GRNN: Generative Regression Neural Network - A Data Leakage Attack for Federated Learning

Hanchi Ren, Jingjing Deng and Xianghua Xie

Computer Vision and Machine Learning Group
Department of Computer Science
Swansea University, United Kingdom
`csvision.swansea.ac.uk`

Abstract

Data privacy has become an increasingly important issue in Machine Learning (ML), where many approaches have been developed to tackle this challenge, *e.g.* cryptography (Homomorphic Encryption (HE), Differential Privacy (DP), *etc.*) and collaborative training (Secure Multi-Party Computation (MPC), Distributed Learning and Federated Learning (FL)). These techniques have a particular focus on data encryption or secure local computation. They transfer the intermediate information to the third party to compute the final result. Gradient exchanging is commonly considered to be a secure way of training a robust model collaboratively in Deep Learning (DL). However, recent researches have demonstrated that sensitive information can be recovered from the shared gradient. Generative Adversarial Network (GAN), in particular, has shown to be effective in recovering such information. However, GAN based techniques require additional information, such as class labels which are generally unavailable for privacy-preserved learning. In this paper, we show that, in the FL system, image-based privacy data can be easily recovered in full from the shared gradient only via our proposed Generative Regression Neural Network (GRNN). We formulate the attack to be a regression problem and optimize two branches of the generative model by minimizing the distance between gradients. We evaluate our method on several image classification tasks. The results illustrate that our proposed GRNN outperforms state-of-the-art methods with better stability, stronger robustness, and higher accuracy. It also has no convergence requirement to the global FL model. Moreover, we demonstrate information leakage using face re-identification. Some defense strategies are also discussed in this work.

1 Introduction

More often than not, the success of Deep Learning (DL) [1, 2] relies on the availability of a large quantity of data. A centralized learning scheme with cloud-based distributed computing system is commonly used to speed up the training and scale up to larger datasets [3, 4, 5, 6, 7]. However, due to data protection and privacy requirements such systems are generally infeasible as they require centralized data for training. For instance, data owners (*e.g.* hospital, finance company, or government agent) are not willing or not able to share private data with algorithm and computing platform providers, which causes the so-called "Data Islands" issue. Therefore, decentralized training approaches with data privacy protection are more attractive.

Federated Learning (FL) [8] was proposed to jointly train a model without directly accessing the private training data. Instead of sharing data, the aggregation server shares a global model and requires the individual data owners to expose the gradient information computed privately only. The gradient is generally considered to be secure to share. However, recent studies found that the gradient-sharing scheme is in fact not privacy-preserved [9, 10, 11, 12]. For example, the presence of a specific property of the training dataset can be identified given the gradient information [12]. Hitaj *et al.* [11] showed the feasibility of generating images that are similar to training images using Generative Adversarial Network (GAN) given any known class label [13]. Deep Leakage from Gradients (DLG) based models [14, 15] were proposed to approximate the leaked gradient via learning the input while fixing the model weights. However, they are usually unstable and sensitive to the size of the training batch and the resolution of input image. Furthermore, Geiping *et al.* [16] discussed the theoretical aspect of inverting gradient to its corresponding training data and used magnitude-invariant cosine similarity loss function in proposed Inverting Gradient (IG), which is capable of recovering high-resolution images (*i.e.* 224*224) with a large number of training batch (*i.e.* $\#Batch = 100$). However, we find that the success rate is relatively low.

GAN is a generative model proposed by Goodfellow *et al.* [13] for image generation. Inspired by the GAN and DLG model, we introduce a gradient guided image generation strategy that properly addresses the stability and data quality issues of DLG based methods. The proposed Generative Regression Neural Network (GRNN), a novel data leakage attack method is capable of recover private training image up to a resolution of 256*256 and a batch size of 256. The method is particularly suitable for FL as the local gradient and global model are readily available in the system setting. GRNN consists of two branches for generating fake training data and corresponding label. It is trained in an end-to-end fashion by approximating the fake gradient that is calculated by the generated data and label to the true gradient given the global model. Mean Square Error (MSE), Wasserstein Distance (WD) and Total Variation Loss (TVLoss) are used jointly to evaluated divergence between true and fake gradients. We empirically evaluate the performance of our method on several image classification tasks, and comprehensively compared against the state-of-the-art. The experimental results confirm that the proposed method is much more stable and capable of producing image with better quality when a large batch size and resolution are used. The contributions of this work are five-folds:

- We propose a novel method of data leakage attack for FL, which is capable of recovering private training image up to a resolution of 256*256, a batch size of 256 as well as the corresponding labels from the shared gradient.
- We conduct comprehensive evaluation, where both qualitative and quantitative results are presented to prove the effectiveness of GRNN. We also compare the proposed method against the state-of-the-art, which shows that GRNN is superior in terms of success rate of attack, the fidelity of recovered data and the accuracy of label inference. In addition, our method is much more stable than others with respect to the size of the training batch and input resolution.
- We conduct a face re-identification experiment that shows using the image generated by the proposed GRNN can achieve higher *Top-1*, *Top-3* and *Top-5* accuracies compared to the state-of-the-art.
- We discuss the potential defense strategies and quantitatively evaluate the effectiveness of our method against noise addition defense strategy.
- The implementation of the method is publicly available to ensure its reproducibility.

The rest of the paper is organized as follows: An overview of related works on data leakage attack are presented in Section 2. The proposed method is described in Section 3. The details of experimental results, discussions, and potential defense strategies are provided in Section 4. Section 5 concludes the paper.

2 Related Work

Given a trained victim model and a target label, Hitaj *et al.* [11] proposed a GAN based data recovery method that can generate a set of new data having close distribution to the training dataset. First, each participant trains a local model on its own dataset for several iterations to achieve an accuracy above the pre-set threshold which is used as a discriminator in the next stage. To train the generator, the weights of the discriminator are firstly fixed. Then, given a specified class, the generator is learned via producing an image that maximizes the classification confidence of the discriminator. The generated image has no explicit correspondence to the training data, and the method is sensitive to the variance of training data [12].

DLG proposed by Zhu *et al.* [14] is one of the first works investigating the training data leakage in collaborative learning. It formulates the image recovery as a regression problem as follows: A batch of randomly initialized "dummy" images and labels are used to compute "dummy" gradient via a forward-backward pass on the global model. The true local gradient is readily available in the training system. DLG updates the "dummy" images and labels via minimizing the MSE distance between the "dummy" gradient and the true gradient. It can achieve exact pixel-wise data recovering without additional information other than the shared global model and local gradient. Zhao *et al.* [15] introduced Improved DLG (iDLG) that addresses the divergence and inconsistent label inference issues of DLG. They found that the derivative value corresponding to the ground-truth label drops in the range of [-1, 0], and other cases lie in the range of [0, 1]. It is then feasible to identify the correct label in a such naïve way. In addition to the low accuracy of label inference, DLG based methods often fail to recover the image from the gradient when the variance of the data is large, which is very common for the dataset that has a large number of classes. IG [16] improved the stability of DLG and iDLG by introducing magnitude-invariant cosine similarity measurement for loss function, namely Cosine Distance (CD). It aims to find images that pursue similar prediction changes from the classification model rather than those that can generate the close values with the shared gradient. It shows recognizable results

Table 1: Comparison of GRNN with the closely related works.

Method	Recovery Mode	#Batch	Resolution	Loss Function
DLG[14]	Discriminative	Small, 8	Low 64*64	MSE
iDLG[15]	Discriminative	Small, 1 only	Low 64*64	MSE
IG[16]	Discriminative	Medium, 100	High 224*224	CD & TVLoss
GRNN	Generative	Large, 256	High 256*256	MSE & WD & TVLoss

of recovering high-resolution images (*i.e.* 224*224) with a large number of training batch (*i.e.* $\#Batch = 100$). Compared to the closely related works [14, 15, 16], our proposed method takes a generative approach having higher stability for recovering high-resolution images (*i.e.* up to 256*256) with a large batch size (*i.e.* $\#Batch = 256$). Table 1 presents the key differences between DLG, iDLG IG and proposed GRNN.

Cryptography based privacy-preserving methods, such as secure aggregation protocol [17], Homomorphic Encryption (HE) [18, 19], Differential Privacy (DP) [20, 21, 22] have been developed for privacy-preserving learning, for instance, linear regression model [23, 24], decision trees [25, 26], deep neural networks [19, 27, 28]. However, the cryptography operations are computationally expensive, and the consistency of visual patterns of images is generally not guaranteed in encrypted data format, which usually leads to a learning model with poor generalization ability. Gradient encryption is an alternative defense strategy [29, 30, 31], where HE is used to encrypt the gradient shared between server and clients, therefore, the aggregation can be done directly on the encrypted gradient. In Section 4.5, we discuss the potential defense strategies and quantitatively evaluate the effectiveness of proposed GRNN against noise addition.

3 Proposed Method

There are three major challenges that have not been well addressed by existing data recovery methods as follows: model stability, the feasibility of recovering data from large batch size, and fidelity with high-resolution. Both [14, 15, 16] treat the data recovery as a high-dimensional fitting problem driven by the gradient regression objective. The state-of-the-art models are fairly sensitive to the data initialization, easily fail to capture the individual image characteristic when a large number of gradients are aggregated (*i.e.* Federated Averaging (FedAvg)), and hardly maintain the natural image structures in detail when resolution is increased. In this paper, we propose GRNN model which formulates the data recovery task as a data generation problem that is guided by the gradient information. We introduce a GAN model as the image data generator and a simply Fully-Connected (FC) layer as a label data generator, where the so-called fake gradient can then be computed given the shared global model. By jointly optimizing both two generators to approximate the true gradient, GRNN can well align the latent space of GAN with the gradient space of the shared global model, furthermore, generate the training data with high fidelity stably. Therefore, GRNN is able to recover the data from the local gradient shared between client and server in a FL setting, which can also be used for malicious purposes, such as stealing private data from a client. In this section, we will first introduce a neural network model based FL system in Section 3.1, then present the proposed GRNN method in Section 3.2.

3.1 Federated Learning

FL is a distributed collaborative training scheme that consists of multiple clients and one parameter server. The gradients calculated restrictively on client nodes are aggregated on the server node using fusion functions [8, 32]. We use the classic FedAvg method to train deep neural network over multiple parties which runs a number of steps of Stochastic Gradient Descent (SGD) in parallel on client node and then averages the resulting model updates via a central server periodically. The global model is merged by taking the average of gradients from local models according to $\omega\prime = \sum_i^N \frac{1}{N}\omega_i$, where $\omega\prime$ and ω_i are the gradients of global model and the i_{th} local model, and N is the total number of the clients.

We learn a Convolutional Neural Network (CNN) based image classification model, whereas FedAvg is applicable to any finite-sum objective. Formally, at iteration t, the i_{th} ($i \in \{1, 2, ..., C\}$) client computes the CNN model θ_t and the local gradient g_t^i based on its local training data (x_t^i, y_t^i) (see Equ. 1). $\mathcal{F}(\bullet)$, θ_t and $\mathcal{L}(\bullet)$ are the global learning model, network parameters at iteration t and loss function respectively. The local gradient g_t^i is calculated using typical SGD at client node independently. The server aggregates the local gradient g_t^i and then updates the global model weights θ_{t+1}, as shown in Equ. 2 where C is the number of clients.

$$g_t^i = \frac{\partial \mathcal{L}(\mathcal{F}(< x_t^i, y_t^i >, \theta_t))}{\partial \theta_t} \tag{1}$$

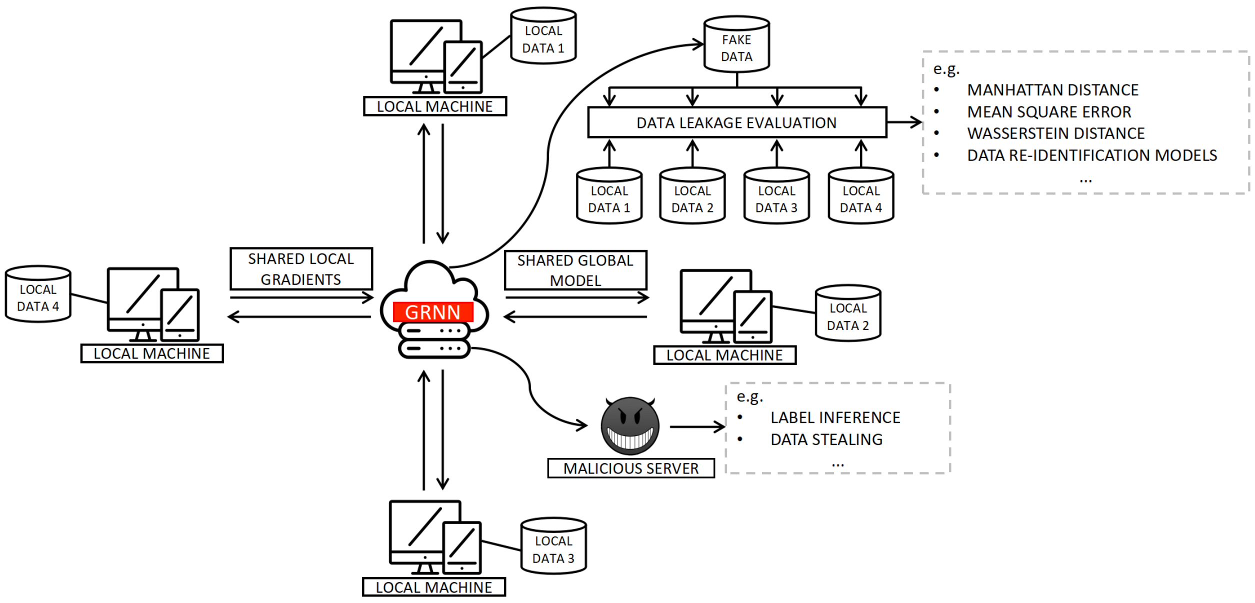


Figure 1: Illustration of how GRNN as a malicious server is deployed in a FL system.

$$\theta_{t+1} = \theta_t - \frac{1}{C}\sum_{i=1}^{C} g_i \tag{2}$$

3.2 GRNN: Data Leakage Attack

The proposed GRNN is deployed at the central server as illustrated in Fig. 1. The GRNN has access to the neural network architecture and parameters of the global model. The only information that can be obtained between server and client is the gradient calculated based on the current global model. To compute the gradient of the model, the private data, corresponding label, and the global model at current iteration are required at the local client. We hypothesize that the shared gradient contains the information of private data distribution and space partitioning given the global model. The architecture of GRNN shown in Fig. 2 has two branches with the same input that is sampled from a common latent space. We consider that each point within this latent space represents a pair of image data and its corresponding label. The objective of the GRNN is to separate this tied representation in the latent space and recover the local image data and corresponding label by approximating the shared gradient that is accessible at the server node.

The top branch is used to recover image data, namely fake-data generator, which consists of a generative network model. We followed the design principle of iWGAN [33] where the image is generated from a coarse scale to a fine-scale gradually. The concept is consistent with a reasonable intuition, where the image is drawn from sketch then the details are gradually added. The input random vector is first fed into a fractionally-strided convolutional layer [34, 35] to produce a set of feature maps with a resolution of 4*4, which then goes through several upsampling blocks that gradually increase the spatial resolution. The number of the upsampling blocks is determined by the resolution of the target image. For example, if the target image resolution is 32*32, then 3 upsampling blocks ($4 \rightarrow 8 \rightarrow 16 \rightarrow 32$) are used. We also investigate the large resolution of the input image in Section 4.2. In upsampling block, nearest-neighbor interpolation is used to recover the spatial resolution of feature maps from the previous layer. It then passes through a standard convolutional sub-block for rectifying the detail feature representation, which contains a convolutional layer, a batch normalization layer, and a Gated Linear Unit (GLU) [36] activation layer. Empirically, we found GLU is far more stable than *ReLU* and can learn faster than *Sigmoid*. Therefore, we adopted GLU as the activation function for our model while the learning strategy using GAN driven by a regression objective is the key novelty. The proposed method also works with other activation functions, such as *ReLU*. The details of the upsampling block are listed in Table 2. At the end of the top branch, a data sample that has the same dimension as the training input of the FL system is generated.

The bottom branch is used to recover label data, namely fake-label generator which contains a FC layer followed by a softmax layer for classification. It takes a randomly sampled vector from latent space as input and outputs its

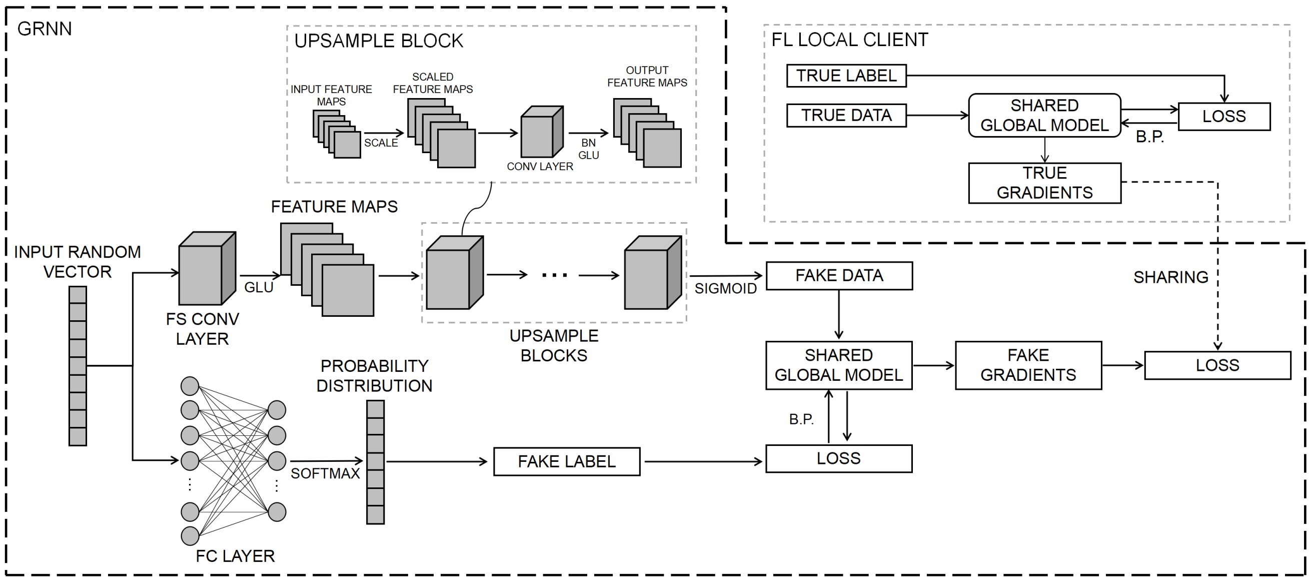


Figure 2: Details of the proposed GRNN where the top branch is for generating the fake image and the bottom branch is for inferring the label. "FC LAYER" is fully-connected layer. "FS CONV LAYER" is fractionally-strided convolutional layer. For details of upsampling block, please refer to Table 2.

Table 2: Construction of upsampling block.

Layer Name	Setting
Upsampling Layer	scale factor: 2 mode: nearest
Convolutional Layer	kernel size: 3 stride: 1 padding: 1
Batch Normalization Layer	-
Gated Liner Unit	-

corresponding fake label. We assume the elements in the input vector are independent of each other and subject to a standard Gaussian distribution. The label set in GRNN is identical to the one used in the FL system. Formally, the fake image ($\hat{x}_t^j$ and fake label $\hat{y}_t^j$) data generation can be formulated as in Equ. 3, where $\hat{\theta}$ and v_t are trainable parameters of GRNN and input random vector is sampled from a unit Gaussian distribution.

$$(\hat{x}_t^j, \hat{y}_t^j) = \mathcal{G}(v_t|\hat{\theta}_t) \tag{3}$$

Given a pair of fake image and label that are generated as described above, a fake gradient on the current global model can be obtained via feeding them as training input and performing one iteration of SGD according to Equ. 1. The objective of GRNN is to approximate the true gradient, therefore, the whole model can be trained by minimizing the distance between the fake gradient $\hat{g}_t^j$ and shared true gradient g_t^j, *i.e.* the most commonly used loss MSE is adopted here:

$$\arg\min_{\hat{\theta}} ||g_t^j - \hat{g}_t^j||^2 \Longrightarrow \arg\min_{\hat{\theta}} ||\frac{\partial \mathcal{L}(\mathcal{F}(< x_t^j, y_t^j >, \theta_t))}{\partial \theta_t} - \frac{\partial \mathcal{L}(\mathcal{F}(< \hat{x}_t^j, \hat{y}_t^j >, \theta_t))}{\partial \theta_t}||^2$$

Note that the gradient of the model is a vector, the length of which is equal to the number of trainable parameters. In addition to measuring discrepancy between the true and fake gradients based on Euclidean distance, we also introduce the WD [37] loss to minimize the geometric difference between two gradient vectors and TVLoss [38] to impose the smoothness constrain on generated fake image data. Therefore, the loss function for GRNN, namely $\hat{\mathcal{L}}(\bullet)$ is formulated as:

$$\hat{\mathcal{L}}(g, \hat{g}, \hat{x}) = MSE(g, \hat{g}) + WD(g, \hat{g}) + \alpha \cdot TVLoss(\hat{x}) \tag{4}$$

where we weight the MSE loss and WD equally and α is the weighting parameter for smoothness regularization. Both branches of the proposed GRNN are parameterized using a neural network that is completely differentiable and can be jointly trained in an end-to-end fashion. The complete training procedure for GRNN is described in Algorithm 1.

Algorithm 1 GRNN: Data Leakage Attack

```
1: $g_t^j \leftarrow \partial\mathcal{L}(\mathcal{F}(<x_t^j, y_t^j>, \theta_t))/\partial\theta_t$; # Produce true gradient on local client.
2: $v_t \leftarrow$ Sampling from $\mathcal{N}(0,1)$; # initialize random vector inputs for GRNN.
3: for each iteration $i \in [1, 2, ..., I]$ do
4:     $(\hat{x}_{t,i}^j, \hat{y}_{t,i}^j) \leftarrow \mathcal{G}(v_t|\hat{\theta}_i)$; # Generate fake images and labels.
5:     $\hat{g}_{t,i}^j \leftarrow \partial\mathcal{L}(\mathcal{F}(<\hat{x}_{t,i}^j, \hat{y}_{t,i}^j>, \theta_t))/\partial\theta_t$; # Calculate fake gradient on shared global model.
6:     $\mathcal{D}_i \leftarrow \hat{\mathcal{L}}(g_t^j, \hat{g}_{t,i}^j, \hat{x}_{t,i}^j)$; # Calculate GRNN loss between true gradient and fake gradient.
7:     $\hat{\theta}_{i+1} \leftarrow \hat{\theta}_i - \eta(\partial\mathcal{D}_i/\partial\hat{\theta}_i)$; # Update GRNN model.
8: end for
9: return $(\hat{x}_{t,I}^j, \hat{y}_{t,I}^j)$; # Return generated fake images and labels.
```

4 Experiment and Discussion

4.1 Dataset and Experimental Setting

A number of experiments on typical computer vision tasks including digit recognition, image classification, and face recognition were conducted to evaluate our proposed method. We used four public benchmarks, MNIST [39], CIFAR-100 [40], Labeled Faces in the Wild (LFW) [41] and VGG-Face [42] for those tasks. There are 7000 gray-scale handwritten digit images of 28*28 resolution in the MNIST dataset. CIFAR-100 consists of 60000 color images of size 32*32 with 100 categories. LFW is a human face dataset that has 13233 images from 5749 different people. We treated each individual as one class, hence, there are 5749 classes in total. VGG-Face consists of over 2.6 million human face images and 2622 identities.

LeNet [43] and *ResNet-18* [1] are used as the backbone networks for training image classifiers in FL system. All neural network models are implemented using PyTorch [44] framework and the source code has been made publicly available[1] for reproducing the results. We replace all *ReLU* function with *Sigmoid* function in order to ensure the model is second-order differentiable, which is as the same as DLG and iDLG in order to intractably compute the MSE loss. Our method is also applicable to the network with *ReLU* function where an second-order differentiable approximation to the *ReLU* function can be used while the computation is intractable compared to using *Sigmoid* function. The batch size varies across different experiments, and a comprehensive comparison study was carried out. *RMSprop* optimizer with a learning rate of 0.0001 and a momentum of 0.99 is used for GRNN. Regarding the loss function of GRNN, we set the weights of TVLoss to $1e-3$ and $1e-6$ for *LeNet* and *ResNet-18*, respectively.

4.2 Image Recovery

We first trained three CNN based image classifiers over one iteration on MNIST, CIFAR-100, and LFW datasets separately, and then used the proposed GRNN to conduct data leakage attack on those models. In order to demonstrate that our method has no requirement on the convergence of the shared global model and deployment flexibility, we conducted two sets of experiments of the data leakage attack using GRNN. One attack was carried out after the first iteration (non-converged state) and another one was carried out after the change of the loss of shared global model is sufficiently small $\delta G \leq 1e-4$ (converged state).

Some qualitative results are presented in Table 3, where the middle column shows the image data recovered from the true gradient. The images are recovered gradually with the number of iterations for training GRNN increases, while the true gradient is obtained only at the first iteration of global FL model, which indicates the gradient can lead to data leakage in FL regardless the convergence of shared global model. Table 4 shows that image data generated from converged global model performs worse than from non-converged global model. We argue that the reason for lower performance is caused by a large proportion of zero gradients produced by the converged model. We also quantify the quality of generated image using Peak Signal-to-Noise Ratio (PSNR) score, an objective standard for image evaluation which is defined as the logarithm of the ratio of the squared maximum value of RGB image fluctuation over MSE between two images. The formal definition is given as such: $PSNR = 10 \cdot \lg(\frac{255^2}{MSE(img1, img2)})$. The higher

[1] https://github.com/Rand2AI/GRNN

Table 3: Examples of data leakage attack using the proposed GRNN on the global model trained over one iteration.

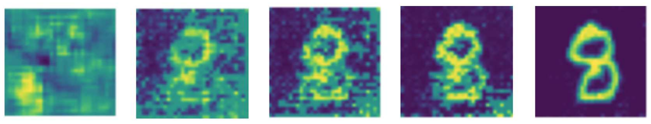
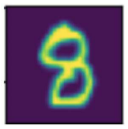
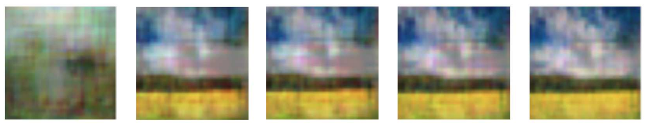
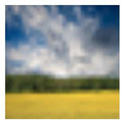
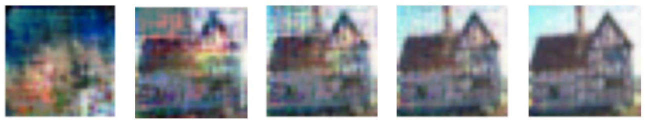
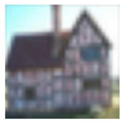
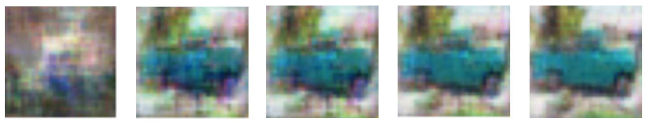
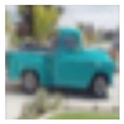
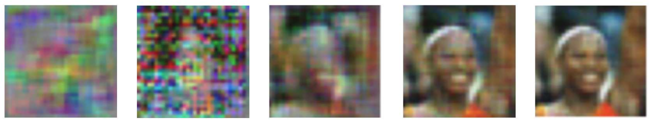
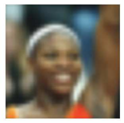
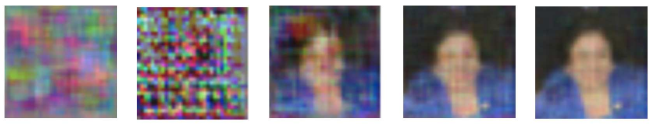
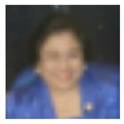
Dataset	**Generated Data**	**True Data**
MNIST		
CIFAR-100		
		
		
LFW		
		

Table 4: Examples of data leakage attack using the proposed GRNN on converged global model.

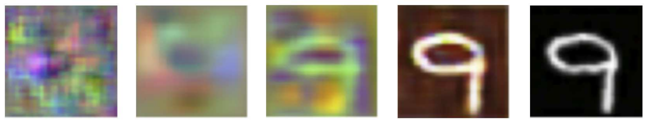
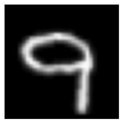
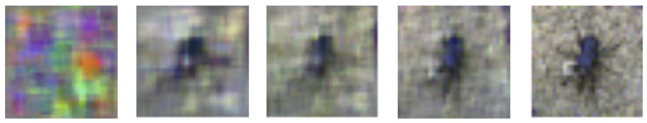
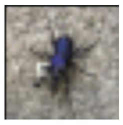
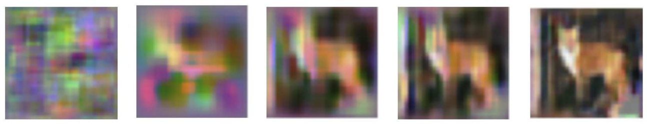
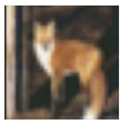
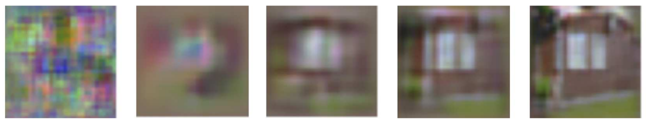
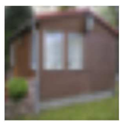
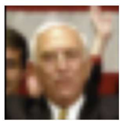
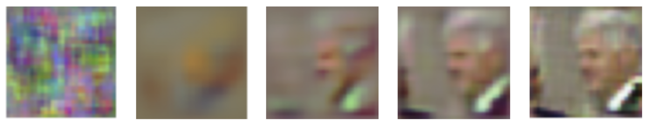
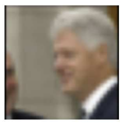
Dataset	**Generated Data**	**True Data**
MNIST		
		
CIFAR-100		
		
LFW		
		

Table 5: Average PSNR scores achieved by DLG and GRNN with the batch size of 1 using non-converged and converged global model. "-" represents that there is no understandable visual image generated.

Method	Model	Dataset	Non-Converged	Converged
DLG	LeNet	MNIST	50.90	-
		CIFAR-100	**48.59**	**40.97**
		LFW	45.05	-
Ours	LeNet	MNIST	**52.37**	39.47
		CIFAR-100	47.58	38.44
		LFW	**45.57**	**38.17**
	ResNet-18	MNIST	42.70	**43.40**
		CIFAR-100	38.85	38.03
		LFW	39.60	38.02

PSNR score, the higher the similarity between two images. Table 5 gives the average PSNR scores achieved by DLG and GRNN with the batch size of 1 using a non-converged and a converged global model. Overall, the non-converged model performs better than the converged model, except using *ResNet-18* on the MNIST dataset, which achieves 0.70 dB less than the converged model (42.70 dB VS. 43.40 dB). DLG failed to recover the training image on MNIST and LFW datasets, as there is no visually recognizable image generated.

To quantitatively evaluate the similarity of recovered images and true images, three metrics, namely MSE, WD and PSNR [45] are computed. Fig. 6 shows the MSE and WD between recovered images and true images with respect to training iteration of the attacking model. *LeNet* model as the global FL model was used in this experiment. The dash lines and the solid lines correspond to the results of DLG and GRNN, respectively. Although GRNN achieves slightly higher scores in MSE and WD when the batch size of 1 is used, our method is much more stable and significantly better when larger batch size is used. When the batch size increases to 4 and 8, DLG only works on MNIST but fails on both CIFAR-100 and LFW, therefore, the corresponding similarity measurements can not converge (see the green and purple dashed lines in Figs. 6 (c) (d), (e) and (f)). DLG fails on all datasets with batch size of 16 while GRNN is able to recover the image data consistently (see Figs. 6 (g) and (h)). We also notice that DLG can well approximate the shared true gradient while generating a poor image. Fig.4 shows that DLG achieves smaller MSE loss (Euclidean distance between true gradient and fake gradient) compared to our method, while it fails to recover the image data.

In Fig. 4, MSE and WD results from GRNN are slightly higher than those from DLG with a batch size of 1, however, the difference between these two generated set of images at this batch size is hardly discernible. Hence, we further calculated PSNR to compare the pixel-wise similarity of the recovered images. Table 5 shows GRNN achieves higher PSNR on MNIST and LFW datasets (+1.45% and +0.52% respectively) and lower on CIFAR-100 dataset (-1.01%) using non-converged global model. Furthermore, our method achieved reasonable PSNR score on attacking *ResNet-18* model while DLG always fails. Table 6 shows some qualitative comparison of recovered images using both methods over different numbers of iterations. We can observe that DLG recover the image pixel by pixel greedily, whereas GRNN also ensures the appearance distribution to be consistent with the true image in a coarser scale, and object details are then gradually filled at a finer scale.

4.2.1 Ablation Study on Batch Size

A comparison study between the proposed GRNN and DLG was carried out using the same setting as described above, while we varied the training batch size for FL model to evaluate the feasibility of data leakage attack. It is reasonable to consider that the data recovery is more challenging when the training batch size is increasing as the shared gradient is averaged over all images data in the batch, where information of an individual image is obscurely mixed. Table 7 lists the success and failure of attack for both two methods. We follow the same principle defined in DLG paper, where a successful attack refers to recovering an image that is visually recognizable. On MNIST dataset, DLG attack starts to fail when the size of training batch is larger than 8 and the *LeNet* is used in FL, whilst our method is able to recover the image data even with a batch size of 256. On CIFAR-100 and LFW datasets, DLG attack only works with a training batch size of 1, however, our method can still successfully perform the attack with a large batch size up to 64 and 128 respectively. Some failure examples of our method when a large training batch size is used can be found in Table 8. Although some failure examples show the consistency of color distribution and geometric similarity of objects, the appearance details are largely inconsistent or hard to match the original ones. We also show that the proposed GRNN can successfully attack complex and large models, such as *ResNet-18*.

Table 6: Comparison of image recovery using DLG and GRNN over different numbers of iterations.

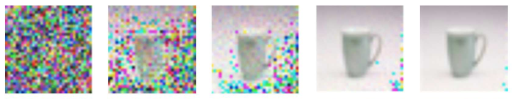
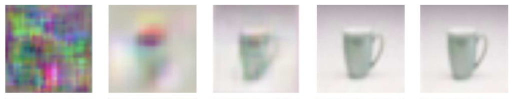
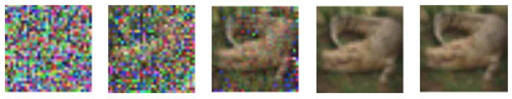
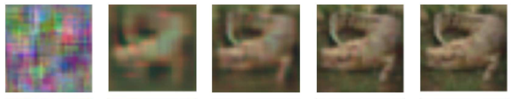
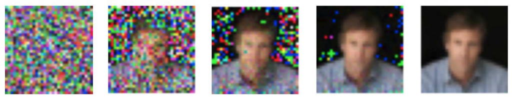
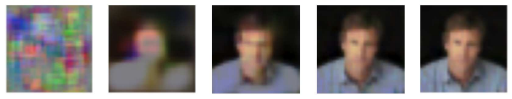
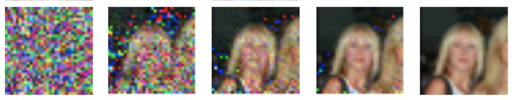
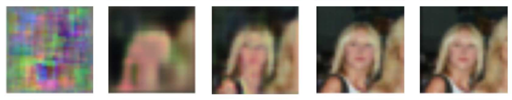
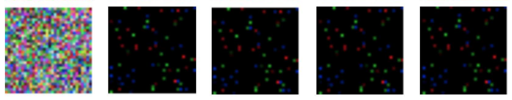
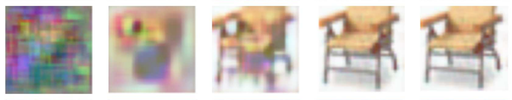
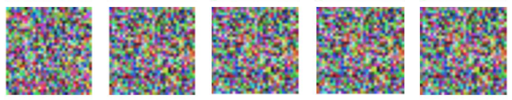
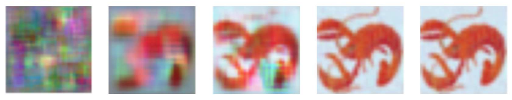
True Data	**DLG**		**GRNN**	
		√		√
		√		√
		√		√
		√		√
		×		√
		×		√

Table 7: Data leakage attack with different training batch sizes for FL model, where "✓" refers to a success and "×" refers to a failure.

Method	**Model**	**#Batch** / **Dataset**	**1**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
DLG	LeNet	MNIST	✓	✓	✓	×	×	×	×	×
		CIFAR-100	✓	×	×	×	×	×	×	×
		LFW	✓	×	×	×	×	×	×	×
Ours	LeNet	MNIST	✓	✓	✓	✓	✓	✓	✓	✓
		CIFAR-100	✓	✓	✓	✓	✓	✓	✓	×
		LFW	✓	✓	✓	✓	✓	✓	×	×
	ResNet-18	MNIST	✓	✓	✓	✓	×	-	-	-
		CIFAR-100	✓	✓	×	×	×	-	-	-
		LFW	✓	×	×	×	×	-	-	-

Table 8: Examples of failed data leakage attack using the proposed GRNN on *LeNet* with batch sizes of 128 and 256. The top row shows the fake images recovered from the shared gradient and the bottom row shows corresponding true images in the private datasets.

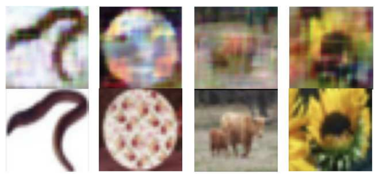
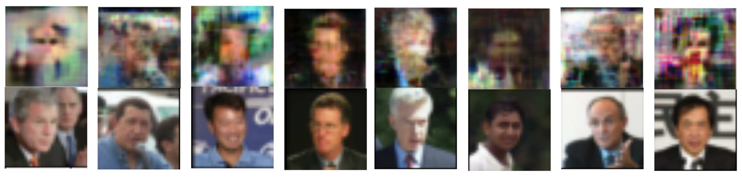
Dataset	CIFAR-100	LFW
Samples		

Table 9: Some randomly selected ground truth (GT) images and corresponding recovered images from GRNN and DLG with different batch sizes using *LeNet*.

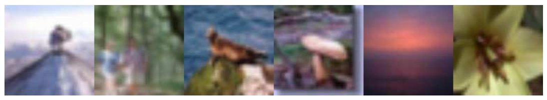
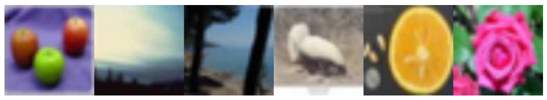
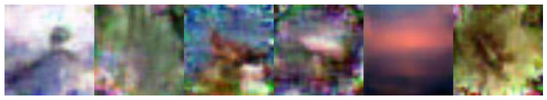
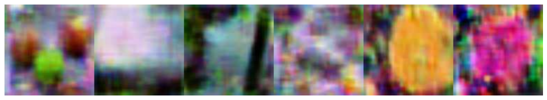
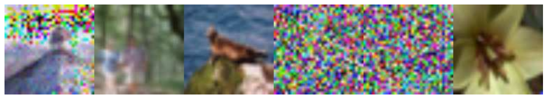
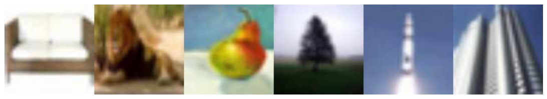
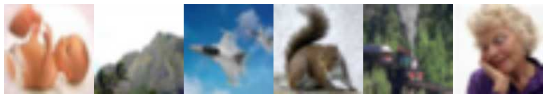
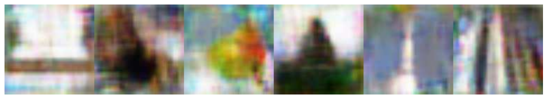
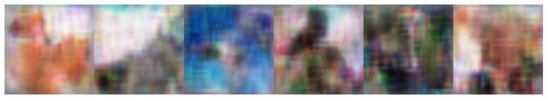
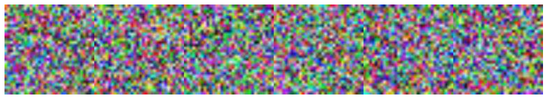
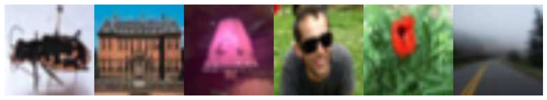
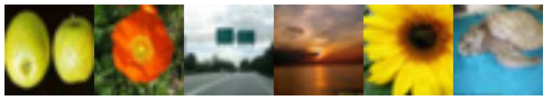
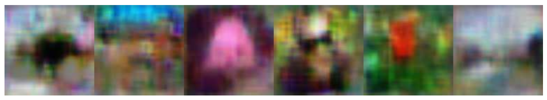
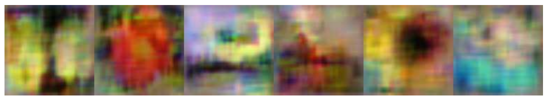
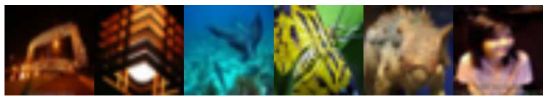
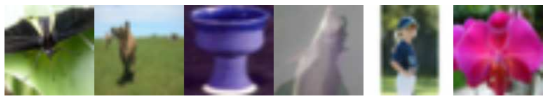
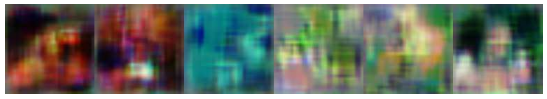
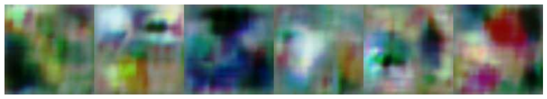
Method	**#Batch & Images**	
	1	4
GT		
GRNN		
DLG		
	8	16
GT		
GRNN		
DLG		
	32	64
GT		
GRNN		
DLG		
	128	256
GT		
GRNN		
DLG		

Table 10: Data leakage attack using GRNN with different image resolutions and sizes of training batch for FL model, where "✓" refers to a success and "×" refers to a failure. The network is *ResNet-18*, and the dataset is CIFAR-100.

Resolution \ **#Batch**	**1**	**4**	**8**	**16**	**32**
32*32	✓	✓	✓	✓	✓
64*64	✓	✓	✓	×	×
128*128	✓	×	×	×	×
256*256	✓	×	×	×	×

4.2.2 Ablation Study on Success Rate

To quantitatively evaluate the success rate of leakage attack, we conducted a comparison experiment on the testing set of CIFAR-100 using both GRNN and DLG. When the number of batch size is larger than 1, we treat the matching problems between ground-truth images and leaked images as a classic assignment problem given the similarity metrics, such as MSE and PSNR. In addition, Structural Similarity (SSIM) was also introduced to evaluate the structural discrepancy between two images. Fig. 5 shows the success rates of both methods with different numbers of batch size against the similarity threshold for so-called "successful attack". The lower MSE score indicates the better match, whereas, the higher PSNR and SSIM imply the better match. Fig. 5 shows that given the same success rate, GRNN achieves significantly lower MSE scores and higher PSNR and SSIM apart from the batch size of 1. For example, with a success rate of 0.6, the MSE threshold is around 0.05 for GRNN and 0.17 for DLG. Similarly, given the same threshold ratio, our method achieves a higher success rate. For instance, when a SSIM threshold of 0.2 is used, most success rates of DLG are dropped down to 0 apart from DLG with a batch size of 1 that achieves around 0.7. However, the worst success rate for GRNN is above 0.77. This experiment can further approve that the proposed method outperforms DLG by a significant margin. More qualitative comparisons that were randomly selected from the generated images are illustrated in Table 9.

4.2.3 Ablation Study on Image Resolution

As aforementioned in Section 3.2, GRNN is capable of handling different resolutions of images, due to the flexible number of upsampling blocks. We evaluate the recovery performance of larger resolutions than 32*32 using *ResNet-18* as the local model and the dataset is CIFAR-100. Table 10 shows the performance with different image resolutions and batch sizes. First, we upsample the original image from 32 to 64, the results show that even with the batch size of 8, GRNN is capable of recovering images from the shared gradient. Then we further explore the resolutions of 128*128 and 256*256, both experiments success with batch size of 1. Some qualitative results in resolutions are showed in Table 11. Table 12 shows the comparison results of GRNN and IG using a resolution of 256*256, the similarity between the recovered images and original images calculated with MSE, PSNR and SSIM are also given. GRNN outperforms IG by a significant margin for all samples when the SSIM metric is used. It is noticeable that our method is better than IG virtually even though the PSNR values of GRNN are lower for some cases with the batch size of 4 and 8. Based on this study, we can conclude that our method is also capable of recovering the global structure and color appearance in the image when a large batch is used, while IG likely produces virtually unrecognizable images.

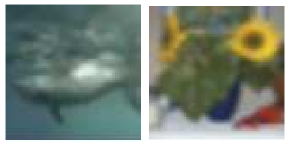
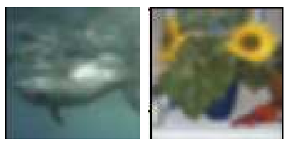
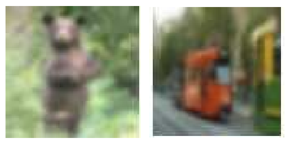
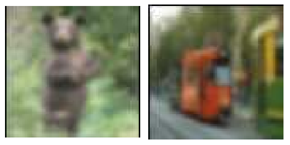
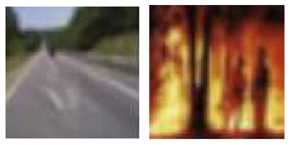
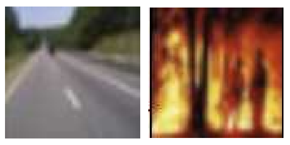
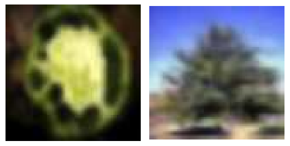
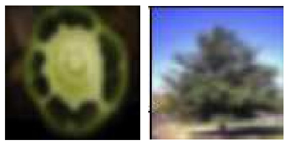
4.2.4 Ablation Study on Loss Function

MSE loss is widely used in regression tasks, however, it can be easily biased to the outlier or noisy data point at a pixel-wise level. We believe that the distribution information embedded in the gradient vectors indicates the global structure of the image data. Therefore, we introduced WD distance to measure the geometric discrepancy between the fake gradient and true gradient and guide the image generation process. In addition, we carried out comparison experiments to evaluate different combinations of loss functions including MSE, WD, TVLoss and CD. The results can be found in Table 13. The experimental results show that the proposed loss objective combining MSE, WD and TVLoss achieves the best performance. It is noteworthy mentioning that the color distortion and artifact can be suppressed by using TVLoss and WD jointly as they can effectively penalize the spurious noises locally and globally.

4.3 Label Inference

In addition to image data recovery, we also investigated the performance of label inference in those experiments. Table 14 provides the comparative results of label inference accuracy (%) for DLG and GRNN, where each experiment

Table 11: Recovered images with different resolutions using GRNN. The network is non-converged *ResNet-18*, batch size is 1 and dataset is CIFAR-100.

Resolution	Recovered Images	Ground Truth
32*32		
64*64		
128*128		
256*256		

was repeated 10 times, the mean and standard deviation were reported. We can observe that the label inference accuracy decreases while the size of the training batch increases for both DLG and GRNN. However, our method outperforms DLG in all experiments except the one on MNIST with a batch size of 8. Furthermore, GRNN is significantly better than DLG when a large batch size is used. For example, GRNN achieves 99.84% on LFW using *LeNet* and a batch size of 256, whereas DLG only obtains 79.69% using the same setting. Note that having a correct label prediction does not necessarily indicate the corresponding image data can be recovered successfully (see Table 7). We can conclude that recovering image information is much more challenging than recovering labels as the distribution of image data is in a much higher dimension than its corresponding label. The accuracy of label inference on *ResNet-18* achieves 100% in almost all experiments except the one on LFW with a batch size of 32 (94.06%), which is higher and more stable than *LeNet*. *ResNet-18* has much more trainable parameters than *LeNet*, therefore, the gradient with a larger number of elements is much more informative for finding decision boundaries for the classification task.

We also noticed that the number of label classes has an impact on its inference performance. In our experiment, MNIST, CIFAR-100, and LFW have 10 classes, 100 classes, and 5749 classes, respectively. The average accuracy of DLG is 94.23% on MNIST while decreasing to 93.51% and 85.41% on CIFAR-100 and LFW, respectively. In contrast, GRNN achieves higher accuracy on LFW compared to the other two datasets. In DLG, the label is obtained via updating the image input and label input jointly during the backward pass phase in SGD, whereas in GRNN, the label is calculated by the fake label generator in the forward pass phase. We believe that by adding the image data and label generators, GRNN can better capture the correspondence between image data and its label in the joint latent space and, furthermore, can generate more individualized images with respect to different classes.

4.4 Face Re-Identification

As shown in Table 6, recovered images look almost the same as their corresponding true images, there are still slight deviations that may produce classification misleading results if we use generated data to replace the original ones. This behavior of DL methods are well documented in the literature, *e.g.* [46, 47, 48]. Taking face recognition as an example, the recovered face image may look identical to its original image visually, however, it cannot produce correct prediction by the face recognition model to identify the person, see Table 16. Therefore, we applied the face re-identification experiment to evaluate the feasibility of data leakage attack using GRNN. We first used the GRNN to recover the face image data from the FL system during the training stage. Then, the fake image was passed to the face recognition model as input to predict the identity label. The success of re-identification was counted if the prediction label matches the true label. VGG-Face dataset was used in this experiment which contains 2622 identities.

Table 12: Randomly selected images recovered by GRNN and IG. The network is non-converged *ResNet-18*, resolution is 256*256, and dataset is CIFAR-100.

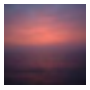
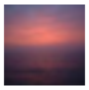
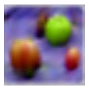
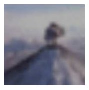
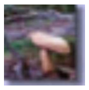
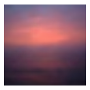
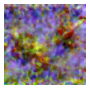
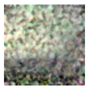
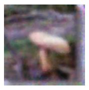
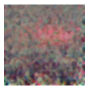
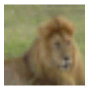
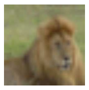
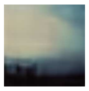
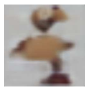
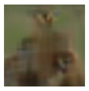
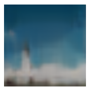
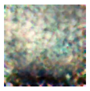
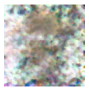
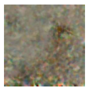
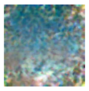
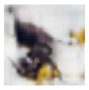
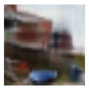
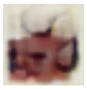
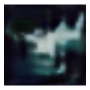
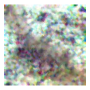
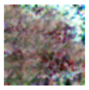
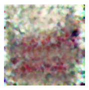
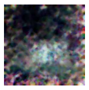
Method	**#Batch & Images**								
	1								
GT									
	GRNN				—	IG			
									
MSE	4107.90	**3656.38**	**45.75**	**53.21**	—	**2272.82**	4853.13	94.19	1729.50
PSNR	**38.29**	37.88	**40.43**	**39.82**	—	38.09	**37.98**	39.56	38.11
SSIM	**0.49**	**0.90**	**0.95**	**0.98**	—	0.31	0.26	0.85	0.55
	4								
GT									
	GRNN				—	IG			
									
MSE	**1482.89**	5996.54	1391.17	**1799.29**	—	2261.19	**4481.52**	**716.74**	2147.60
PSNR	37.97	38.03	37.83	37.92	—	**38.07**	**38.05**	**38.27**	**38.12**
SSIM	**0.90**	**0.61**	**0.75**	**0.86**	—	0.39	0.34	0.56	0.40
	8								
GT									
	GRNN				—	IG			
									
MSE	4874.68	**1280.33**	**1441.29**	3380.90	—	**4836.14**	3309.53	2693.98	**3059.81**
PSNR	37.94	**38.14**	**38.09**	**38.67**	—	**38.01**	38.08	38.06	38.11
SSIM	**0.54**	**0.59**	**0.70**	**0.54**	—	0.24	0.28	0.35	0.33

Table 13: Comparison of image recovery using different loss functions on MNIST dataset.

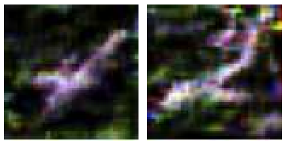
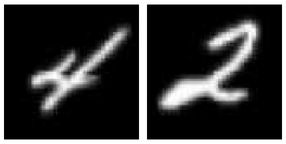
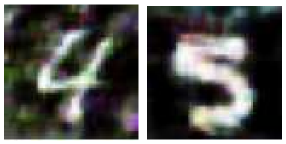
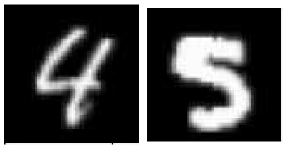
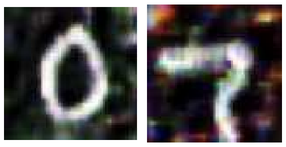
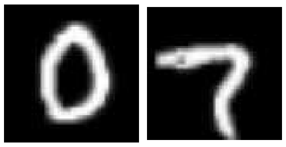
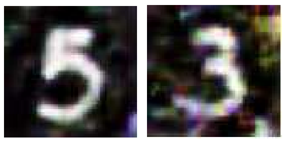
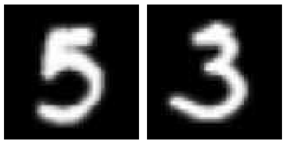
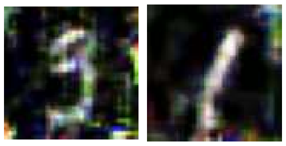
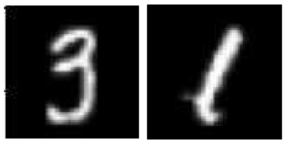
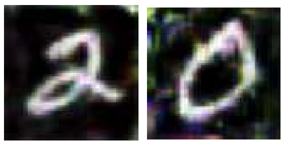
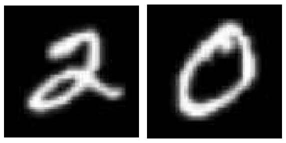
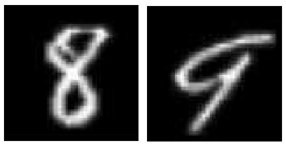
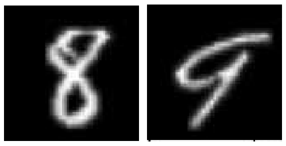
Loss Function	**Recovered Images**	**Ground Truth**
MSE		
WD		
CD		
MSE & WD		
MSE & CD		
MSE & CD & TVLoss		
MSE & WD & TVLoss		

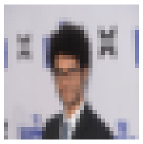
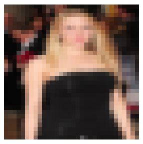
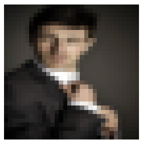
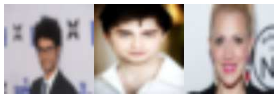
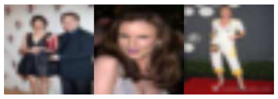
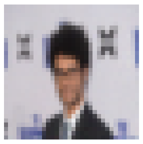
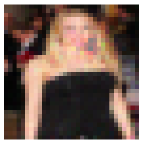
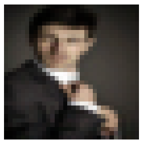
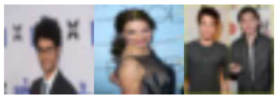
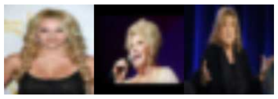
Table 14: Comparison of label inference accuracy (%) using DLG and GRNN, where L. and R. refer to LeNet and ResNet-18 respectively.

#Batch			**1**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**Avg**
DLG	L.	MNIST	**100.0± 0.0**	97.50± 0.0750	**100.0± 0.0**	94.38± 0.0337	90.62± 0.0242	90.31± 0.0260	91.41± 0.0152	92.42± 0.0181	94.23± 0.0254
		CIFAR	**100.0± 0.0**	97.50± 0.0750	98.75± 0.0375	97.50± 0.0306	92.50± 0.0287	89.84± 0.0329	86.33± 0.0295	85.70± 0.0289	93.51± 0.0329
		LFW	**100.0± 0.0**	80.00± 0.3317	90.00± 0.1561	85.00± 0.2358	88.44± 0.1683	70.62± 0.3924	89.53± 0.2985	79.69± 0.3985	85.41± 0.2477
Ours	L.	MNIST	**100.0± 0.0**	**100.0± 0.0**	97.50± 0.0500	97.50± 0.0415	97.50± 0.0187	**96.25± 0.0188**	**96.25± 0.0171**	**96.37± 0.0165**	97.73± 0.0203
		CIFAR	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	99.38± 0.0188	99.69± 0.0094	**98.91± 0.0122**	**98.75± 0.0080**	**96.80± 0.0108**	99.19± 0.0074
		LFW	**100.0± 0.0**	97.50± 0.0750	98.75± 0.0375	**100.0± 0.0**	**100.0± 0.0**	**99.69± 0.0062**	**99.69± 0.0071**	**99.84± 0.0019**	**99.43± 0.0160**
	R.	MNIST	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	- -	- -	- -	**100.0± 0.0**
		CIFAR	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	- -	- -	- -	**100.0± 0.0**
		LFW	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	**100.0± 0.0**	94.06± 0.0452	- -	- -	- -	98.81± 0.0090

Table 15: Performances of different network architectures, where training accuracy refers to predicted results of true images and relevant ground truth label. Re-identification accuracy is from predicted results of fake images and relevant ground truth labels. DLG and GRNN both use *LeNet* as backbone. Training and testing samples are from VGG-Face dataset. Res18 represents to ResNet-18 and Dense121 is DenseNet-121.

Method	**Network**	**Train Acc**	**#B**	**Re-identification Accuracy**			**Sample No.**
				Top-1	**Top-3**	**Top-5**	
DLG	Res18	97.27%	1	25.14%	45.57%	51.86%	700
	Dense121	97.11%	1	**15.57%**	33.57%	42.14%	700
GRNN	Res18	97.27%	1	**30.66%**	**74.79%**	**88.40%**	700
			4	17.45%	24.64%	31.11%	1112
			8	6.14%	13.58%	22.13%	2224
			16	2.90%	10.43%	22.08%	4352
	Dense121	97.11%	1	11.46%	**43.12%**	**63.03%**	700
			4	9.53%	19.87%	26.80%	1112
			8	3.06%	10.25%	19.83%	2224
			16	1.52%	8.23%	19.12%	4352

Table 16: The re-identification results of true images and their corresponding generated fake images.

Ground Truth Label	**#42**	**#26**	**#69**
True Input Image			
Top-3 Labels	**#42** — #97 — #94	**#26** — #22 — #64	#22 — #64 — #23
Top-3 Images			
Top-3 Confidences	40.70% — 10.84% — 4.08%	90.34% — 4.04% — 1.75%	66.10% — 5.02% — 4.71%
Generated Input Image			
Top-3 Labels	**#42** — #4 — #22	#57 — #31 — #75	**#69** — #76 — #88
Top-3 Images			
Top-3 Confidences	11.89% — 7.58% — 7.46%	73.10% — 7.88% — 3.91%	59.52% — 7.96% — 3.29%

We used the top 100 identities that have the most image samples for training. We selected the successfully recovered images data from the output of GRNN which ended up with 700 fake face images in total when batch size is 1, and 1112 images, 2224 images, and 4352 images for batch size 4, 8, and 16 respectively. In the meantime, we trained two face recognition models using *ResNet-18* and *DenseNet-121* using the same training set. Table 15 reports the top-1, top-3 and top-5 accuracies of re-identification of those fake face images. Although the top-1 accuracy on *ResNet-18* and *DenseNet-121* are 30.66% and 11.46% when batch size is 1, they are significantly better than random prediction (1%). The re-identification accuracy increases dramatically when we consider using top-3 and top-5 metrics. The recovery performance becomes worse with the increasing batch size, so the accuracy decreases as well. We say that the face recognition deep models are sensitive to the minor perturbation that is produced during the image recovery process. The images generated using GRNN achieve significantly higher accuracies compared to DLG, *i.e.* +5.52%, +29.22% and +36.54% can be achieved in *Top-1*, *Top-3* and *Top-5* accuracies using *ResNet-18*.

4.5 Defense Strategy

The most relevant defense approach for GRNN is noise addition, where, in our scenario, the clients can add a level of Gaussian or Laplacian noise onto the shared gradient. We empirically evaluated the effectiveness of GRNN when the noise addition defense strategy was used. In this study, the *LeNet* was used as the global FL model with different batch sizes. As for the Laplacian mechanism, it adds Laplacian-distributed noise to function f. In this paper, we set the l1-sensitivity Δf to be 1 and varied ϵ, which can be defined as: $\lambda = \frac{\Delta f}{\epsilon} \in [1e-1, 1e-4]$. As for the Gaussian mechanism, it also adds randomness with a normal distribution. Technically, the Gaussian mechanism uses l2-sensitivity and parameter δ is counted on. We simplify the Gaussian mechanism to add the noise whose distribution has 0 mean and only ranges standard deviation from 1e-1 to 1e-4. Fig. 3 shows the MSE distance between recovered image and true image when different levels and types of noises are added. GRNN fails to recover the image when a high level of noise is added to the gradient (see Figs. 3 (a) and (e)), however, this usually leads to poor performance on the global FL model as the noisy gradients are aggregated. We observed that GRNN is able to recover the image data successfully and obtains reasonable results when the scale of noise is reduced to 0.01 (see Figs. 3 (b) and (f)). The average PSNR scores are presented in Table 17. Compared to the no-defense approach shown in Table 5, the noise added to the gradient can result in decreasing of PSNR of the generated images, which indicates the effectiveness of the noise addition strategy. However, the proposed GRNN is still capable of recovering image data when a high level of noise is added to the gradient, *i.e.* 1e-2. The PSNR scores with Gaussian noise scale of 1e-4 on MNIST dataset are even larger than that without noise (52.51 dB VS. 52.37 dB), as well on LFW dataset (46.26 dB VS. 45.57 dB). On the other hand, we found that even the image recovery fails, the label can still be inferred correctly using our GRNN. However, DLG totally fails to recover images and inference labels in all of our experiments.

5 Conclusion

In this paper, we proposed a data leakage attack method, namely GRNN, for FL system which is capable of recovering both data and its corresponding label. Compared to the state-of-the-art methods, the proposed method is much more stable when a large resolution and a batch size are used. It also outperforms state-of-the-art in terms of fidelity of recovered data and accuracy of label inference. Meanwhile, the experimental results on face re-identification task suggest that GRNN outperforms DLG by a margin in terms of *Top-1*, *Top-3* and *Top-5* accuracies. We also discussed the potential defense strategies and empirically evaluate the performance of GRNN when noise addition defense is applied. We conclude that our method can successfully and consistently recover the data in FL when a high level of noise is added to the gradient. The implementation of our method is publicly available to ensure its reproducibility.

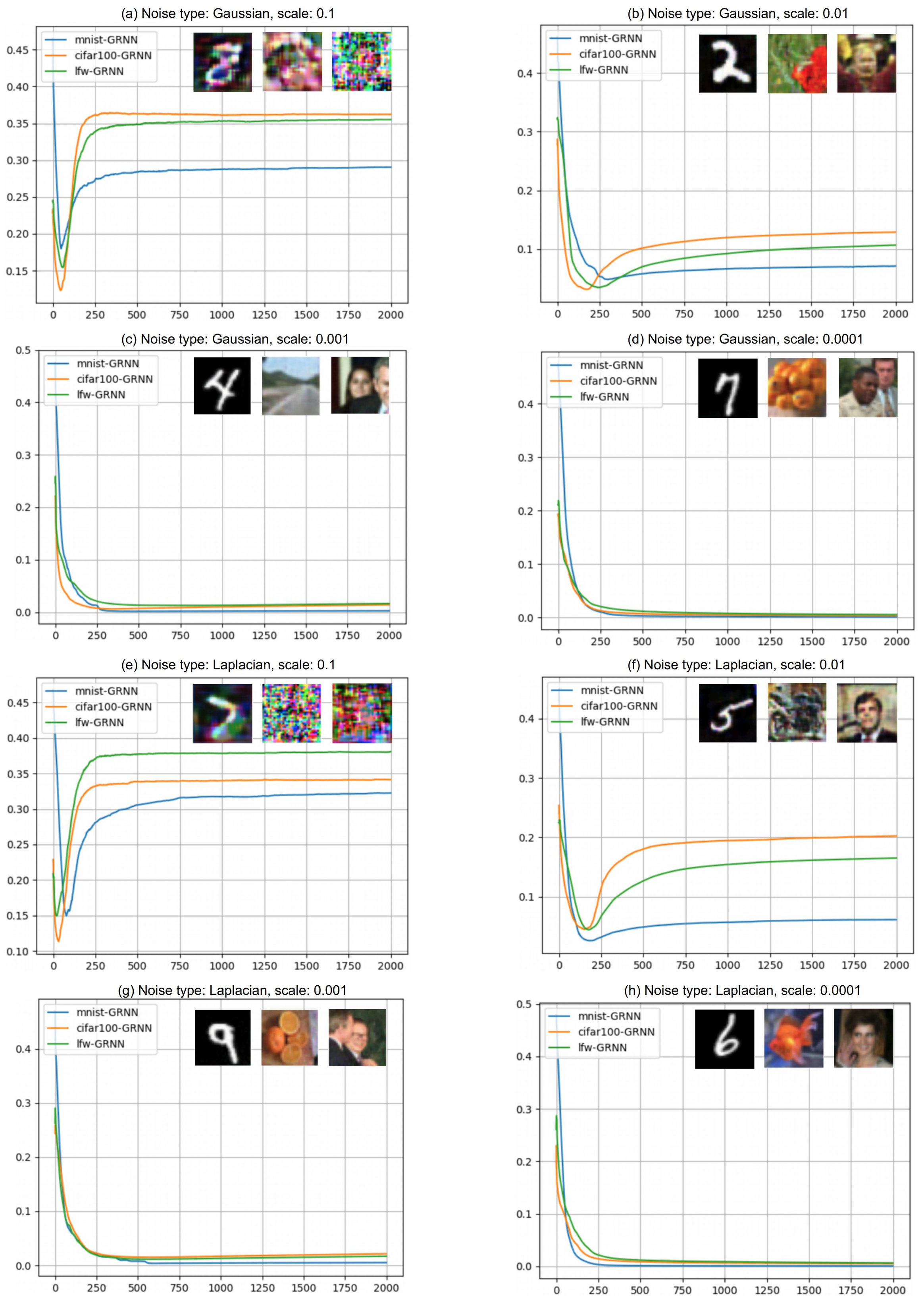


Figure 3: MSE distances between true images and generated fake images during training and illustration of generated fake images from GRNN with different noise types and scales on three datasets. (a) - (d) are the results of adding Gaussian noise, and (e) - (h) are the results of adding Laplacian noise. The horizontal axis is the number of iteration for training the attack model.

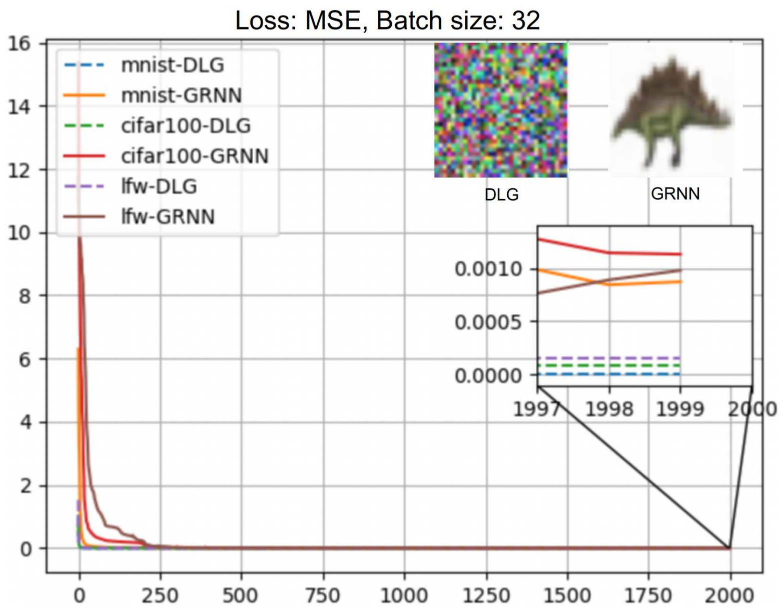
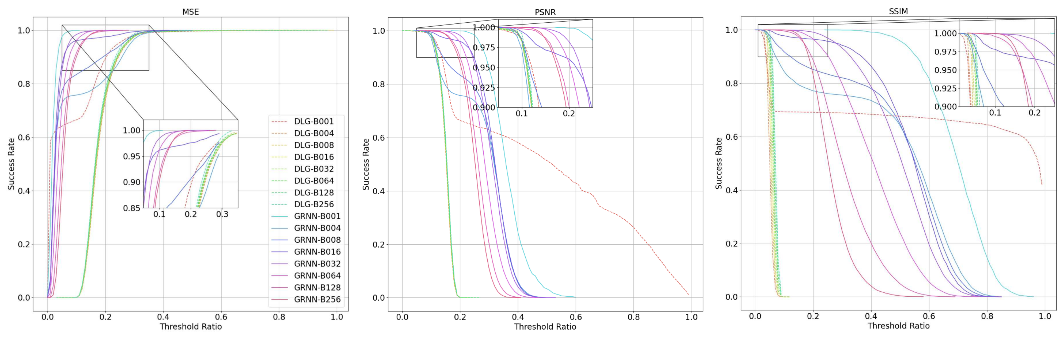


Figure 4: MSE loss visualization for DLG and GRNN with batch size 32.

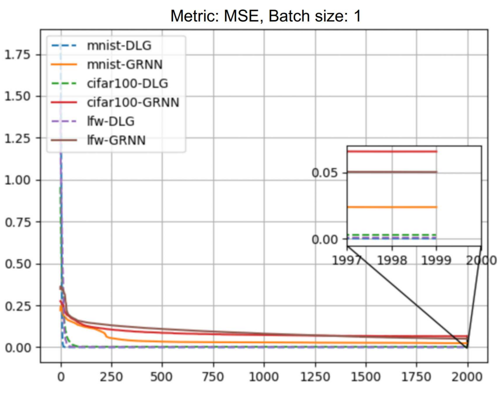


(a) MSE (b) PSNR (c) SSIM

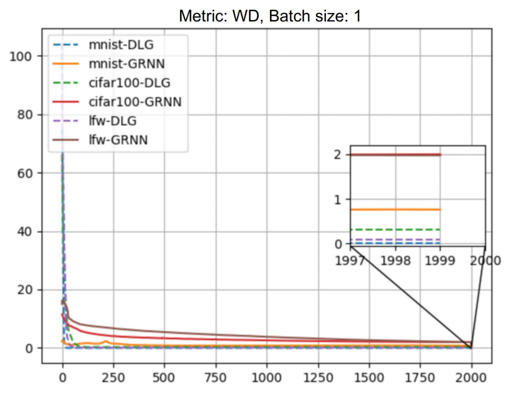
Figure 5: Successful attack rate for different batch size over normalized threshold ratio on MSE, PSNR and SSIM similarity metrics. The dash line refers to the results from DLG, and the solid line is from the proposed GRNN

Table 17: Average PSNR scores with different noise types and scales. "×" means the method fails the experiment, whereas PSNR score is given only if it successes. DLG failed completely, as it had no visible success among all the experiments.

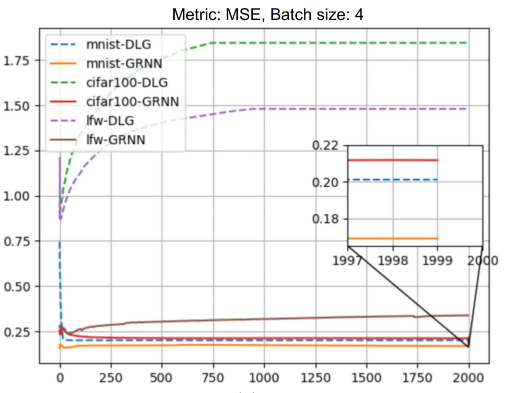
Method	**Dataset**	**Type** \ **Scale**	**#Batch**	**1e-1**	**1e-2**	**1e-3**	**1e-4**
DLG	MNIST	Gaussian	1	×	×	×	×
			4	×	×	×	×
		Laplacian	1	×	×	×	×
			4	×	×	×	×
	CIFAR-100	Gaussian	1	×	×	×	×
			4	×	×	×	×
		Laplacian	1	×	×	×	×
			4	×	×	×	×
	LFW	Gaussian	1	×	×	×	×
			4	×	×	×	×
		Laplacian	1	×	×	×	×
			4	×	×	×	×
Ours	MNIST	Gaussian	1	39.73	39.73	47.20	52.51
			4	×	39.66	39.67	44.43
			8	×	40.11	39.65	40.71
			16	×	×	39.73	39.78
		Laplacian	1	39.69	39.55	45.08	51.80
			4	×	39.67	39.83	45.39
			8	×	×	39.66	41.02
			16	×	×	39.62	40.00
	CIFAR-100	Gaussian	1	×	38.18	41.78	45.75
			4	×	×	38.29	40.64
			8	×	×	38.18	39.26
			16	×	×	×	38.65
		Laplacian	1	×	38.13	40.17	46.20
			4	×	×	38.14	40.32
			8	×	×	38.08	39.14
			16	×	×	×	×
	LFW	Gaussian	1	×	38.15	41.72	46.26
			4	×	×	38.25	40.59
			8	×	×	×	×
			16	×	×	×	×
		Laplacian	1	×	38.09	40.96	46.09
			4	×	×	38.19	40.26
			8	×	×	×	×
			16	×	×	×	×



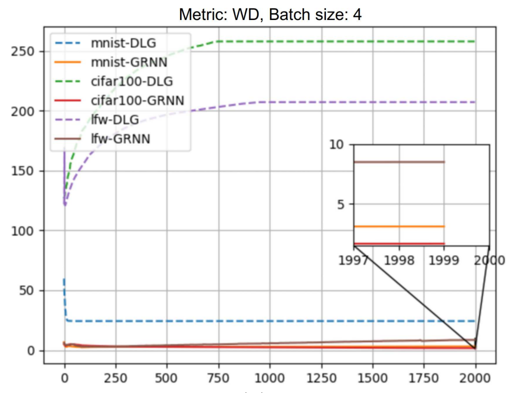
(a)



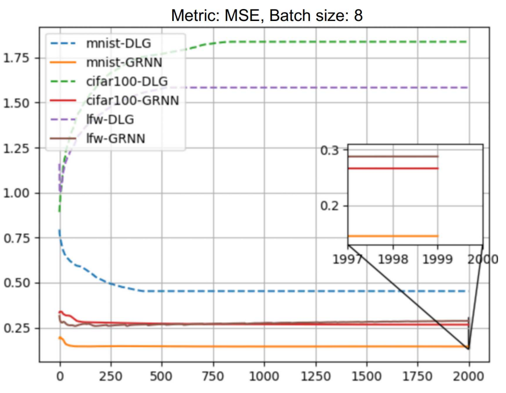
(b)



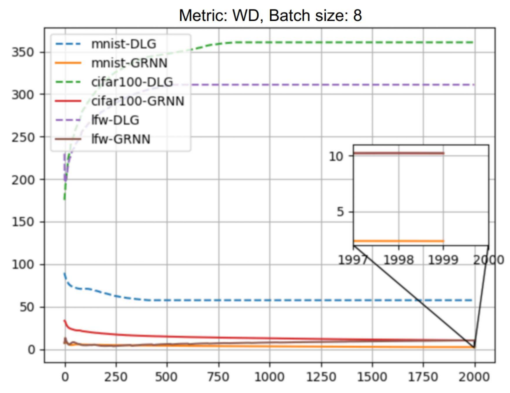
(c)



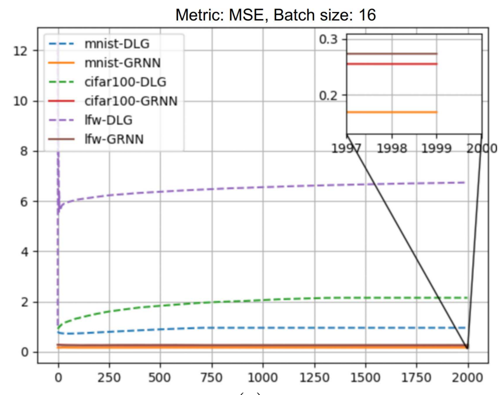
(d)



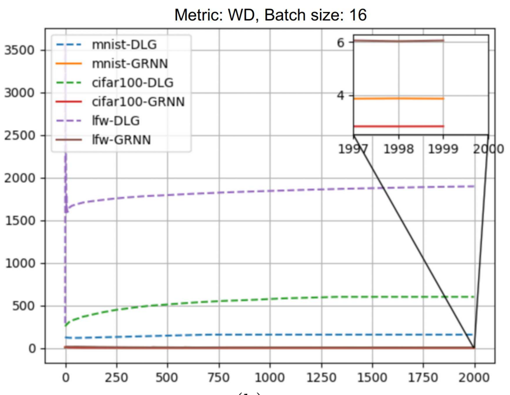
(e)



(f)



(g)



(h)

Figure 6: Distances between true and generated images with respect to training iteration for DLG and GRNN on three datasets. The horizontal axis corresponds to the number of training iterations of two attacking models and the vertical axis corresponds to the similarity metrics.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, pages 770–778, 2016.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[3] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598, 2014.

[4] Eric P Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Trans. on Big Data*, 1(2):49–67, 2015.

[5] Philipp Moritz, Robert Nishihara, Ion Stoica, and Michael I Jordan. Sparknet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015.

[6] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. Firecaffe: near-linear acceleration of deep neural network training on compute clusters. In *IEEE CVPR*, pages 2592–2600, 2016.

[7] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. 2018.

[8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *PMLR AISTATS*, pages 1273–1282, 2017.

[9] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, pages 1322–1333, 2015.

[10] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE SP*, pages 3–18, 2017.

[11] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *ACM CCS*, pages 603–618, 2017.

[12] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE SP*, pages 691–706, 2019.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[14] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NIPS*, pages 14774–14784, 2019.

[15] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[16] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

[17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[18] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptol.*, 2015:1192, 2015.

[19] Owusu-Agyemang Kwabena, Zhen Qin, Tianming Zhuang, and Zhiguang Qin. Mscryptonet: Multi-scheme privacy-preserving deep learning in cloud computing. *IEEE Access*, 7:29344–29354, 2019.

[20] Cynthia Dwork. Differential privacy. In *Springer ICALP*, pages 1–12, 2006.

[21] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in TCS*, 9(3-4):211–407, 2014.

[22] Zhiqi Bu, Jinshuo Dong, Qi Long, and Weijie J Su. Deep learning with gaussian differential privacy. *Harvard data science review*, 2020(23), 2020.

[23] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Secure linear regression on vertically partitioned datasets. *IACR Cryptol.*, 2016:892, 2016.

[24] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE SP*, pages 19–38, 2017.

[25] Louis JM Aslett, Pedro M Esperança, and Chris C Holmes. Encrypted statistical machine learning: new privacy preserving methods. *arXiv preprint arXiv:1508.06845*, 2015.

[26] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[27] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[28] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Deep neural networks classification over encrypted data. In *ACM CODASPY*, pages 97–108, 2019.

[29] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[30] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. on IFS*, 13(5):1333–1345, 2017.

[31] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. Towards efficient and privacy-preserving federated deep learning. In *IEEE ICC*, pages 1–6, 2019.

[32] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *ACM CCS*, pages 1310–1321, 2015.

[33] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. 2017.

[34] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *IEEE CVPR*, pages 2528–2535, 2010.

[35] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE ICCV*, pages 2018–2025, 2011.

[36] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *PMLR ICML*, pages 933–941, 2017.

[37] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *PMLR ICML*, pages 214–223, 2017.

[38] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[39] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[40] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[41] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. 2008.

[42] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[43] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NIPS*, 32:8026–8037, 2019.

[45] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *IEEE ICPR*, pages 2366–2369, 2010.

[46] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE CVPR*, pages 2574–2582, 2016.

[47] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Sparsefool: a few pixels make a big difference. In *IEEE CVPR*, pages 9087–9096, 2019.

[48] Ali Shahin Shamsabadi, Ricardo Sanchez-Matilla, and Andrea Cavallaro. Colorfool: Semantic adversarial colorization. In *IEEE CVPR*, pages 1151–1160, 2020.