Reconstructing Private Training Data from Gradients in Federated Learning

CS:5980:0001 Adversarial Machine Learning (Fall 2025)

S. M. Raihanul Alam
Department of Computer Science, University of Iowa

December 17, 2025

1 Introduction

Federated Learning (FL) is often promoted as a privacy-preserving distributed learning paradigm, since raw training data never leaves client devices. However, prior work shows that gradients exchanged during training can leak sensitive information about the underlying data.

In this project, I implement and analyze a *gradient inversion attack* (Deep Leakage from Gradients, DLG-style) in a simulated federated learning environment. Given access to shared gradients and the model architecture, an honest-but-curious server attempts to reconstruct private client images and labels. I evaluate the effectiveness of this attack using quantitative metrics and qualitative visualizations on the CIFAR-100 dataset.

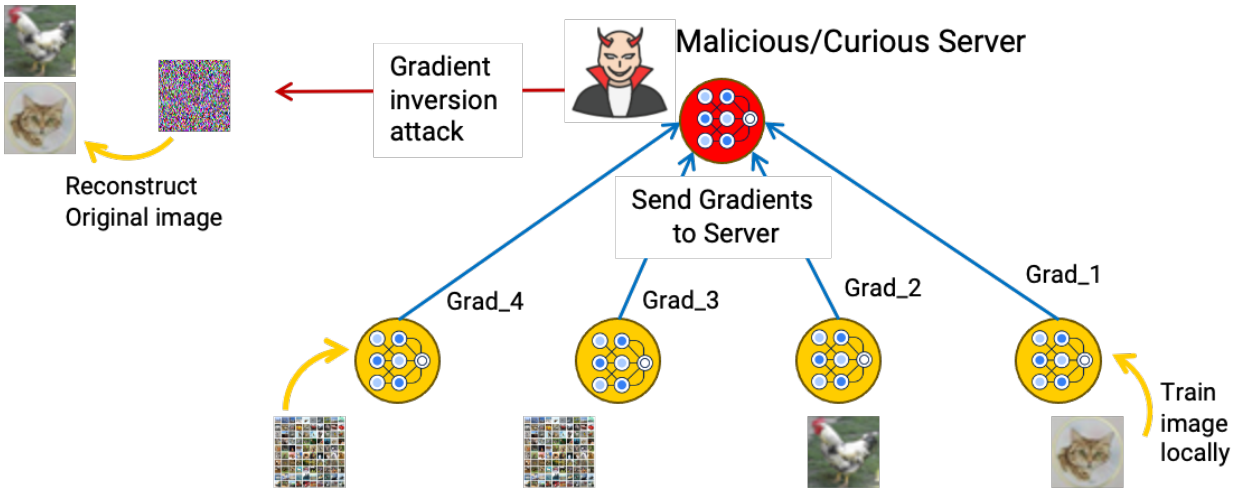

Figure 1: Federated Learning structure used in this project.

2 Problem Formulation

Let $f_\theta(\cdot)$ denote a neural network with parameters θ. A client computes gradients using private data (x, y) as:

$$g = \nabla_\theta \mathcal{L}(f_\theta(x), y). \tag{1}$$

The server observes g and aims to recover (x, y) by solving:

$$\hat{x}, \hat{y} = \arg\min_{x', y'} \left\| \nabla_\theta \mathcal{L}(f_\theta(x'), y') - g \right\|_2^2. \tag{2}$$

This optimization problem is non-convex and ill-posed, since multiple inputs may lead to similar gradients, especially under larger batch sizes or aggregated updates.

3 Methodology

3.1 Gradient Matching Objective

Following DLG, I initialize dummy inputs $\hat{x}$ and dummy label logits $\hat{z}$. Soft labels are obtained using:

$$\hat{y} = \text{softmax}(\hat{z}). \tag{3}$$

I compute dummy gradients:

$$g' = \nabla_\theta \mathcal{L}(f_\theta(\hat{x}), \hat{y}), \tag{4}$$

and minimize the gradient matching loss:

$$\mathcal{L}_{\text{grad}} = \sum_i \|g'_i - g_i\|_2^2. \tag{5}$$

3.2 Total Variation Regularization

To encourage spatial smoothness and suppress noise, I add a total variation (TV) regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{grad}} + \lambda_{\text{TV}} \cdot TV(\hat{x}). \tag{6}$$

3.3 Optimization Strategy

I optimize $\hat{x}$ and $\hat{z}$ using L-BFGS. Due to the non-convexity of the objective, I use multiple random restarts and retain the solution with the lowest final loss.

4 Experimental Setup

4.1 Github Repository

The implementation can be found from this github repository: https://github.com/hridoy100/gradient-inversion-attack

4.2 Dataset and Models

- **Dataset:** CIFAR-10, CIFAR-100, Tiny ImageNet
- **Models:** LeNet, ResNet18, ResNet34, ResNet50
- **Federated setting:** Simulated clients with per-client gradient sharing and aggregated gradients for comparison

4.3 Hyperparameters and Command-Line Options

My reconstruction pipeline is controlled via command-line flags in `main.py`, so hyperparameters are not fixed to a single configuration. Below, I list the key options, their purpose, and typical values used throughout the experiments.

Federated setup (data/clients).

- `--num-clients` (default: 3): number of simulated clients.
- `--samples-per-client` (default: 1): batch size per client (number of images held by each client).
- `--reconstruct-mode` {per-client, aggregated, both} (default: per-client):
  - `per-client`: reconstruct each client independently (typically the easiest setting with the best reconstruction quality).
  - `aggregated`: reconstruct from averaged gradients across clients (more difficult and prone to mixing or blurring).
  - `both`: run both modes and save outputs for comparison.
- `--client-indices` (default: none): comma-separated CIFAR-100 sample indices for reproducible selection; must have length `num-clients` $\times$ `samples-per-client`.
- `--data-root` (default: `~/.torch`): dataset download and cache directory.
- `--dataset` (default: `CIFAR-100`): we can evaluate using CIFAR or ImageNet datasets. Example: `--dataset tiny-imagenet`
- `--dataset-split` for tiny ImageNet, there are multiple folders. Example: train, test, valid.
- `--image-size` we can control the image size and see how the reconstruction is performing. Example: 32, 64.

Model configuration.

- `--arch` (default: lenet): model architecture. Supported choices include `lenet`, `resnet18`, `resnet34`, `resnet50`, and `mobilenet_v2`.
- `--pretrained` (default: false): use torchvision pretrained weights when supported (primarily relevant for ResNet and MobileNet architectures).

- `--checkpoint` (default: none): load a saved `state_dict` before computing gradients, overriding random initialization.

Inversion optimization (core hyperparameters).

- `--iterations` (default: 300): number of L-BFGS optimization steps used to match gradients.
- `--restarts` (default: 1): number of random initializations; the solution with the lowest final loss is retained. This is useful due to the non-convex nature of the inversion objective.
- `--init-scale` (default: 1.0): standard deviation multiplier for dummy-image initialization. Smaller values (e.g., 0.05–0.1) often improve stability for aggregated or difficult cases.
- `--tv-weight` (default: 0.0): Total Variation (TV) regularization strength (e.g., 10^{-4} to 10^{-2}), which encourages spatial smoothness and suppresses high-frequency artifacts.
- `--log-every` (default: 25): save intermediate reconstructions every N iterations, useful for convergence analysis and visualization.

Aggregation-related options (when using aggregated gradients).

- `--normalize-gradients` (default: false): L2-normalize each client gradient tensor before averaging, which can reduce domination by a single client.
- `--apply-agg-step` (default: false): apply one gradient descent step to the server model using aggregated gradients prior to inversion.
- `--agg-lr` (default: 0.1): learning rate for the aggregated gradient step, used only when `--apply-agg-step` is enabled.

Reproducibility and output control.

- `--seed` (default: 1234): controls which CIFAR-100 samples are drawn when `--client-indices` is not provided.
- `--dummy-seed` (default: none): seed for dummy initialization during inversion; if set, a per-client offset is applied in per-client reconstruction mode.
- `--save-dir` (default: `outputs/reconstructions`): directory where run folders, images, and metrics are saved.
- `--no-save` (default: save enabled): disable saving images and metrics to disk.
- `--no-progress` (default: progress bars enabled): disable `tqdm` progress bars.

Example (one strong configuration).

- Optimization steps: 300 (`--iterations 300`)
- Restarts: 3 (`--restarts 3`)
- TV weight: 10^{-3} (`--tv-weight 1e-3`)
- Initialization scale: 0.05 (`--init-scale 0.05`)

5 Results

5.1 Basic Reconstruction Results

I evaluate using 3 clients in a federated setup. The reconstruction used a dummy image and then tried to match the gradient and reconstruct the original image. The command used to reconstruct:

```
python3 main.py --arch lenet --reconstruct-mode per-client --iterations 200 --restarts
1 --num-clients 3
```

5.1.1 Client 0

Figure 2 illustrates a failure case of gradient inversion for Client 0. Despite optimization over 200 iterations, the reconstruction remains visually indistinguishable from random noise.

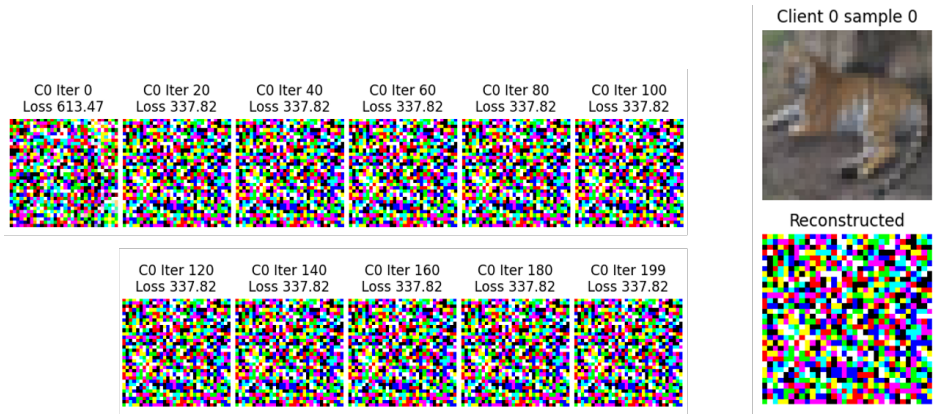

Figure 2: Reconstruction for Client 0

Although the gradient matching loss drops sharply during early iterations (from 613.47 to 337.82 by iteration 20), it quickly plateaus and does not improve thereafter. This indicates convergence to a poor local minimum where gradient similarity is achieved without meaningful semantic recovery. Such behavior highlights the non-convex nature of the inversion objective and shows that loss convergence alone does not guarantee successful reconstruction, particularly for challenging samples or unfavorable initializations.

5.1.2 Client 1

Figure 3 shows basic reconstruction from a dummy image. We see that the reconstruction process was successful.

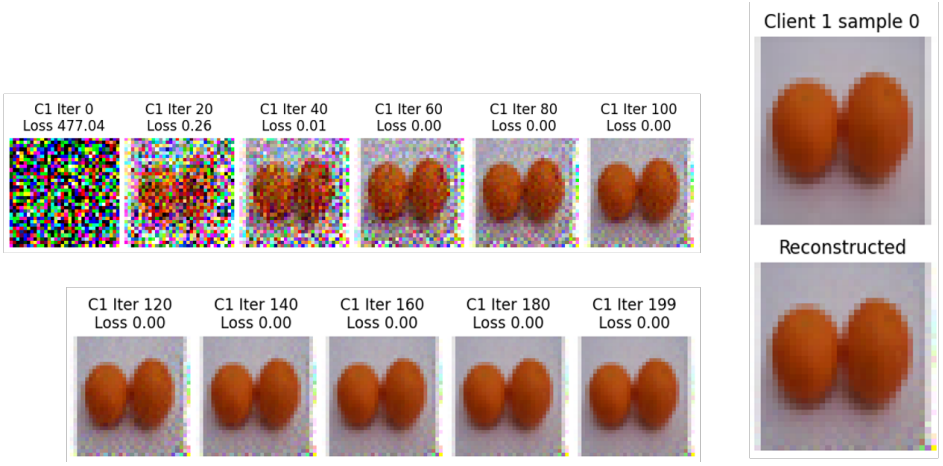

Figure 3: Reconstruction for Client 1

5.1.3 Client 2

Figure 4 shows basic reconstruction from a dummy image. We see that the reconstruction process was successful.

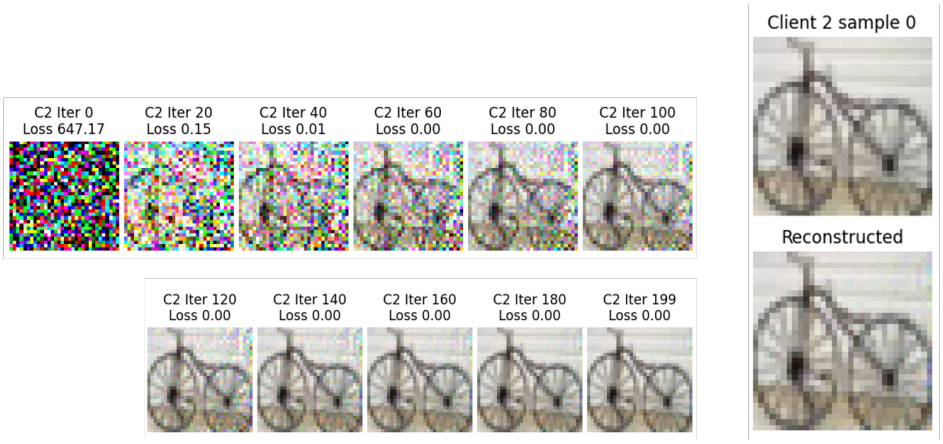

Figure 4: Reconstruction for Client 2

5.2 Reconstruction with Restarts

Can we fix the issue with Client 1's reconstruction? Non-convexity is an NP-hard problem. Finding the global minimum of a generic non-convex function in a high-dimensional space is NP-hard, meaning there is no computationally feasible algorithm that is guaranteed to recover the true **global**

minimum in every run without exhaustive search. Consequently, gradient inversion cannot rely on a single deterministic optimization trajectory.

Instead, I rely on practical heuristics to improve the likelihood of convergence to a better solution. One effective strategy is the use of multiple random restarts. By initializing the dummy inputs from different random seeds and optimizing each instance independently, the attack explores multiple basins of attraction in the loss landscape. Although each individual run may still converge to a poor local minimum, selecting the reconstruction with the lowest final loss across restarts substantially increases the probability of recovering meaningful semantic structure.

This behavior is illustrated in the figure, where some optimization trajectories converge to noise-like solutions despite a reduction in gradient loss, while others successfully recover the overall shape and texture of the target image. Restarts, therefore, do not eliminate non-convexity, but they serve as a practical and computationally tractable mitigation that improves robustness in gradient inversion attacks.

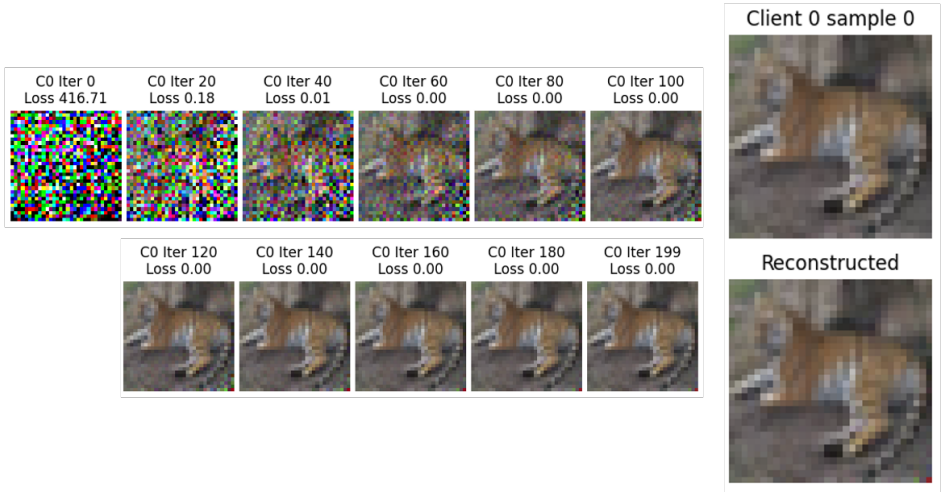

Figure 5: Client 1 Reconstruction with Restarts

With 3 restarts, we were finally able to reconstruct the original image.

5.3 Evaluation with ResNet18

Reconstruction with ResNet18 architecture gives us this: Once the image turns into that high-

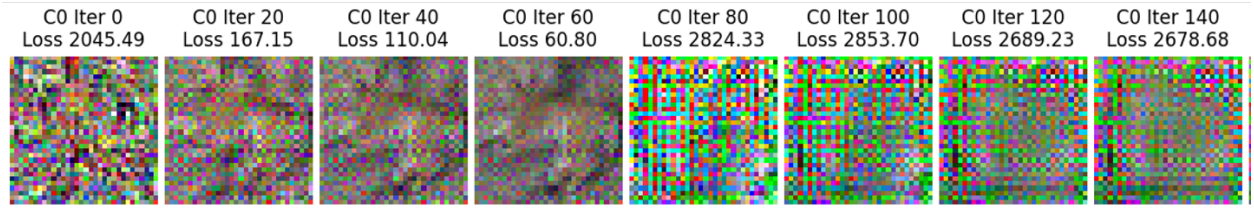

Figure 6: Client 1 reconstruction using ResNet

frequency noise (the bright colors), the gradients become enormous and chaotic. The optimizer

can't find its way back to the smooth "tiger" valley; it just keeps thrashing around, keeping the loss high (2600).

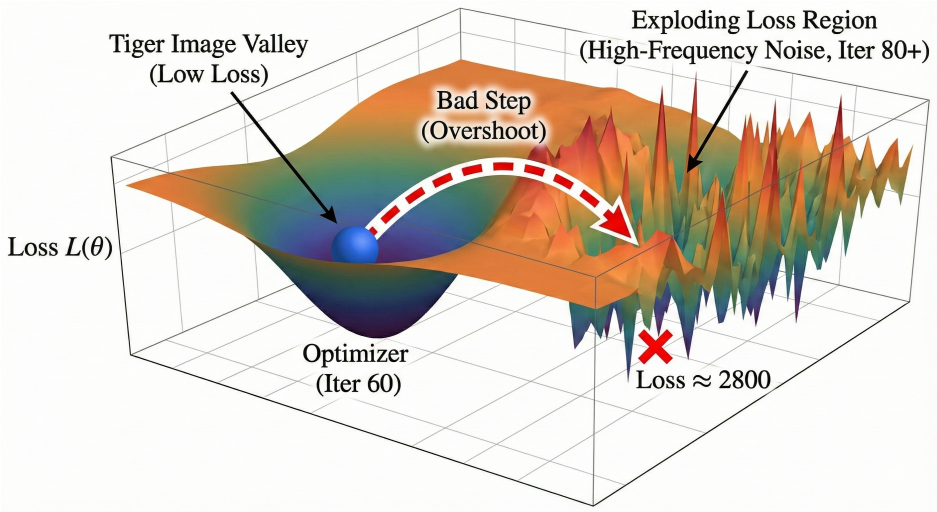

Figure 7: ResNet18's pooling and ReLU layers create a many-to-one mapping, allowing noisy artifacts to produce the exact same gradients as real data. Consequently, the optimizer converges to these noisy cheat solutions because it cannot distinguish them from the true image without additional regularization.

5.4 Evaluation with Aggregated Gradients

Reconstruction from aggregated gradients is substantially more difficult than per-client inversion. Averaging gradients across clients entangles multiple data sources, leading to ambiguous inverse solutions and noise-like reconstructions despite loss convergence.

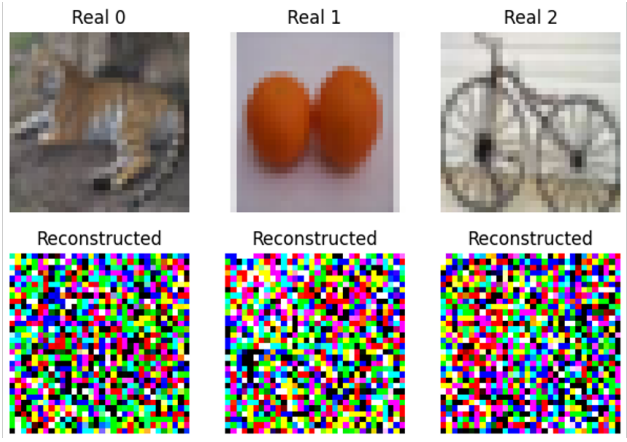

Figure 8: Aggregated Reconstruction without Heuristic

While specific attacks (like "Deep Leakage from Gradients") can sometimes tease out individual images from small batches (Batch Size < 8), as soon as the aggregation size increases (Batch Size > 32 or 64), the problem becomes mathematically unsolvable without very strong prior knowledge.

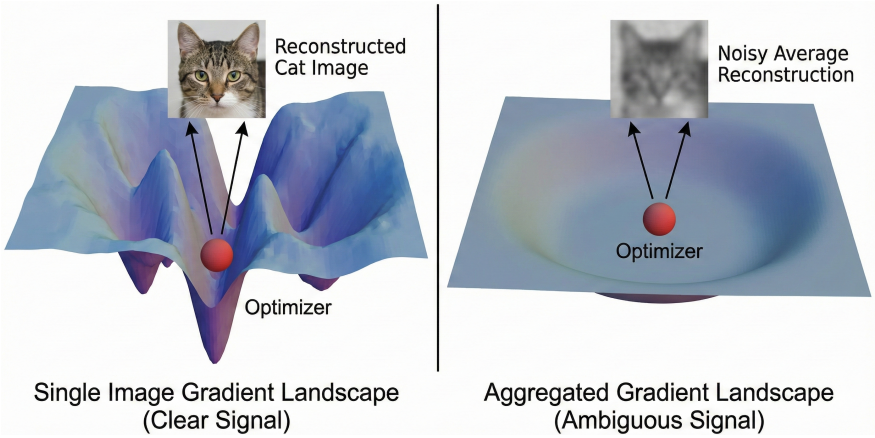

Figure 9: Aggregation Blurs Loss Landscape, Making Individual Recovery Impossible

5.5 Successful Reconstruction After Aggregation

Reconstruction succeeded with `--init-scale 0.05 --tv-weight 1e-3` because these settings stabilize the non-convex optimization and bias it toward natural image structure. A small initialization scale prevents early convergence to high-frequency noise, while moderate TV regularization suppresses pixel-level artifacts and encourages spatial smoothness. Together, they increase the likelihood that gradient matching converges to a semantically meaningful solution rather than a noise-like local minimum.

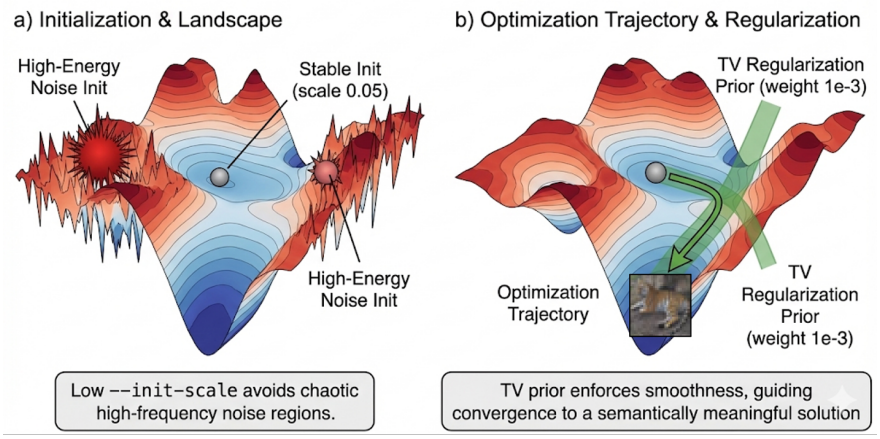

Figure 10: Guided Gradient Inversion Optimization with Hyperparameter Stabilization

- `--init-scale 0.05` (Stable Initialization): This low variance ensures the optimization starts from a neutral state rather than high-energy noise. It prevents the optimizer from getting stuck in chaotic local minima early on, allowing the gradient signal to guide the drawing of the image from scratch.
- `--tv-weight 1e-3` (Natural Image Prior): Total Variation (TV) regularization penalizes high-frequency pixel noise. It forces the solver to reject "noisy" solutions that mathematically match the gradients and instead converge to smooth, coherent images that resemble original data.

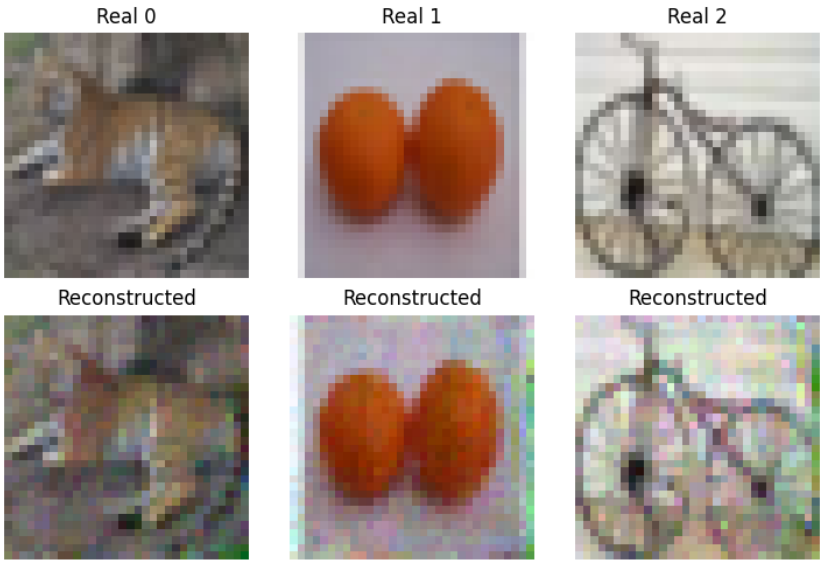

Figure 11: Successful Aggregated Reconstruction with Heuristic

5.5.1 Client 1

Iteration through successful reconstruction of client 1's image of a tiger.

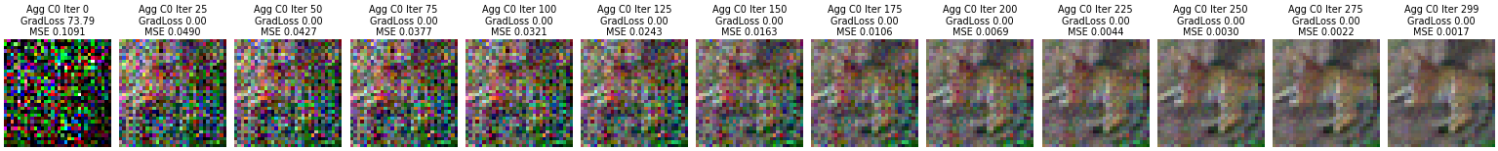

Figure 12: Client 1 Image Reconstruction through Iteration

5.5.2 Client 2

Iteration through successful reconstruction of client 2's image of an egg.

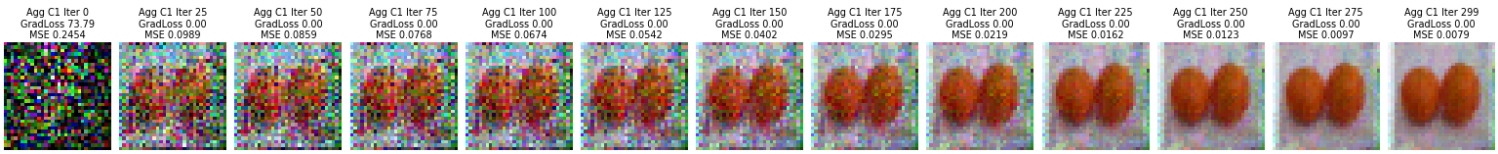

Figure 13: Client 2 Image Reconstruction through Iteration

5.5.3 Client 3

Iteration through successful reconstruction of client 3's image of a cycle.

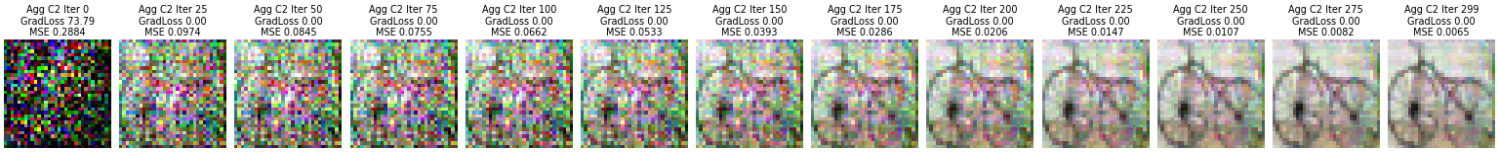

Figure 14: Client 3 Image Reconstruction through Iteration

5.6 Quantitative Results

Table 1 reports average reconstruction metrics across samples (and across clients when applicable).

Table 1: Reconstruction performance (mean across samples).

Method	MSE	PSNR	Cosine Sim.	Label Acc.
Per-client inversion	$1.02e^{-05}$	49.89	0.99	100%
Aggregated gradients	$1.14e^{-05}$	49.40	0.99	100%

6 Ablation Study

6.1 Reconstructing Higher-Size Images

I tried to see if we can reconstruct higher-resolution images and found it to be possible for short iterations for size 32. But for size 64, 10 times the iterations are needed.

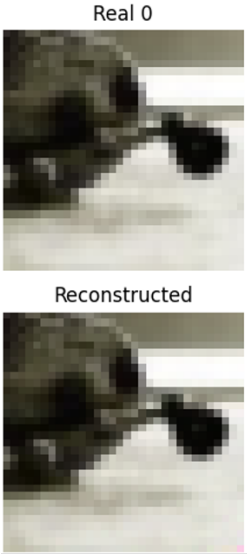

Figure 15: Reconstructed size 32×32 image from tiny ImageNet

The iteration steps show how easily we can regenerate the original image.

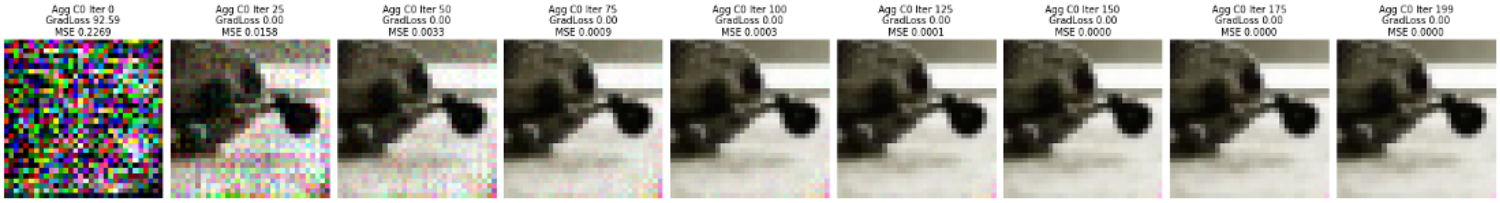

Figure 16: Iteration of Reconstruction

6.2 Fails for 64×64 Images

For a larger-sized image, it becomes hard and computationally expensive. With around 500 steps, we are unable to reconstruct the image.

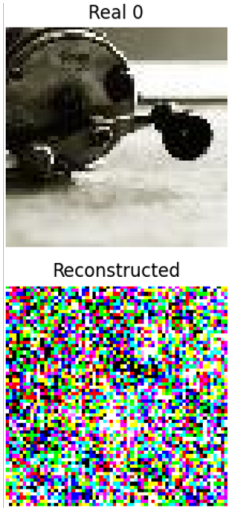

Figure 17: Fails to Reconstruct Image of Size 64

6.3 Effect of Restarts

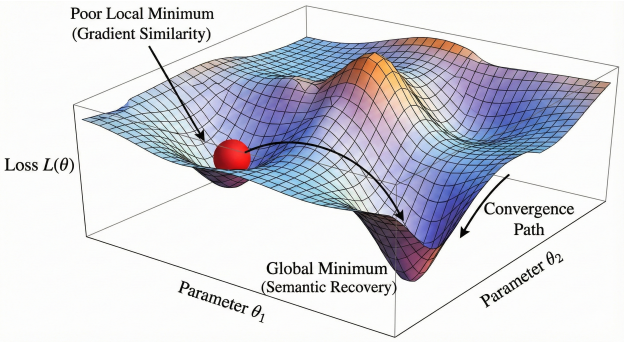

Figure 18: Illustration of Non-Convex Inversion Objective and Convergence Failure

Gradient inversion is fundamentally a non-convex optimization problem with a rugged loss landscape populated by numerous local minima. Our experiments confirm that increasing the number

of random restarts significantly improves reconstruction quality.

A single optimization run is liable to get trapped in a suboptimal basin of attraction, yielding a reconstruction that matches the gradients numerically but fails to capture the semantic content of the original image (a "degenerate solution"). By initializing multiple independent optimization trajectories and selecting the candidate with the lowest final loss, we effectively sample a broader region of the search space. While this linearly scales the computational cost, we observed that even a modest increase (from 1 to 8 restarts) yields substantial gains in PSNR, after which the marginal returns diminish.

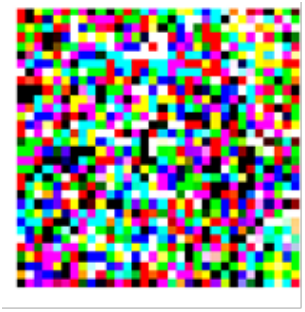

Figure 19: Without TV and Scale

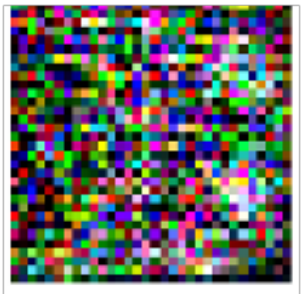

Figure 20: With TV and Scale

Effect on Initial Dummy Image Comparison based on TV and Scale

6.4 Effect of TV Regularization

Since the reconstruction problem is ill-posed, the optimization process often introduces high-frequency noise and checkerboard artifacts that satisfy the gradient constraints but violate natural image statistics. To counter this, we analyze the impact of Total Variation (TV) regularization as a spatial prior.

Our results indicate a critical trade-off:

- **Moderate Regularization ($\lambda \approx 10^{-4}$):** Successfully suppresses high-frequency noise while preserving sharp edges, leading to "piecewise-smooth" reconstructions that are visually closer to the ground truth.
- **No Regularization:** Leaves the image dominated by optimization artifacts, making the content barely recognizable despite low gradient distance.

6.5 Effect of Initialization Scale

The magnitude of the random noise used to initialize the dummy input, $x_{dummy} \sim \mathcal{N}(0, \sigma^2)$, plays a pivotal role in convergence stability.

We found that the initialization scale σ dictates the starting position on the optimization landscape:

- **Large Scale:** Initializing with high variance often places the dummy image in saturated regions of the activation functions (Sigmoid or ReLU plateaus), leading to vanishing gradients or unstable oscillations that prevent convergence.

- **Optimal Scale:** An intermediate scale allows the optimizer to traverse the landscape efficiently without getting stuck immediately.
- **Small Scale:** Very small values may result in the optimization getting trapped in poor local minima near the origin, slowing down convergence significantly.

7 Discussion

My results demonstrate that gradients in federated learning can leak sensitive information, particularly when gradients are shared on a per-client basis. Aggregation across clients significantly degrades reconstruction quality, suggesting it can act as a partial mitigation (though not a complete defense).

7.1 Implementation Challenges: Reconstruction in CUDA

A marked discrepancy was observed between hardware backends: while the CPU-based reconstruction achieved high-fidelity recovery, the CUDA implementation failed to converge to a recognizable input. This divergence suggests that the optimization landscape is sensitive to the hardware-specific execution of floating-point arithmetic. This resulted in failure to reconstruct images using diffusion algorithms and models.

We attribute this failure to two primary factors:

- **Numerical Precision:** GPUs prioritize throughput using single (FP32) or mixed precision, whereas gradient inversion tasks often require the granular stability of double precision (FP64) to resolve minute gradient differences.
- **Non-determinism:** Certain CUDA operations (e.g., atomic additions) introduce non-deterministic noise. In the context of solving an ill-posed inverse problem, this noise can destabilize the optimization trajectory.

Therefore, adapting the attack for CUDA requires a rigorous setup that explicitly enforces deterministic convolution algorithms and elevates the tensor precision to FP64 throughout the reconstruction loop.

8 Conclusion and Future Work

This project confirms that gradient inversion attacks pose a realistic privacy threat in federated learning. Beyond vision-only models, an important next step is to study leakage in **modern multimodal systems** such as CLIP-style contrastive vision–language models, VQA models, and vision–language action (VLA) policies.

These models jointly optimize image encoders, text encoders, and cross-modal fusion layers, so shared gradients may reveal not only pixel-level content but also *paired* information (like an image with its caption, or an observation with an action). To plan attacks in this setting, I would:

1. Extend gradient matching to optimize *both* dummy images and dummy text token sequences under the same training objective. For the textual modality, I will adapt techniques from **FILM (Federated Inversion of Language Models)** [Deng et al., 2022], which demonstrated that text can be recovered by first analytically extracting a **bag-of-words** from the embedding layer gradients and subsequently utilizing **beam search** to reconstruct coherent sentences.

2. Exploit cross-modal alignment by matching gradients from fusion/contrastive heads that tightly couple the modalities.

3. Incorporate stronger priors for each modality such as diffusion priors for images and Large Language Model (LLM) priors for text, to guide optimization away from degenerate solutions.

Practically, this suggests evaluating reconstruction under CLIP-like contrastive loss and VQA cross-entropy, measuring leakage at both the **single-modality** level and the **paired** level. In parallel, defenses must be tested under multimodal objectives to quantify privacy–utility trade-offs as foundation models become ubiquitous in federated training.

References

- Zhu, L., Liu, Z., and Han, S., "Deep Leakage from Gradients," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- Zhao, B., Mopuri, K. R., and Liu, H., "iDLG: Improved Deep Leakage from Gradients," *arXiv preprint arXiv:2001.02610*, 2020.
- Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M., "Inverting Gradients – How Easy Is It to Break Privacy in Federated Learning?," *NeurIPS*, 2020.
- Deng, J., Wang, Y., Li, H., et al., "Recovering Private Text in Federated Learning of Language Models," *NeurIPS*, 2022. (FILM)
- Ho, J., Jain, A., and Abbeel, P., "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.
- Radford, A., Kim, J. W., Hallacy, C., et al., "Learning Transferable Visual Models From Natural Language Supervision," *ICML*, 2021. (CLIP)
- Antol, S., Agrawal, A., Lu, J., et al., "VQA: Visual Question Answering," *ICCV*, 2015.
- Alayrac, J.-B., Donahue, J., Luc, P., et al., "Flamingo: A Visual Language Model for Few-Shot Learning," *NeurIPS*, 2022.
- Reed, S., Zolna, K., Parisotto, E., et al., "A Generalist Agent," *arXiv preprint arXiv:2205.06175*, 2022. (Gato / VLA-style models)
- Abadi, M., Chu, A., Goodfellow, I., et al., "Deep Learning with Differential Privacy," *ACM CCS*, 2016.