The Resource Problem of Using Linear Layer Leakage Attack in Federated Learning

Joshua C. Zhao[1], Ahmed R. Elkordy[2], Atul Sharma[1], Yahya H. Ezzeldin[2], Salman Avestimehr[2], Saurabh Bagchi[1]

Purdue University[1], University of Southern California[2]

CVPR ‘23, TUE-AM-379

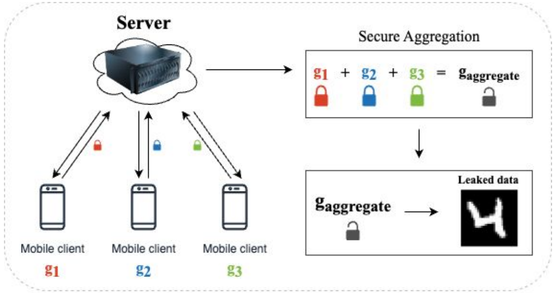
Presentation overview

- Introduction
- Federated learning
- Linear layer leakage
- Resource challenges
- MANDRAKE
- Key idea: Sparsity
- Experiments

Federated learning

- Federated learning was proposed to allow model training in a decentralized fashion while still maintaining privacy of user data.
- General training round:
    - Server sends a global model to participating clients
    - Clients train the model on local data and send their local update (encrypted) to the server
    - Server aggregates the received updates and updates the global model

- Prior work has shown user data can still be leaked through gradients

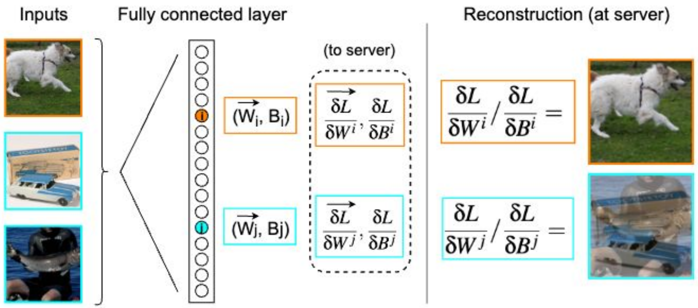


Linear layer leakage

- Inputs to a fully-connected (linear) layer can be directly leaked through the gradients of the layer. This can be in the input images if placed at the start.

$$x = \frac{\delta L}{\delta W^i} / \frac{\delta L}{\delta B^i}$$

- This requires a single input image to activate a neuron in the layer
    - If multiple images activate a single neuron, the reconstructions fail
- Robbing the Fed[1] (RtF) built upon this idea and proposed a more efficient linear layer leakage approach
    - Higher leakage rate
    - Better scalability (batch size/secure aggregation)



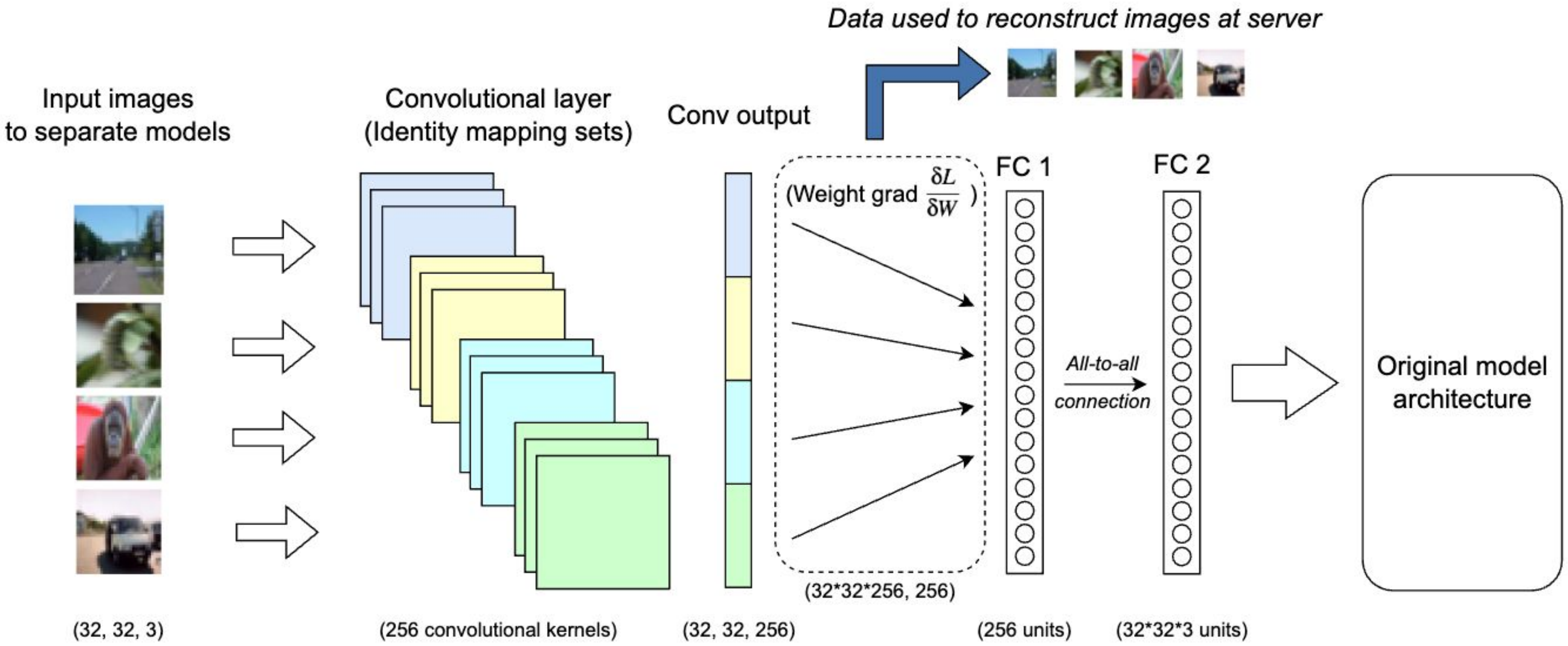
[1]Fowl et. al., "*Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models*". ICLR, 2022.

Resource challenges for linear layer leakage

Secure aggregation only allows a server to view an aggregated update. Individual updates are encrypted.

- Linear layer leakage (Robbing the Fed) can still attack secure aggregation and maintain a high leakage rate by scaling up the size of the linear layer. However, **the model overhead can be extremely large**.
    - For batch size 64 and 1 client on Tiny ImageNet, RtF gets ~77% leakage with FC size of 256.
    - With 100 clients (64 * 100 = 6,400 total images), and FC layer size of 25,600 is required.
        - 2.34GB model size overhead

MANDRAKE[2]

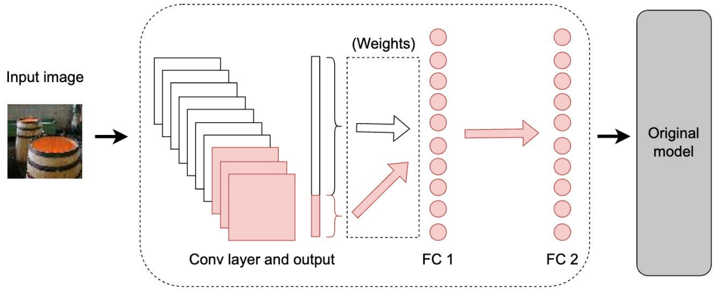


[2]Zhao et. al., "*Secure Aggregation in Federated Learning is not Private: Leaking User Data at Large Scale through Model Modification*". 2023.

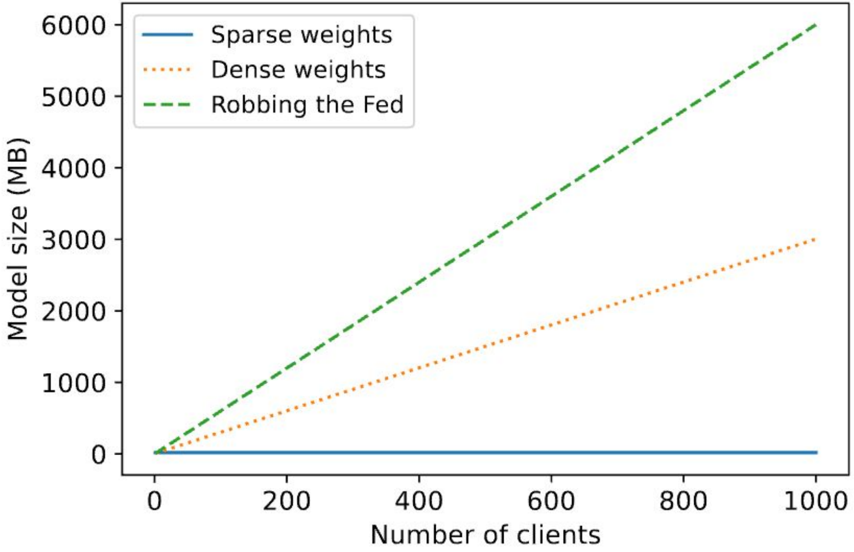
Key idea: Sparsity

- The number of weight parameters in the model needs to be large enough to store the pixel information of all images.
    - Will be even more weight parameters (as leakage isn't perfectly efficient)
    - Increases multiplicatively with a larger number of clients

- However, this increase comes from an incorrect perspective of treating an aggregate update as attacking a large super-batch. Even for an aggregate attack, **clients models only need enough parameters to store their own images.** All other params can be zero.
- This allows for sparsity to decrease the resource cost of model size and computation.



Experiments

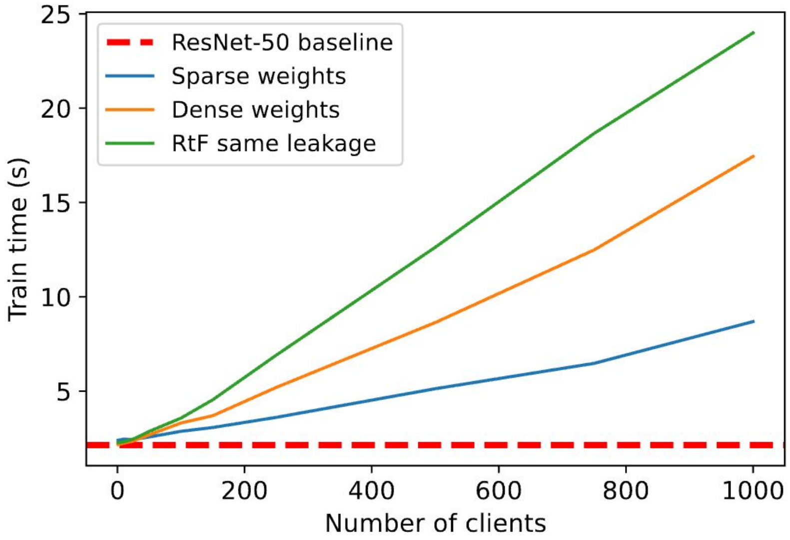


	Model size (MB)	Sparse attack	Robbing the Fed
MobileNet v3 (L)	20.9161	87.65%	28690.76%
ResNet-18	44.5919	41.11%	13457.57%
ResNet-50	97.4923	18.80%	6155.35%
Inception v3	103.6120	17.69%	5791.79%
VGG-11	506.8334	3.62%	1184.02%

Experiments

	Clients	Robbing the Fed	Dense weights	Sparse weights
MNIST (28x28x1)	100	153.2	77.3	4.6
	1000	1532.2	766.4	4.6
CIFAR-100 (32x32x3)	100	600.1	303.0	18.0
	1000	6001.0	3003.3	18.3
Tiny ImageNet (64x64x3)	100	2400.1	1212.1	72.1
	1000	24001.0	12012.4	72.4
ImageNet (256x256x3)	100	38400.9	19392.8	1152.8
	1000	384001.7	192193.1	1153.1

	Sparse MANDRAKE	Robbing the Fed
CIFAR-100	77.5% (4957)	77.1% (4931)
MNIST	71.0% (4546)	75.1% (4803)
Tiny ImageNet	77.8% (4978)	77.7% (4970)



Thank you!