

A PROOF FOR PROPOSITION 1

Proposition 4. Consider the gradient inversion problem on the 1-layer neural network and a mini-batch data of N samples. Eq (2) and (3) obtain a zero loss when the generated images $\hat{x} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]$ satisfy:

$$N \cdot \nabla W = \hat{x} \cdot \hat{P}, \quad (5)$$

where \hat{P} refers to a matrix defined by the prediction probabilities $\{\hat{p}_{i,j}\}$ and the one-hot encoding for the labels y^* .

Proof: Consider a 1-layer neural network of weight $W = [w_1, w_2, \dots, w_C] \in \mathbb{R}^{D \times C}$.

If only one sample $\hat{x} \in \mathbb{R}^{D \times 1}$ is generated, the linear layer outputs $\hat{a} = w^T \hat{x} = [w_1^T \hat{x}, w_2^T \hat{x}, \dots, w_C^T \hat{x}]^T \in \mathbb{R}^{C \times 1}$ and the softmax probability is computed as

$$\hat{p}_k = \frac{e^{\hat{a}_k}}{\sum e^{\hat{a}_j}}.$$

Compute gradient w.r.t. each w_k :

1. If $k \neq y$, then

$$\begin{aligned} \frac{\partial \ell}{\partial w_k} &= \frac{\partial \ell}{\partial \hat{p}_y} \cdot \frac{\partial \hat{p}_y}{\partial \hat{a}_k} \cdot \frac{\partial \hat{a}_k}{\partial w_k} \\ &= \frac{1}{\hat{p}_y} \cdot \frac{e^{\hat{a}_y}}{(\sum e^{\hat{a}_j})^2} \cdot e^{\hat{a}_k} \cdot \hat{x} \\ &= \frac{e^{\hat{a}_k}}{\sum e^{\hat{a}_j}} \cdot \hat{x} \\ &= \hat{p}_k \cdot \hat{x} \end{aligned}$$

2. If $k = y$, then

$$\begin{aligned} \frac{\partial \ell}{\partial w_y} &= \frac{\partial \ell}{\partial \hat{p}_y} \cdot \frac{\partial \hat{p}_y}{\partial \hat{a}_y} \cdot \frac{\partial \hat{a}_y}{\partial w_y} \\ &= -\frac{1}{\hat{p}_y} \cdot \frac{e^{\hat{a}_y} \cdot \sum e^{\hat{a}_j} - e^{\hat{a}_y} e^{\hat{a}_y}}{(\sum e^{\hat{a}_j})^2} \cdot \hat{x} \\ &= -\frac{\sum e^{\hat{a}_j} - e^{\hat{a}_y}}{\sum e^{\hat{a}_j}} \cdot \hat{x} \\ &= (\hat{a}_y - 1) \cdot \hat{x} \end{aligned}$$

Consider one-hot encoding for $y^* = [y_1, y_2, \dots, y_C]$, then the above two cases can be unified into

$$\frac{\partial \ell}{\partial w_k} = (\hat{p}_k - y_k) \cdot \hat{x}.$$

For a mini-batch of N samples $\hat{x} = [\hat{x}_1, \hat{x}_2, \dots, \hat{x}_N]$, note

$$\nabla w_i := \frac{1}{N} \sum_{j=1}^N \frac{\partial \ell_j}{\partial w_i} = \frac{1}{N} \sum_{j=1}^N (\hat{p}_{j,i} - y_{j,i}) \cdot \hat{x}_j$$

Rearranging the gradient leads us to

$$\begin{aligned} N \cdot \nabla W &= [N \nabla w_1 \quad N \nabla w_2 \quad \dots \quad N \nabla w_C] \\ &= [\hat{x}_1 \quad \hat{x}_2 \quad \dots \quad \hat{x}_N] \cdot \begin{bmatrix} \hat{p}_{1,1} - y_{1,1} & \hat{p}_{1,2} - y_{1,2} & \dots & \hat{p}_{1,C} - y_{1,C} \\ \hat{p}_{2,1} - y_{2,1} & \hat{p}_{2,2} - y_{2,2} & \dots & \hat{p}_{2,C} - y_{2,C} \\ \dots & \dots & \dots & \dots \\ \hat{p}_{N,1} - y_{N,1} & \hat{p}_{N,2} - y_{N,2} & \dots & \hat{p}_{N,C} - y_{N,C} \end{bmatrix} \\ &= \hat{x} \cdot \hat{P} \end{aligned}$$

This concludes the proof. \square

B DETAILS OF CI-NET

We present details of CI-Net in this part, including how we tailor the original network and gradient inversion algorithm with CI-Net.

B.1 TAILORING PROGRESSIVE MODEL

We illustrate our motivations to tailor the original model in ProGAN Karras et al. (2017) as follows. This generative model is designed to generate high-fidelity figures, but we observe the a few drawbacks when applying it directly to the gradient inversion problem. Figure 8 provides a simple example when applying the linear interpolation (blue line) in the original paper to the gradient inversion problem, when compared with the proposed nearest interpolation method in this paper. Moreover, we occasionally observe such a linear-interpolation could lead algorithm to failure in certain scenarios, especially when the parameters are not sufficiently large for CI-Net. As such, the original model should be tailored to better fit the gradient inversion problem itself.

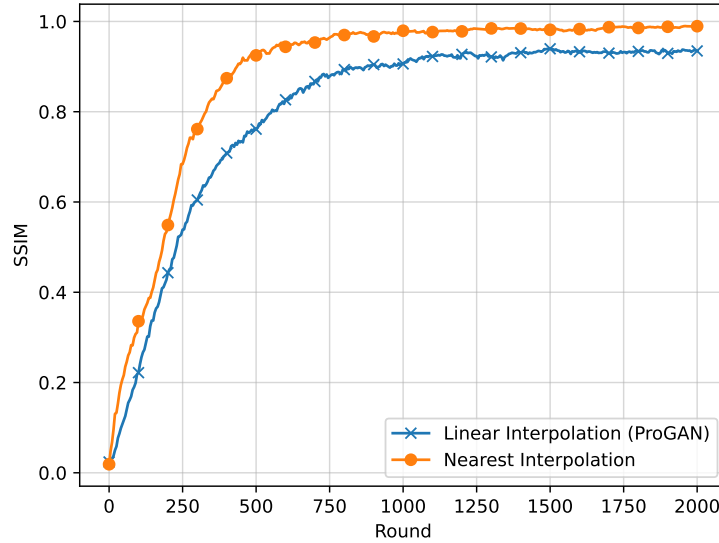


Figure 8: Image reconstruction with different interpolation methods.

Specifically, we make the following changes:

1. The original pixel-norm is shown to hurt the performance, and we replace it with the spectral normalization Miyato et al. (2018).
2. The original linear-interpolation could lead experiments to failure in certain scenarios. We replace it with nearest-interpolation to boost pixel similarities.
3. The two To-RGB layers are now replaced with a direct output layer, for simplicity. Similarly, Resnet-block is also no-longer used in our network.
4. We set the channel number to be a hyper-parameter, in order to allow over-parameterization.
5. ReLU layer is now replaced with LeakyReLU as the latter shows better performance.

B.2 ARCHITECTURES OF CI-NET

After tailoring, the architecture of our CI-Net is now plotted in Figure 9. In general, we first generate a random latent vector z_0 as the input for our convolutional network. Similar to most GAN works, the choice of z_0 is rather arbitrary and we do not need to choose a specific vector. Such a linear latent vector then goes through the “Linear to Conv” layer, followed by a series of convolutional blocks.

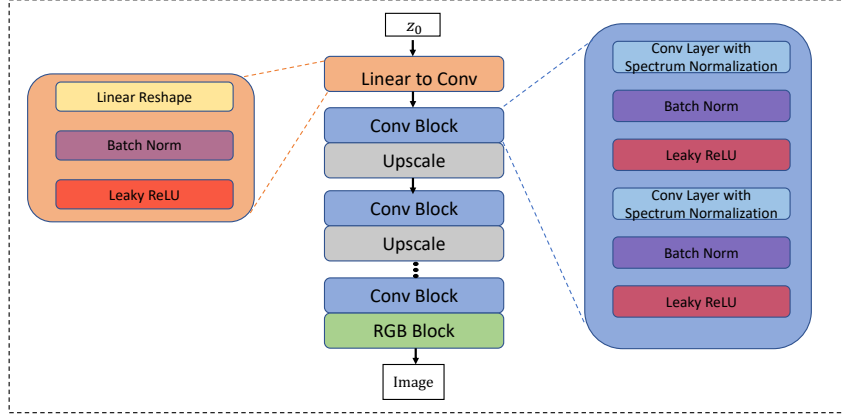


Figure 9: Architecture of CI-Net

B.3 ALGORITHM

The complete algorithm is depicted in Alg 1. Note the keys steps are: 1) we require the system to compute $P(F)$ in order to require over-parameterization in the later step; 2) A gradient descent is replaced with a signed counterpart, since we observe the gradient matching loss is relatively small in most cases.

Algorithm 1 Gradient Inversion with CI-Net

Input: Differential machine learning model F and uploaded gradient ∇W , learning rate η

- 1: Set $P(F) = P(\nabla W)$ ▷ Obtain parameter number
- 2: Load CI-Net $G(\theta)$ and select a channel number so that $P(G) > P(F)$
- 3: Obtain y^* from ∇W
- 4: $\theta_0 \leftarrow \mathcal{N}(0, 1)$ ▷ Set initial values for CI-Net
- 5: **for** $t \leftarrow 0$ to $T - 1$ **do**
- 6: Generate images $\hat{x}_t = G(z_0, \theta_t)$ ▷ Generate fake images
- 7: Obtain gradient $\nabla_W L(\hat{x}_t, y^*)$
- 8: Compute $L_{\text{grad}}(\hat{x}, W, \nabla W) = \|\nabla_W L(\hat{x}_t, y^*) - \nabla_W L(x^*, y^*)\|^2$
- 9: Compute $\nabla_{\theta} L_{\text{grad}}$
- 10: $\nabla \theta_t = \text{sign}(\nabla_{\theta} L_{\text{grad}})$ ▷ Replace gradient with signed gradient
- 11: Update parameters $\theta_{t+1} = \theta_t - \eta \nabla \theta_t$ ▷ Update parameters
- 12: **end for**

Output: Generated Images $\hat{x} = G(z_0, \theta_T)$

C CIFAR-10 ADDITIONAL EXPERIMENT

C.1 BATCH SIZE=64

For validation purposes, we also numerically test the performance of GIAS and the proposed method by setting batch size to 64. Table 4 reports the experimental results by showing four similarity metrics. Unlike the case of $bs=128$, here the GIAS obtains satisfactory performance by obtaining a mean SSIM value of 0.87 and a mean LPIPS value of 0.04. These values reveal the fact that this pioneering work can partially reveal the groundtruth when the batch size is not too large. But yet, the proposed method in this paper still shows advantages by obtaining better performance in all four metrics. Specifically, it reaches 1.00 SSIM value, indicating the reconstructed images are almost identical to the groundtruth.

Algorithm	SSIM \uparrow	FSIM \uparrow	PSNR \uparrow	LPIPS (VGG) \downarrow
GIAS Jeon et al. (2021)	0.87 ± 0.01	0.94 ± 0.01	16.06 ± 0.38	0.04 ± 0.01
Ours	1.00 ± 0.00	1.00 ± 0.00	33.72 ± 0.05	0.01 ± 0.00

Table 4: Algorithm performance of gradient inversion on 64 CIFAR-10 images.

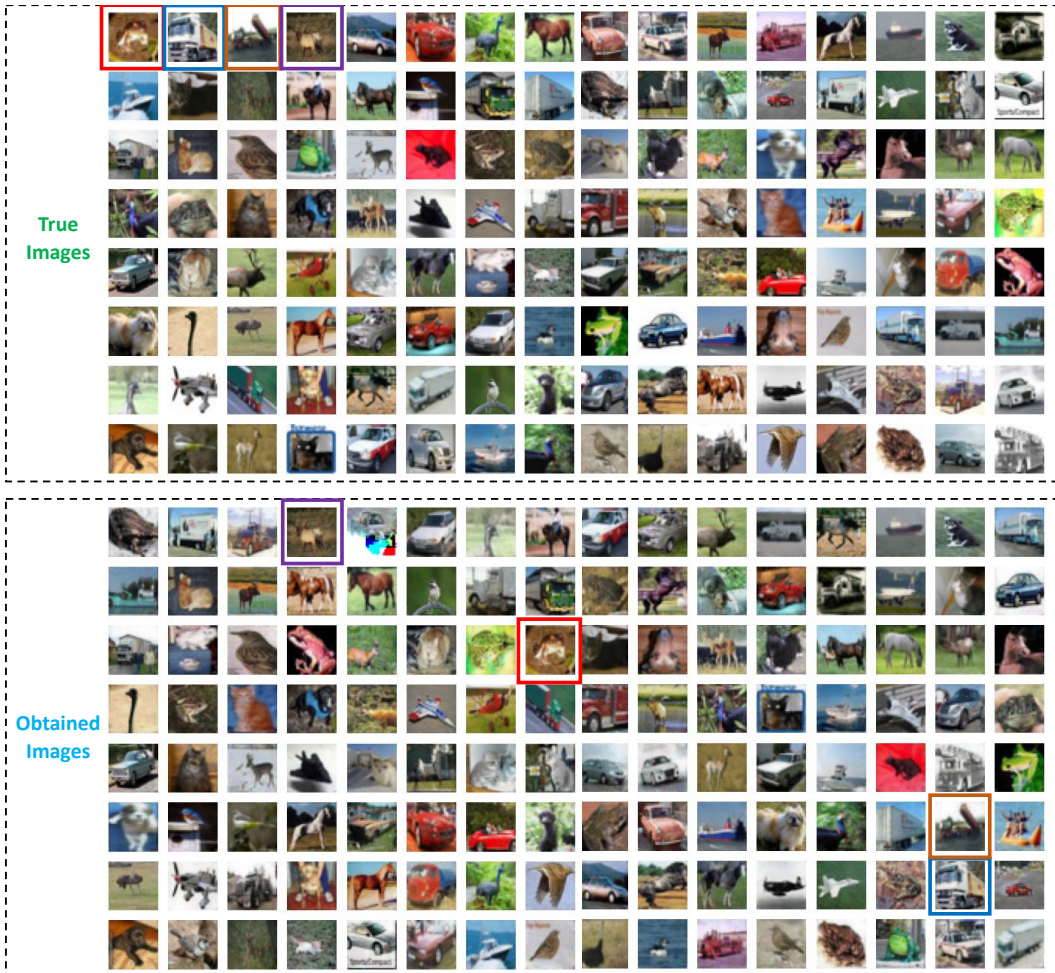


Figure 10: 128 Groundtruth images and the obtained results by CI-Net. Bounding boxes for the first 4 images are plotted for better visualization.

C.2 BATCH SIZE=128

In Figure 6, we sample 12 figures from the 128 obtained images in CI-Net for better visualization. For completeness, all images are plotted in Figure 10, with bounding boxes provided for the first 4 images. In general, these obtained images are highly similar to the groundtruth, hence could be utilized to reveal the hidden true data.

C.3 BATCH SIZE=256

We also numerically test the case when the batch size equals to 256, and this is the maximum batch size our GPU can support. Conventional methods already fail when $bs=128$, hence here we shall only report the performance of CI-Net after training it for 150k rounds.

Results in Figure 11 indicate that even for a larger size batch size, the proposed method can still reconstruct the true images with high fidelities. Note local training in FL would not use such a big batch size for training, but the server may aggregate local gradients into an averaged value, which acts as a proxy for a very large batch of images. These results address the concerns that a third-party may still be able to reconstruct the images from this averaged gradient from the server. Using a large batch-size may not provide a safety guarantee in federated learning.

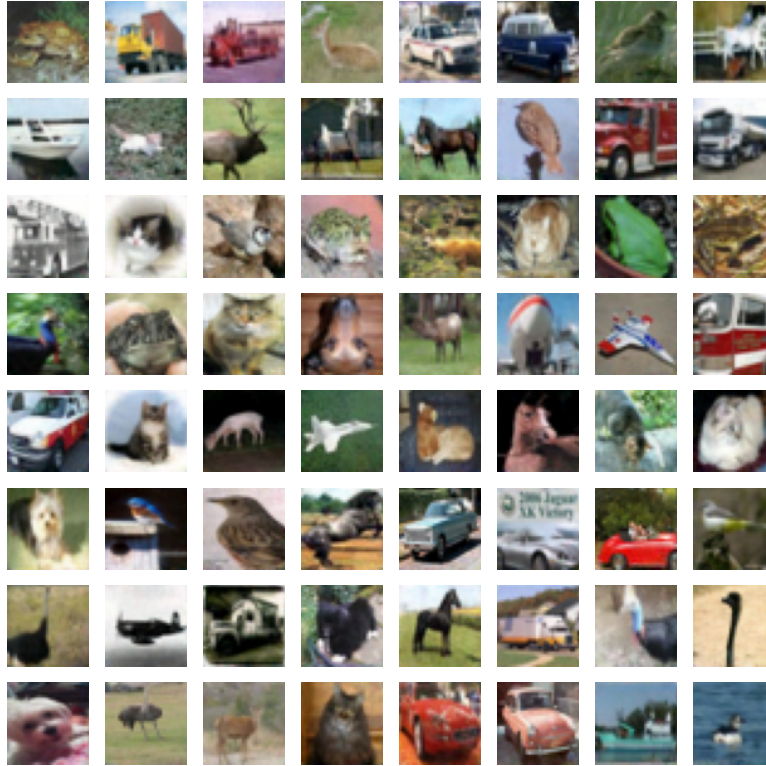


Figure 11: The first 64 reconstructed images from CI-Net, when batch size equals 256.

Algorithm	SSIM \uparrow	FSIM \uparrow	PSNR \uparrow	LPIPS (VGG) \downarrow
Ours	0.98 ± 0.01	0.99 ± 0.01	34.11 ± 0.13	0.02 ± 0.01

Table 5: Algorithm performance of gradient inversion on 256 CIFAR-10 images.

D IMAGENET ADDITIONAL EXPERIMENT

D.1 BATCH SIZE=24

For completeness, we plot the complete 24 reconstructed images in Figure 12, when using CI-Net on ImageNet. The groundtruth images are plotted on the left for visual comparison.

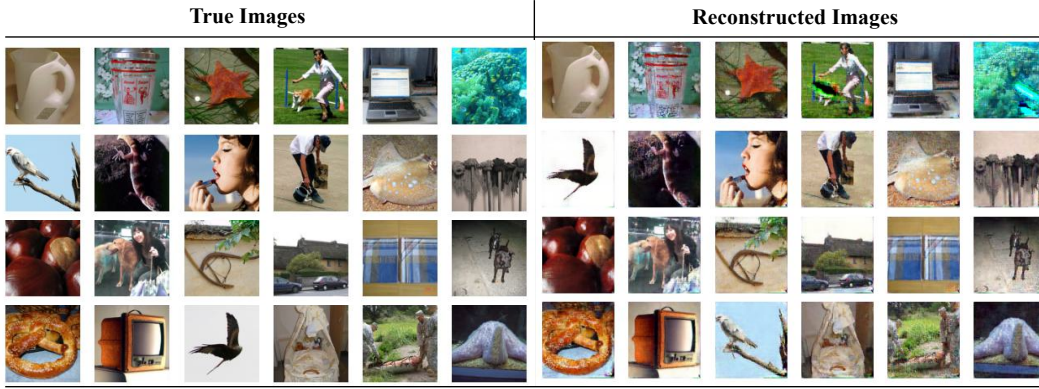


Figure 12: The groundtruth images and reconstructed images from CI-Net. Batch size=24.

D.2 BATCH SIZE=32

We further extend our experiment to the case of 32 images on ImageNet. Note such a large batch size and an over-parameterized network consumes 39.6 GB memories, reaching the limit of our hardware device (NVIDIA-A100, 40GB).

Results are consistent with our previous demonstrations when bs equals 24. Potential attackers can find out the groundtruth from these reconstructed images, also some of them may be slightly blurred when compared with the groundtruth.

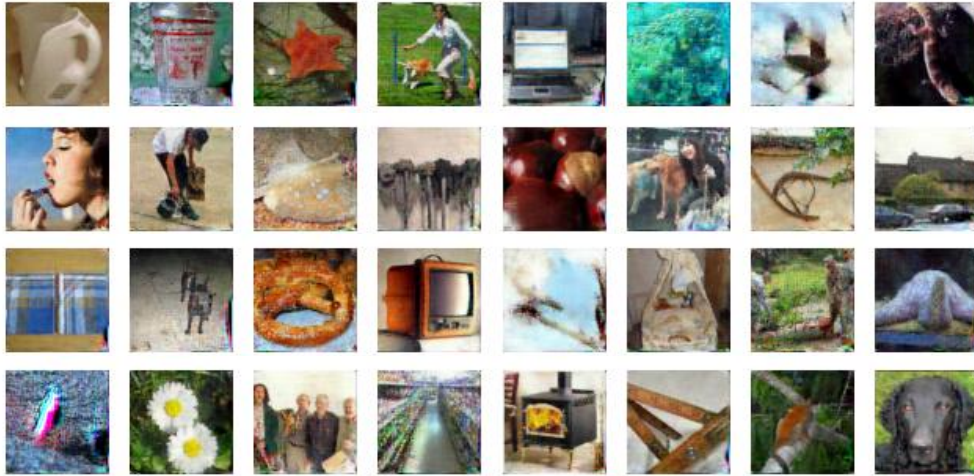


Figure 13: Gradient inversion with CI-Net on ImageNet. Batch size=32.

D.3 DISCUSSIONS ON THE PROPOSED METHOD

During these experiments, we also notice the additional convolutional model G requests extra memory costs, especially with the new over-parameterized requirement. This can be a potential drawback of our method.