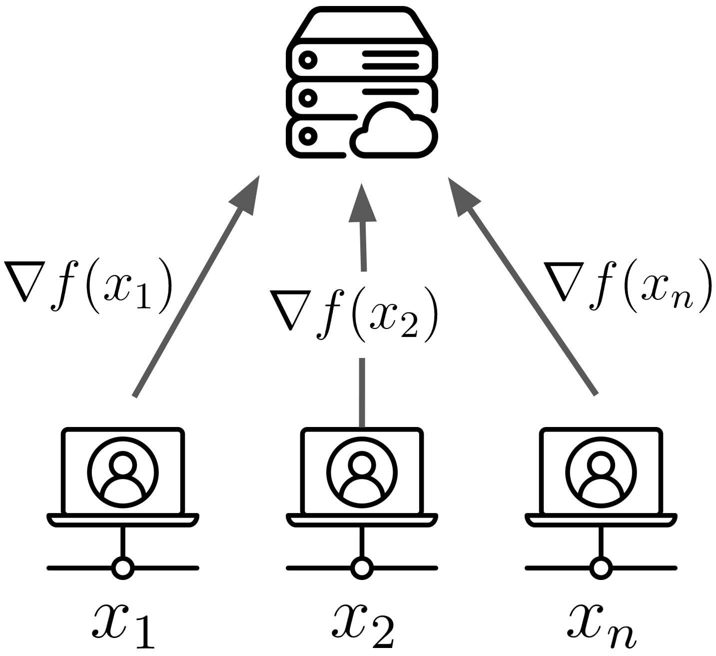
SPEAR: Exact Gradient Inversion of Batches in Federated Learning

Code: https://github.com/eth-sri/SPEAR



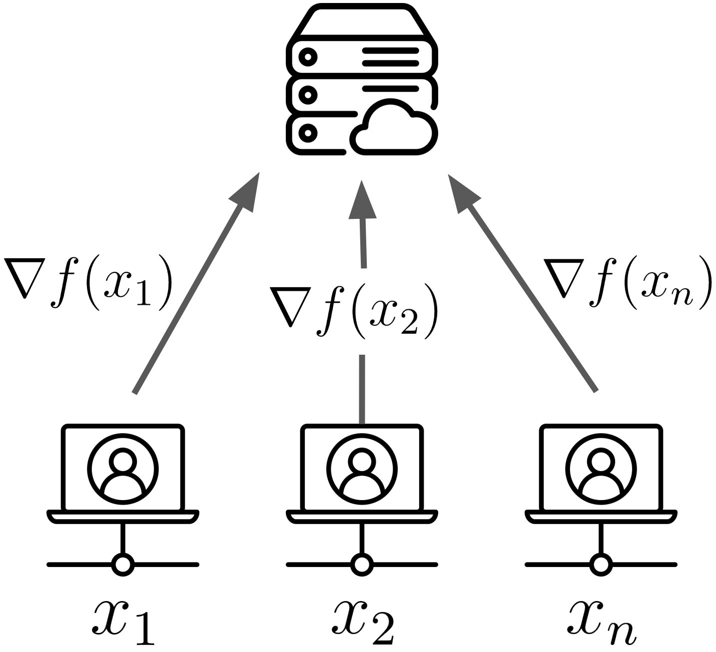

Dimitar I. Dimitrov
Maximilian Baader
Mark Müller
Martin Vechev

Federated Learning and Gradient Inversion



In **federated learning** a model is trained in a distributed fashion where clients only send **gradient updates** to the server in order to **preserve their data privacy**.

Federated Learning and Gradient Inversion



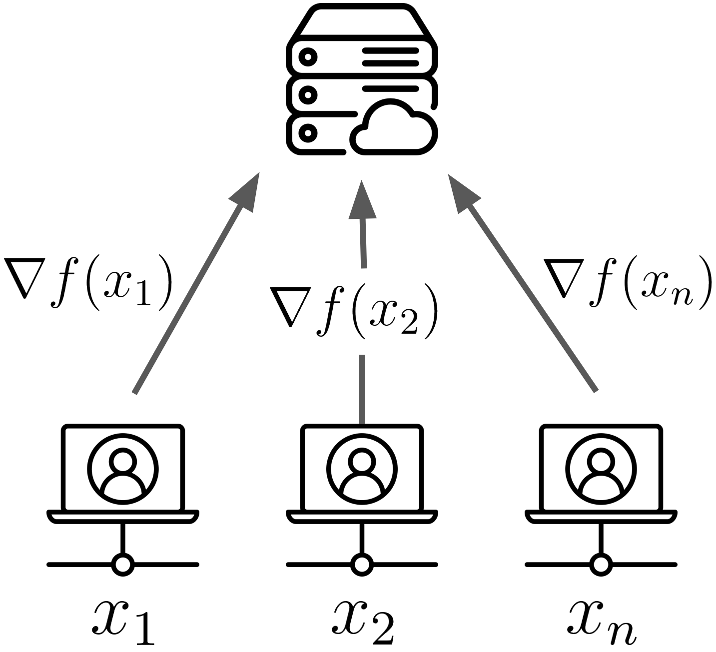
In **federated learning** a model is trained in a distributed fashion where clients only send **gradient updates** to the server in order to **preserve their data privacy**.

Unfortunately, an **adversarial server** can **reconstruct client data** x from received gradient updates:

$$\underset{z}{\operatorname{argmin}}\, \mathcal{E}(\nabla f(x), \nabla f(z))$$

Deep Leakage from Gradients, Nuerips 2019, Zhu et. al https://arxiv.org/abs/1906.08935

Federated Learning and Gradient Inversion



In **federated learning** a model is trained in a distributed fashion where clients only send **gradient updates** to the server in order to **preserve their data privacy**.

Unfortunately, an **adversarial server** can **reconstruct client data** x from received gradient updates:

$$\underset{z}{\operatorname{argmin}}\, \mathcal{E}(\nabla f(x), \nabla f(z))$$

Key Question: When is it possible to exactly recover the client data?

Deep Leakage from Gradients, Nuerips 2019, Zhu et. al https://arxiv.org/abs/1906.08935

Low Rankness of Gradient Updates

Assume linear layer with ReLU activation:

$$Z = W \cdot X + b \qquad X \in \mathbb{R}^{N \times B}$$

$$Y = ReLU(Z) \qquad W \in \mathbb{R}^{M \times N}$$

Low Rankness of Gradient Updates

Assume linear layer with ReLU activation:

$$Z = W \cdot X + b \qquad X \in \mathbb{R}^{N \times B}$$
$$Y = ReLU(Z) \qquad W \in \mathbb{R}^{M \times N}$$

The gradient of the linear layer can be written in the form:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial W} = \frac{\partial \mathcal{L}(f(x),y)}{\partial Z} \cdot X^T$$

Low Rankness of Gradient Updates

Assume linear layer with ReLU activation:

$$Z = W \cdot X + b \qquad X \in \mathbb{R}^{N \times B}$$
$$Y = ReLU(Z) \qquad W \in \mathbb{R}^{M \times N}$$

The gradient of the linear layer can be written in the form:

$$\underbrace{\frac{\partial \mathcal{L}(f(x),y)}{\partial W}}_{M \times N} = \underbrace{\frac{\partial \mathcal{L}(f(x),y)}{\partial Z}}_{M \times B} \cdot \underbrace{X^T}_{B \times N}$$

Key Observation: The gradient of W is not full rank for $B < \min(N, M)$.

Sparse Gradients of ReLU Activations

Assume linear layer with ReLU activation:

$$Z = W \cdot X + b \qquad X \in \mathbb{R}^{N \times B}$$
$$Y = ReLU(Z) \qquad W \in \mathbb{R}^{M \times N}$$

Gradients of ReLU activations:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial Z} = \frac{\partial \mathcal{L}(f(x),y)}{\partial Y} \underbrace{\odot \, \mathbb{I}_{[Z>0]}}_{\text{0/1 mask based on the sign of Z}}$$

Sparse Gradients of ReLU Activations

Assume linear layer with ReLU activation:

$$Z = W \cdot X + b \qquad X \in \mathbb{R}^{N \times B}$$
$$Y = ReLU(Z) \qquad W \in \mathbb{R}^{M \times N}$$

Gradients of ReLU activations:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial Z} = \frac{\partial \mathcal{L}(f(x),y)}{\partial Y} \underbrace{\odot\ \mathbb{I}_{[Z>0]}}_{\text{0/1 mask based on the sign of Z}}$$

Key Observation: The ReLU activation makes gradient of Z sparse

Gradient Decomposition

Use **SVD** to create **low-rank decomposition** of the **gradient of W**:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial W} = U \cdot \Sigma \cdot V^T = \underbrace{\overbrace{U \cdot \sqrt{\Sigma}}^{L}}_{M \times B} \cdot \underbrace{\overbrace{\sqrt{\Sigma} \cdot V^T}^{R}}_{B \times N}$$

Gradient Decomposition

Use **SVD** to create **low-rank decomposition** of the **gradient of W**:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial W} = U \cdot \Sigma \cdot V^T = \underbrace{U \cdot \sqrt{\Sigma}}_{M \times B}^{L} \cdot \underbrace{\sqrt{\Sigma} \cdot V^T}_{B \times N}^{R}$$

We show that under mild assumptions: there exists an **unique** $Q \in \mathbb{R}^{B \times B}$ **s.t.:**

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial Z} = L \cdot Q \qquad X^T = Q^{-1} \cdot R$$

Key Observation: The Low-Rankness simplifies the problem from N X B to B X B

Exploiting Gradient Sparsity (Example)

Assume that **first 3 neurons are not activated** for some input x in a batch with B=3:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial z} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.1 \\ \vdots \\ 0.5 \end{bmatrix} = L \cdot q$$

Exploiting Gradient Sparsity (Example)

Assume that **first 3 neurons are not activated** for some input x in a batch with B=3:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial z} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.1 \\ \vdots \\ 0.5 \end{bmatrix} = L \cdot q \quad \Rightarrow \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = L_{[1,2,3]} \cdot q$$

Exploiting Gradient Sparsity (Example)

Assume that **first 3 neurons are not activated** for some input x in a batch with B=3:

$$\frac{\partial \mathcal{L}(f(x),y)}{\partial z} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.1 \\ \vdots \\ 0.5 \end{bmatrix} = L \cdot q \quad \Rightarrow \quad \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = L_{[1,2,3]} \cdot q$$

If we also assume that $\operatorname{rank}(L_{[1,2,3]}) == 2$, we have:

$$q = c \cdot \operatorname{basis}(L_{[1,2,3]})$$

Exploiting Gradient Sparsity

Theorem: For **randomly initialized networks** with probability approximately $\frac{1}{2^b}$ a vector **q'** obtained that way is a column of **Q(up to scale).**

Key Idea: Sample random submatrices of L and obtain the respective q

Exploiting Gradient Sparsity

Theorem: For **randomly initialized networks** with probability approximately $\frac{1}{2^b}$ a vector **q'** obtained that way is a column of **Q(up to scale).**

To discard bad vectors **q'** we compute **the sparsity** of their associated gradient:

$$\|\frac{\partial \mathcal{L}(f(x),y)}{\partial z'}\|_0 = \|L \cdot q'\|_0 < \tau \cdot M$$

Key Idea: Sample random submatrices of L and obtain the respective q'

Recovering Scale

Let $q_i = c_i q_i'$, where c_i is **unknown**. Then one **can recover** c_i by:

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_B \end{bmatrix} = L^{-L} \cdot \overline{Q}^{\,-1} \cdot \frac{\partial \mathcal{L}(f(x),y)}{\partial b} \quad \text{with} \quad \overline{Q} = \begin{bmatrix} | & | & & | \\ q_1' & q_2' & \cdots & q_B' \\ | & | & & | \end{bmatrix}$$

Key Observation: The gradient of the bias b can be used to recover scale.

Experimental Results

Main Results

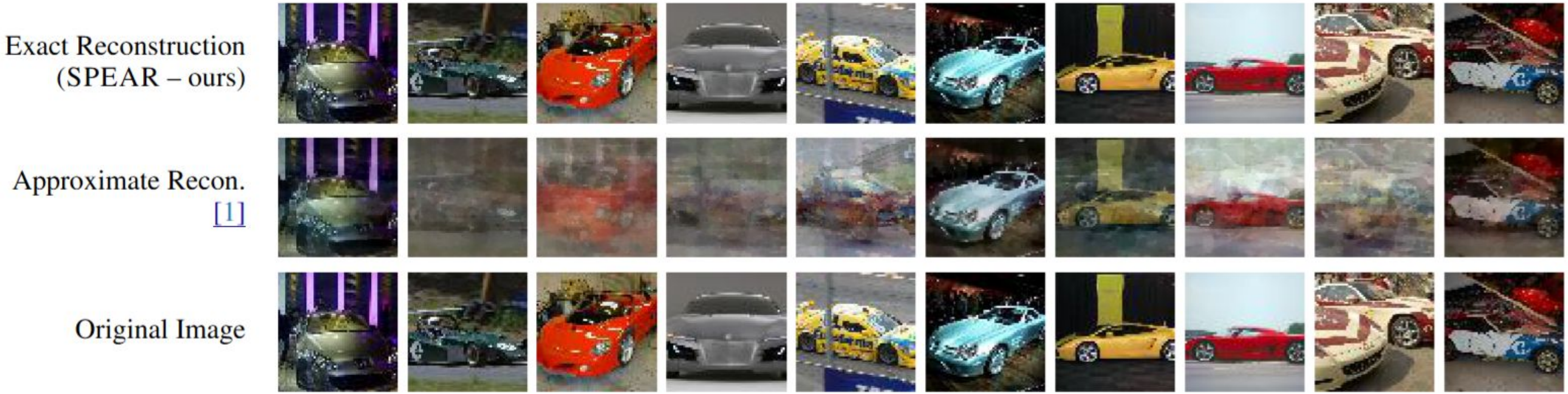
We run our experiments on **6 layers FCNN** with **batch size** B=20 on **image data**:

Method	PSNR ↑	Time/Batch
CI-Net [12] Sigmoid	38.0	1.6 hrs
CI-Net [12] ReLU	15.6	1.6 hrs
Geiping et al. [1]	19.6	18.0 min
SPEAR (Ours)	**124.2**	**2.0 min**

Dataset	PSNR ↑	LPIPS ↓	Acc (%) ↑	Time/Batch
MNIST	99.1	NaN	**99**	2.6 min
CIFAR-10	106.6	1.16×10^{-5}	**99**	1.7 min
TinyImageNet	110.7	1.62×10^{-4}	**99**	**1.4 min**
ImageNet 224 × 224	125.4	1.05×10^{-5}	**99**	2.1 min
ImageNet 720 × 720	**125.6**	$\mathbf{8.08\times10^{-11}}$	**99**	2.6 min

Generative Gradient Inversion via Over-Parameterized Networks in Federated Learning, ICCV 2023, Zhang et. al https://openaccess.thecvf.com
Inverting Gradients -- How easy is it to break privacy in federated learning?, Neurips 2020, Geiping et. al https://arxiv.org/abs/2003.14053

Main Results - Visualisation



Inverting Gradients -- How easy is it to break privacy in federated learning?, Neurips 2020, Geiping et. al https://arxiv.org/abs/2003.14053

Tabular Data Results

We also work with **tabular data** where FCNNs **are common**:

Method	Discr Acc (%) ↑	Cont. MAE ↓	Time/Batch
Tableak [8]	97	4922.7	2.6 min
SPEAR (Ours)	**100**	**20.4**	**0.4 min**

TabLeak: Tabular Data Leakage in Federated Learning, ICML 2023, Vero et. al https://arxiv.org/abs/2209.05578

Convolutional Network Results

We first recover the **input features** to the **linear layers** of the **CNN** and then execute a combination of **feature inversion** and **gradient inversion attack**:

Method	LPIPS ↓	Feature Sim ↑
Geiping et al. [1]	0.562	-
CPA[9] + FI + Geiping et al. [1]	0.388	0.939
SPEAR + FI + Geiping et al. [1]	**0.362**	**0.984**

Cocktail Party Attack: Breaking Aggregation-Based Privacy in Federated Learning using Independent Component Analysis, ICML 2023, Kariyappa et. al https://arxiv.org/abs/2209.05578

Further details can be found in the paper.

NeurIPS 2024 page:

OpenReview: