot ||\boldsymbol{x}_2||}$$

At each convolutional layer, we solve a least square problem followed by corrections from gradient matching.

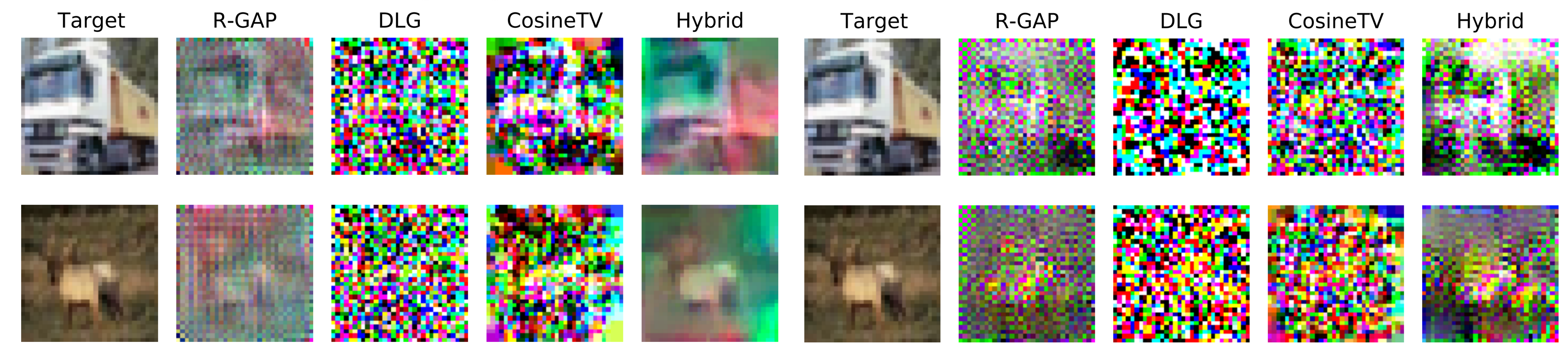
Algorithm 1 Hybrid method.

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 $i, 0 \le i \le d-1$; 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 $\overline{\boldsymbol{x}^{(i+1)}}$.
  Compute $\nabla_{\boldsymbol{z}^{(i)}}\mathcal{L}(\boldsymbol{x}_{true};\boldsymbol{w}^*)\big|_{\boldsymbol{z}^{(i)}=(\boldsymbol{\alpha}^{(i)})^{-1}(\overline{x^{(i+1)}})}$ from $\nabla_{\boldsymbol{x}^{(i+1)}}\mathcal{L}$.
  if the current layer is fully connected **then**
    solve for $\overline{\boldsymbol{x}^{(i)}}$ in close form.
  else if the current layer is convolutional **then**
    Define $\boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}$ 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 layerwise optimisation problem 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 $\boldsymbol{x}$.

Experiments



Reconstructions of training images for three architectures. Values of the metric are also shown for each architecture



Reconstructions for a 4-layer CNN. Left: when the network is untrained; Right: when the network is pre-trained

Definition. 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:

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

where n_i is the dimension of the input for the i-th layer as a vector.

Summary

- We advance our understanding of existing GLA by developing a unified framework which combines solving a linear system at each layer accompanied by gradient matching for corrections.
- The framework partially attributes the vulnerability of a deep network against GTA to its architecture. We provide a metric to quantify this vulnerability.
- For future work, we are interested in problems such as batch-image reconstruction and other architectures such as ResNets.

References

[1] Ligeng Zhu, Zhijian Liu, and Song Han. "Deep Leakage from Gradients". In: Advances in Neural Information Processing Systems. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 14774–14784.

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