Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage
CVPR 2022

Zhuohang Li, Jiaxin Zhang, Luyang Liu, Jian Liu

Presenter : M Anand Krishna
CS22MTECH14003
IIT Hyderabad

Instructor : Prof C Krishna Mohan,
TA : Zarka Bashir

Apr 17, 2023

Contents

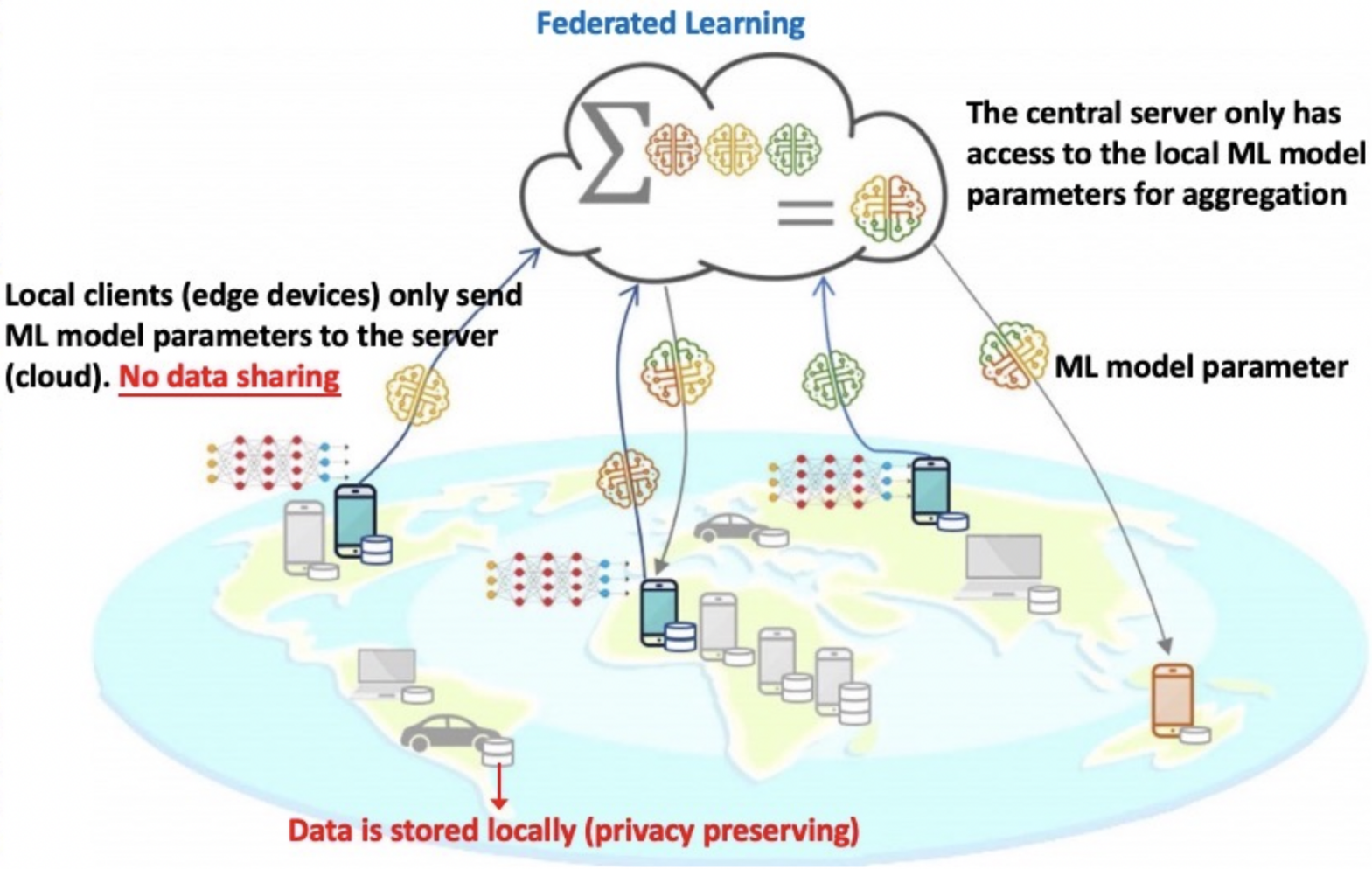
Federated Learning

- Multiple clients collaboratively learn a shared model under coordination of central server

- Clients **do not share private data** instead, they just share the gradient information to central server to update the model

Federated Learning Contd.

- Collaborative learning without centralized training data



Federated Learning Applications.

- Mobile computing
  - e.g. text prediction in Google Gboard
- Telemedicine
  - e.g. multi institutional collaborations for learning medical diagnosis model without sharing patient's private data

Privacy Risks in FL

- Even though only model parameters/gradient information is shared from client, it is found that private information can still be extracted from **exchanged gradients**

Privacy Risks in FL contd..

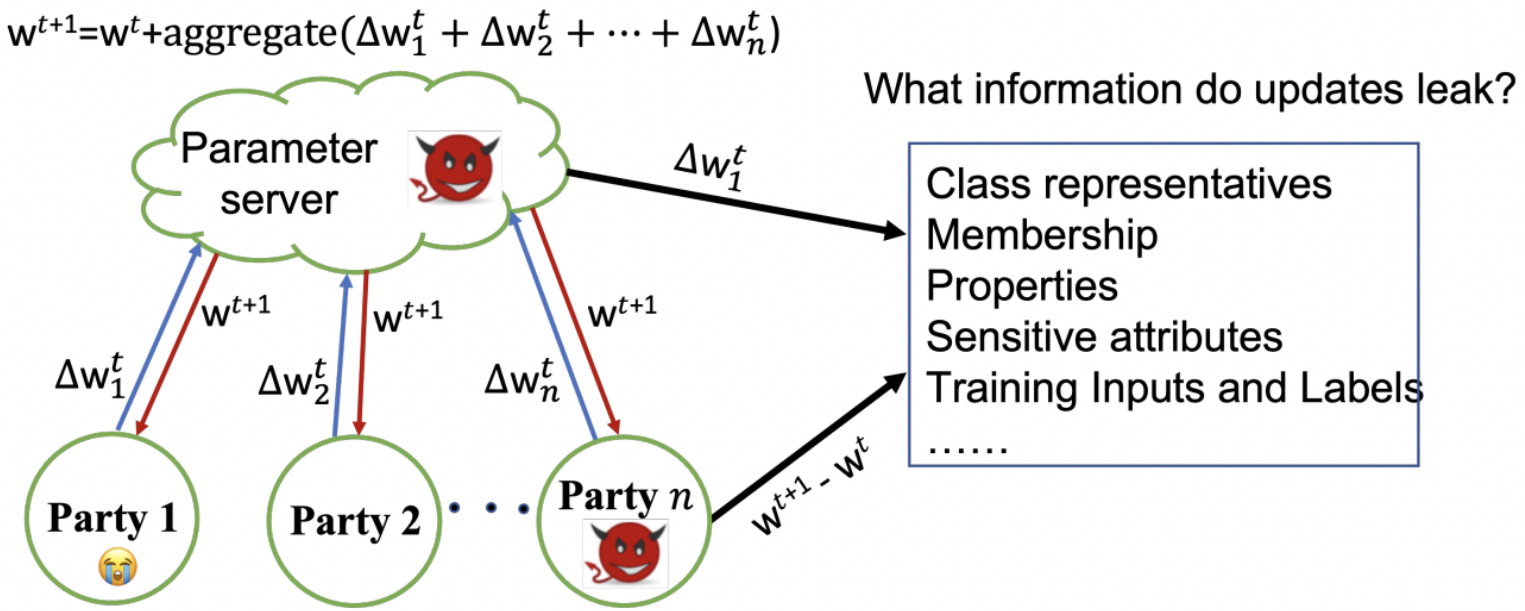


Figure: A demo of privacy leakage in FL. Attacker can infer various private information about the victim participant from the received gradients or the snapshot of the FL model parameters. [2]

[2] Image Credits : **Privacy and Robustness in Federated Learning: Attacks and Defenses**

How to prevent privacy leakage? - Currently Available Privacy Defenses

- Crypto-based solutions
  - e.g. Secure Multi-party computation, Homomorphic encryption. Remains vulnerable to the inference over the output
- Gradient Degradation based solutions
  - Differential Privacy:
    - Use a randomised technique to distort the gradients before sharing it to central server.
    - But there is trade-off in differential privacy is between privacy and utility.
- Gradient Sparsification
  - Compress the gradients by pruning small values

Are these defenses sufficient?

- In this work, author studies the challenging scenario, where clients applies local defenses before sharing the gradients
- And author tries to reconstruct a high fidelity images from the shared degraded gradients.

Are these defenses sufficient? Contd..

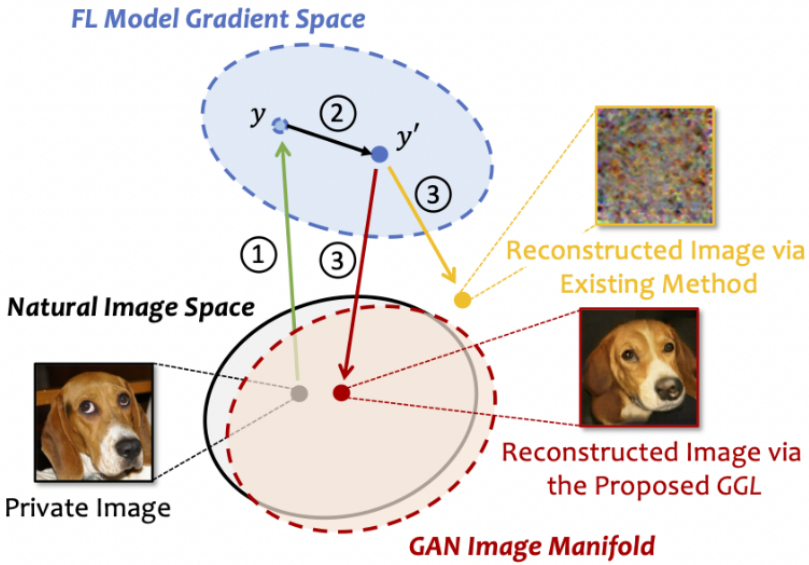


Figure: Illustration of data leakage via gradient.[4]
(1) Client computes gradients on its private data
(2) Client applies defense to degrade the computed gradients y
(3) Adversary attempts to reconstruct the private image from the shared gradients y'

[4]Image Credits : **Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage**

Problem formulation

The task of reconstructing a training image $x \in R^d$ from its gradients $y \in R^m$ can be formulated as a non-linear inverse problem

$$\mathbf{y} = F(\mathbf{x}),$$

where $F(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(f_{\boldsymbol{\theta}}(\mathbf{x}), c\right)$ is forward operator that calculates the gradients of the loss, provided with label c and FL model f_θ
When defense is applied at the client's side, the problem becomes:

$$\mathbf{y} = \mathcal{T}(F(\mathbf{x})) + \boldsymbol{\varepsilon},$$

where $\mathcal{T}(\cdot)$ is referred to as the lossy transformation (e.g., sparsification) and ε is the additive noise (e.g., DP)

Current Approach and Its limitation

- Existing methods aim to solve this inverse problem by using image priors in a penalty form

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathbb{R}^d}{\operatorname{argmin}} \mathcal{D}(\mathbf{y}, F(\mathbf{x})) + \lambda\omega(\mathbf{x}),$$

where $\mathcal{D}(\cdot)$ is a distance metric, $\omega(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$ is the standard image prior (e.g., total variation)

Current Approach and Its limitation

- Only effective for reconstructing images from the actual gradients
- When reconstructing from a set of low-fidelity and noisy gradients, it would suffer from the limited identification ability of hand-crafted priors
- Result: return false solutions that are not valid natural images

Generative Gradient Leakage

- Insight
  - Leverage a generative model trained on public image datasets as a learned natural image prior to ensure that reconstructed image is of good image quality
- Given a well-trained generator $G(\cdot)$, we solve:

$$\mathbf{z}^* = \underset{\mathbf{z}\in\mathbb{R}^k}{\operatorname{argmin}} \underbrace{\mathcal{D}(\mathbf{y}, \mathcal{T}(F(G(\mathbf{z}))))}_{\text{gradient matching loss}} + \lambda \underbrace{\mathcal{R}(G;\mathbf{z})}_{\text{regularization}} ,$$

where $z \in R^k$ is the latent space of the generative model, and $\mathcal{R}(G;\mathbf{z})$ is a regularization term that penalizes latent vectors which deviate from the prior distribution

Optimization strategies

- Challenges
  - The target inverse problem is highly *non convex and non linear*.
  - The existing data reconstruction attacks are based on the **gradient based optimiser** like $L - BFGS$ [5] and *Adam*.

  - Issues with gradient based optimiser
    - Outcome is highly based on choice of initialization
    - For complex models, often high chance that cost function will converge to sub optimal minima

[5] Limited-memory BFGS (L-BFGS or LM-BFGS) is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)

Proposed optimisation strategy

- Author explores two gradient-free optimisation strategies
  - Bayesian optimisation(BO)
  - Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Datasets

- CelebA
  - The dataset was collected from the Internet and contains images of celebrities with a diverse range of ethnicities, ages, and facial features.
- ImageNet
  - ImageNet is a large image database with over 14 million images in 20,000 categories for computer vision tasks.

ImageNet and CelebA has become a popular benchmark dataset for a variety of computer vision tasks, including face recognition, attribute prediction, and generative models.

Results

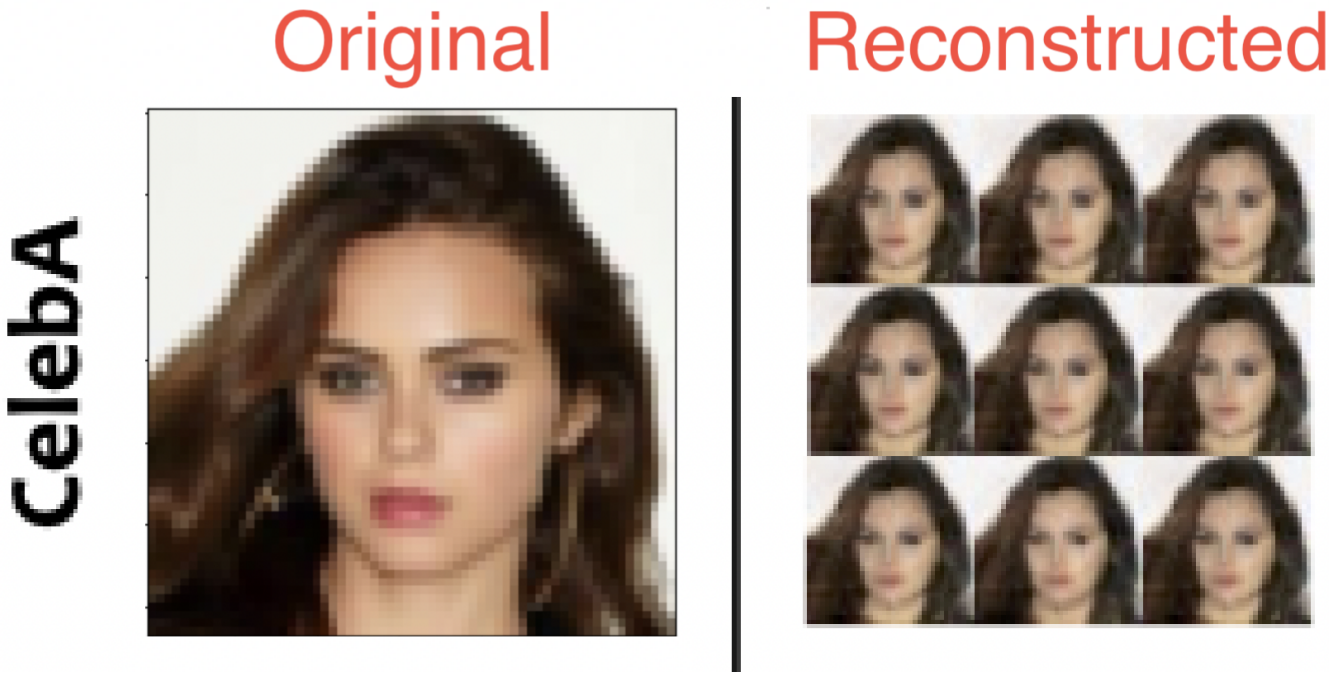


Figure: The images on the right are the reconstruction samples.[6]

[6]Image Credits : **Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage**

My Implementation result

Build the input (ground-truth) gradient

```
idx = 6565
img, label = validloader.dataset[idx]
im1,_ = validloader.dataset[4]
labels = torch.as_tensor((label,), device=setup['device'])
ground_truth = img.to(**setup).unsqueeze(0)
xtest=im1.to(**setup).unsqueeze(0)
plot(ground_truth)
#print([trainloader.dataset.classes[l] for l in labels]);
print(len(validloader.dataset))
```

```
19962
```

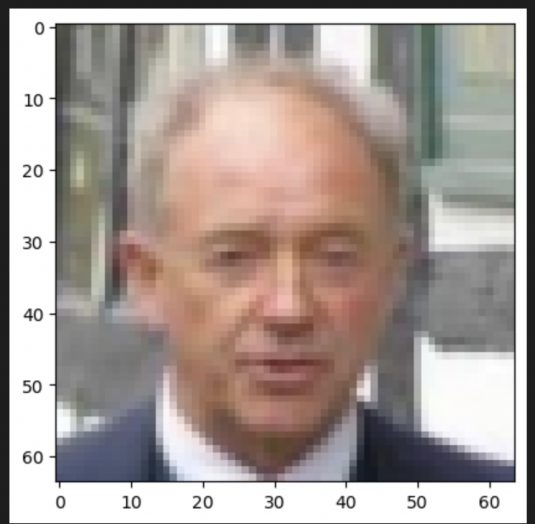


Figure: The Input (ie: Private Data)

My Implementation result(contd)

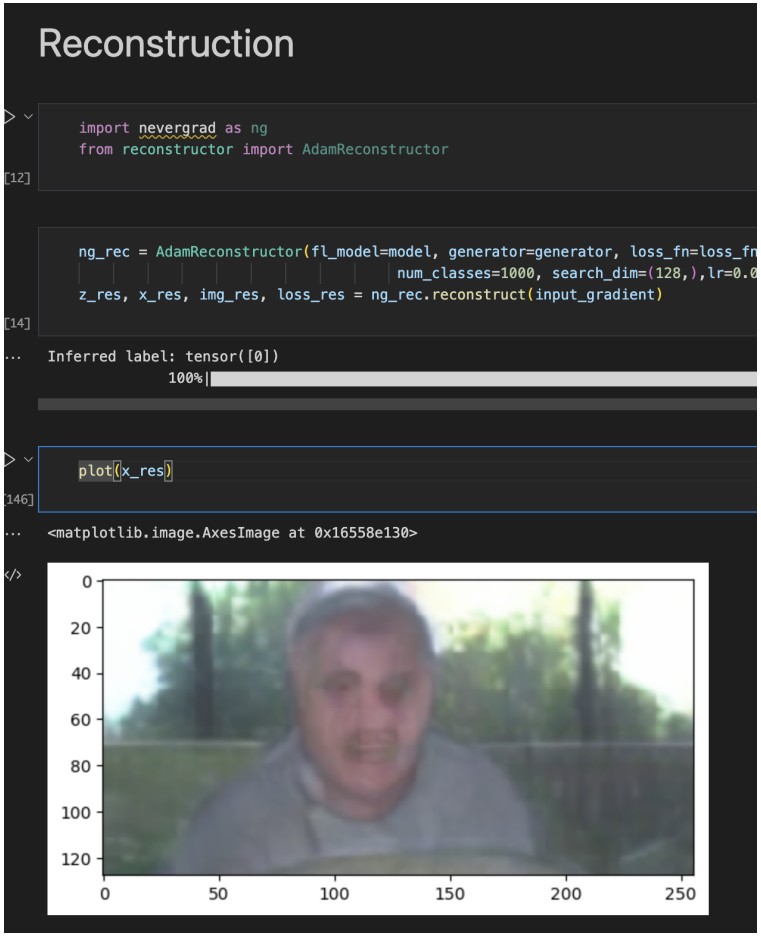


Figure: Reconstructed image through GGL attack

Experiment Results Contd.

Table: Performance Metrics

	Paper Values	**Observed Values**
MSE ↓	0.0427	0.068
PSNR ↑	13.69	5.8
LPIPS ↓	0.1435	0.53

- MSE : the mean squared error (MSE) between the generated and true gradients of the model
- PSNR: the peak signal-to-noise ratio (PSNR) between the generated and true gradients of the model .
- LPIPS: the learned perceptual image patch similarity (LPIPS) between the generated and true gradients of the model

.

Novel idea

- Developing novel privacy defenses specifically designed to defend against the gradient leakage attack
- Degrading the gradients such that attacker won't be able to reconstruct.
- **Gradient quantization** is one such defense technique, where author has not tested his reconstruction attack GGL against it.

Gradient Quantization

- Gradient quantization is a technique that reduces the precision of gradients by using fewer bits to represent their values.
- Gradient quantization can help protect the privacy of the gradients by reducing the amount of information that is transmitted in each round of federated learning.
- This also leads to a reduction in memory usage and communication costs.

Gradient Quantization Steps

- **Determine the number of bits:** Choose the number of bits num_bits for quantization. The number of unique quantization levels will be 2^{num_bits}.
- **Calculate the range of gradient values:** Identify the minimum and maximum values in the gradient tensor.
- **Compute the bin width:** Calculate the bin width for uniform quantization as follows:

$$bin\_width = \frac{max\_value - min\_value}{number\_of\_levels}$$

- **Quantize the gradient values:** For each value in the gradient tensor, perform the following steps:

$$quantized\_value = round\left(\frac{current\_value - min\_value}{bin\_width}\right) \times bin\_width + min\_value$$

- **Replace the original gradient values:** Replace the original gradient values with their quantized counterparts in the gradient tensor.

Why it can be good solution to prevent attack

- Gradient quantization reduces the granularity of information in the gradients by compressing them into fewer unique values.(ie, degrade the sensitive information)
- As a result, the quantized gradients are less representative of the original training data, making it more difficult for an attacker to reconstruct the data from the shared gradients accurately.

Gradient Quantization Implementation details

```python
def quantize_gradient(input_gradient, num_bits=16):
    """
    Quantize gradients to reduce the number of unique values.

    Args:
    - input_gradient (list of torch.Tensor): the input gradient
    - num_bits (int): number of bits for quantization

    Returns:
    - list of torch.Tensor: quantized gradient
    """
    device = input_gradient[0].device
    gradient = [None] * len(input_gradient)

    # Concatenate the gradients into a single tensor
    grad_tensor = torch.cat([grad.flatten() for grad in input_gradient])
    grad_min = grad_tensor.min().item()
    grad_max = grad_tensor.max().item()

    # Calculate the bin width based on the number of bits
    bin_width = (grad_max - grad_min) / (2 ** num_bits)

    for i in range(len(input_gradient)):
        grad_tensor = input_gradient[i].detach().cpu().numpy()
        quantized_grad = np.round((grad_tensor - grad_min) / bin_width) * bin_width + grad_min
        gradient[i] = torch.Tensor(quantized_grad).to(device)

    return gradient
```

Figure: Added gradient quantization as one of the defense setting

Gradient Quantization Implementation result

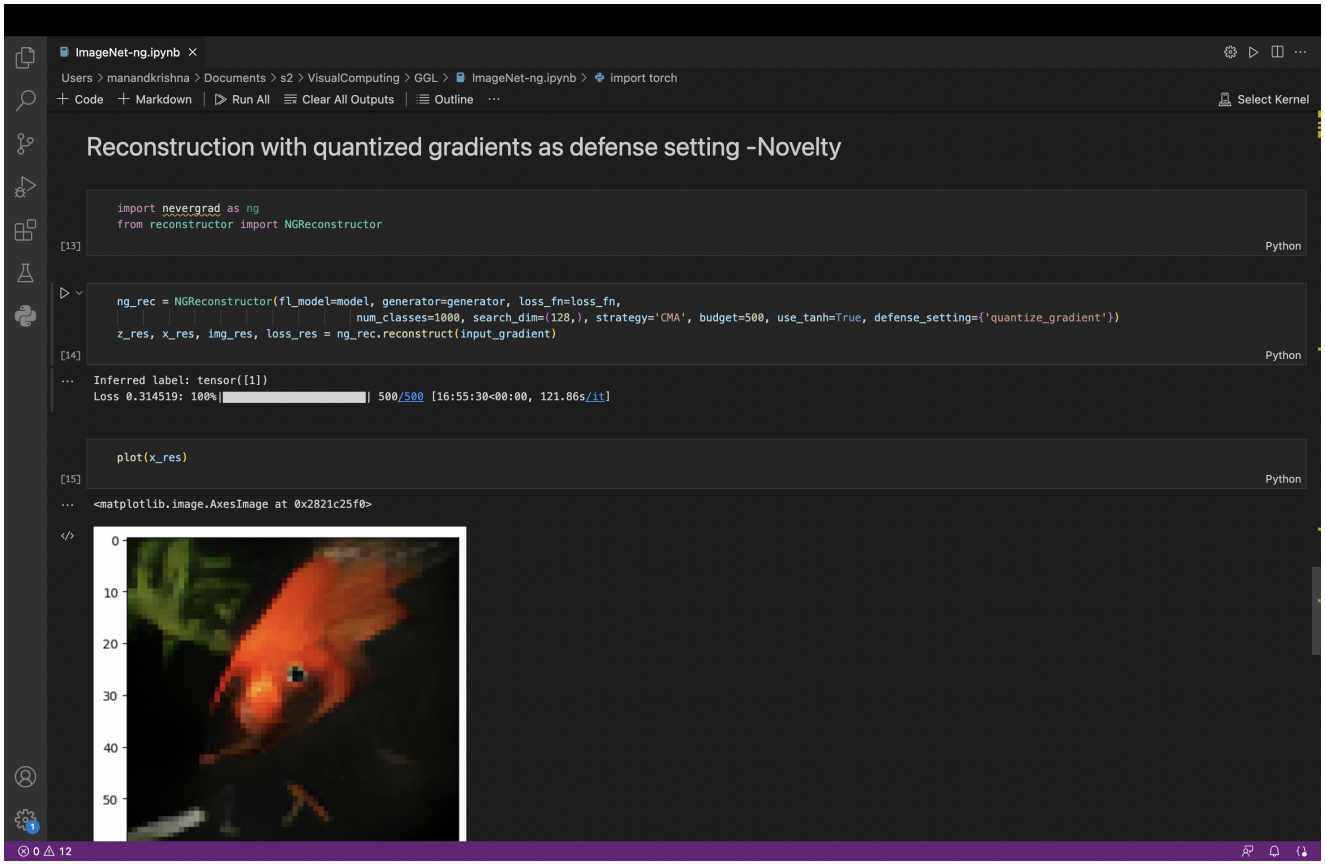


Figure: The Output (Gave an human face as input, but GGL reconstruction gave meaningless output)

Conclusion

- Robustness: GGL is capable of handling various types of perturbations and transformations in gradients, maintaining its effectiveness.
- Privacy Evaluation: By revealing information about the original images, GGL can be utilized to evaluate the strength of existing privacy defenses.
- Defense Improvement: Insights gained from the GGL's performance can inform the design and refinement of more robust privacy protection techniques.

References

Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adria‘ Gasco n

Quotient: two-party secure neural network training and prediction.

Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)

Amir Beck and Marc Teboulle

”Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems”.

IEEE transactions on image processing (2009)

Mordido, Keirsbilck and Keller

Monte Carlo Gradient Quantization

CVPR (2020)

THANK YOU


भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad