MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Data Privacy and Protection on Deep Leakage from Gradients by Layer-Wise Pruning

Liu, Bryan; Koike-Akino, Toshiaki; Wang, Ye; Kim, Kyeong Jin; Brand, Matthew E.; Aeron, Shuchin; Parsons, Kieran

TR2022-081 August 06, 2022

Abstract

In this paper, we study a data privacy and protection problem in a federated learning system for image classification. We assume that an attacker has full knowledge of the shared gradients during the model update. We propose a layer-wise pruning defense to prevent data leakage from the attacker. We also propose a sequential update attack method, which accumulates the information across training epochs. Simulation results show that the sequential update can gradually improve the image reconstruction results for the attacker. Moreover, the layer-wise pruning method is shown to be more efficient than classical element-wise threshold-based pruning on the shared gradients.

IEEE Information Theory and Applications Workshop (ITA) 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Data Privacy and Protection on Deep Leakage from Gradients by Layer-Wise Pruning

Bryan Liu*†, Toshiaki Koike-Akino*, Ye Wang*, Kyeong Jin Kim*,
Matthew E. Brand*, Shuchin Aeron‡, Kieran Parsons*
*Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.
†Electrical Engineering and Telecommunication, University of New South Wales, Sydney, Australia.
‡Electrical and Computer Engineering , Tufts University, MA, USA.
Email: bryan.liu@unswalumni.com, {koike, yewang, kkim, brand, parsons}@merl.com

***Abstract*—In this paper, we study a data privacy and protection problem in a federated learning system for image classification. We assume that an attacker has full knowledge of the shared gradients during the model update. We propose a layer-wise pruning defense to prevent data leakage from the attacker. We also propose a sequential update attack method, which accumulates the information across training epochs. Simulation results show that the sequential update can gradually improve the image reconstruction results for the attacker. Moreover, the layer-wise pruning method is shown to be more efficient than classical element-wise threshold-based pruning on the shared gradients.**

***Index Terms*—Federated learning, Data leakage, Privacy protection**

I. Introduction

Federated learning is a distributed technique for training a model without directly sharing the training data from each local device [1]. A collaborative training process is achieved by sharing and aggregating model weights or gradients, produced from the local training at each device. However, as shown by [2], the common method of gradient sharing produces a potential privacy leakage issue, where an attacker can reconstruct the training data from the knowledge of the shared model gradients. The general approach for the attacker is to optimize the similarity between the shared model gradients and the loss gradients produced by the model when applied to a dummy input. The successful reconstruction attack demonstrated by [2] uses mean-squared error as the attacker loss function when matching gradients produced by the dummy (reconstruction) input to the shared gradients. Further investigation in [3] shows that by exploiting the gradients of the final layer of the neural network model, the attacker might not even need knowledge of the labels of the original data for successful reconstruction.

These attacks were refined in [4], by adding a regularization term (total variation [5]) and adopting cosine similarity for gradient matching, improving successful reconstruction for an entire batch of images, while dealing with deep network structures such as ResNet18 and ConvNet. It is further shown in [6] that combining the reconstruction results from a group of randomly initialized images can enhance the overall reconstruction performances by an attacker.

The studies in [7], [8] evaluate the gradient inversions attack algorithm and defense methods in a federated learning framework. However, the reconstruction performance is mostly measured in a static model, where the trainable parameters in a neural network are unchanged. Meanwhile, to analyze data leakage, the parameters of an untrained/trained model are assumed to be known by the attacker. Our aim is to consider data leakage over the whole training process, where an attacker eavesdrops on the gradient sharing from the beginning of model updates.

In this paper, we study the gradient inversions problem in a dynamic training model for image classification. We assume that an attacker has full knowledge of the shared gradients and the model is updated over several training epochs. By saving and combining information across the training epochs, the attacker can improve the reconstruction accuracy of the source data. From the perspective of effective distributed learning, an important goal is to minimize the degradation upon the final classification accuracy due to privacy protection method. Therefore, this paper considers this privacy-utility trade-off, with the main contributions summarized as follows:

1) We analyse the trade-off between privacy protection and test accuracy for classifying images by computing the area under the curves of test accuracy

B. Liu conducted this research during his internship at MERL.

and structural similarity (SSIM) between the reconstructed and source images.
2) We propose a magnitude-based alternating layer-wise pruning method to prevent data leakage. We demonstrate that the alternating optimization method does not influence the final test accuracy on classification when the number of layers to be pruned is small.
3) By comparing to the element-wise threshold-based pruning method, we demonstrate that the layer-wise pruning method can efficiently prevent data leakage during the model update.
4) We propose a sequential reconstruction method for the attacker, which gradually improves the reconstruction performance during the model update.

II. Preliminary

Before introducing our proposed layer-wise pruning method for data protection and the sequential update method for the attacker's reconstruction, we briefly review the gradient inversion technique that is proposed in [4] and the element-wise threshold-based gradient pruning method for data protection from [2].

A. Inverting Gradients

Inverting gradients in [4] introduces an image reconstruction method for a deep neural network given the shared gradients by a local user. Suppose there is a neural network model $\mathcal{M}$ that is designed for image classification. Let $(\mathbf{x}, \mathbf{y})$ be a mini-batch for training, where $\mathbf{x}$ refers to the input images and $\mathbf{y}$ refers to the corresponding labels. Then, we define $\partial W_{\mathbf{x}}$ as the combined sets of gradients of all trainable parameters W from a model training loss function $\mathcal{F}\Big(\mathcal{M}(\mathbf{x}), \mathbf{y}\Big)$. The main target of an attacker is to use the knowledge of $\partial W_{\mathbf{x}}$ and W to recover the source data $\mathbf{x}$. As proposed in [2], [3], a trainable dummy image $\hat{\mathbf{x}}$ is firstly initialized. Then, an objective function $\mathcal{L}$ is defined to measure the distance between the gradients $\partial W_{\mathbf{x}}$ and $\partial W_{\hat{\mathbf{x}}}$. By computing the derivative $\partial\mathcal{L}/\partial\hat{\mathbf{x}}$, the trainable images $\hat{\mathbf{x}}$ can be gradually tuned by stochastic gradient descent. Successful reconstruction on the source data implies privacy leakage in the distributed training system. The objective function $\mathcal{L}$ in [4] is defined as:

$$\mathcal{L} = \left(1 - \frac{\langle \partial W_{\mathbf{x}}, \partial W_{\hat{\mathbf{x}}} \rangle}{||\partial W_{\mathbf{x}}||_2 ||\partial W_{\hat{\mathbf{x}}}||_2}\right) + \alpha \mathrm{TV}(\hat{\mathbf{x}}), \quad (1)$$

where $\mathrm{TV}(\hat{\mathbf{x}})$ refers to the total variation of the trainable dummy image $\hat{\mathbf{x}}$, α is a scaling factor to control the contribution of total variation to the loss function $\mathcal{L}$ and $||\cdot||_2$ denotes the Euclidean norm function. Compared to the mean-square error loss function in [2], [3], the cosine similarity term, $\frac{\langle \partial W_{\mathbf{x}}, \partial W_{\hat{\mathbf{x}}} \rangle}{||\partial W_{\mathbf{x}}||_2 ||\partial W_{\hat{\mathbf{x}}}||_2}$, improves performance by eliminating the effect of varying magnitudes across the trainable parameters.

B. Element-wise Threshold-based Pruning

To prevent data leakage, a direct way is to modify the shared gradients. As proposed in [2], a threshold-based pruning method is a low-cost and effective way to reduce the reconstruction performance of an attacker. [2] proposes to prune the gradients to 0 if the magnitudes of gradients are smaller than a threshold. To define the threshold, a percentage value p from 0 to 1 is firstly determined. Then, let $\partial W_{\mathbf{x}}^i$ be the ith set of gradients in $\partial W_{\mathbf{x}}$, the threshold l_i is found by $l_i = \mathcal{P}(\partial W_{\mathbf{x}}^i, p)$, where $\mathcal{P}(\cdot)$ represents a percentile function that finds an element in $\partial W_{\mathbf{x}}^i$ that has a magnitude greater than p-percentage of the elements in $\partial W_{\mathbf{x}}^i$. As a result, for the jth element in $\partial W_{\mathbf{x}}^i$, the element-wise threshold-based pruning method returns

$$(\partial \hat{W}_{\mathbf{x}}^i)_j = \begin{cases} 0, & \text{for } |(\partial W_{\mathbf{x}}^i)_j| < l_i, \\ (\partial W_{\mathbf{x}}^i)_j, & \text{for } |(\partial W_{\mathbf{x}}^i)_j| > l_i, \end{cases} \quad (2)$$

We can observe that in the gradients of each network layer, only a portion may be pruned to 0. However, the ground-truth gradients are computed from the chain rule, where the relationship of gradients between consecutive layers can be easily derived. Pruning part of the gradients can be viewed as signals with incomplete sampling [9], which suggests that the attacker can still recover the source images with reasonable accuracy. Therefore, in the following, we propose a layer-wise pruning method, which is more effective than the element-wise threshold-based pruning. In addition, we propose a sequential update method for the attacker, which can improve the attacker's reconstruction performance.

III. Dynamic System - Layer-Wise Pruning and Sequential Update

A. Layer-wise pruning

Different from the threshold-based pruning method discussed in the previous section, our proposed pruning method zeroes out entire layers of gradients, selected by comparing each layer's averaged gradient magnitudes across the network. In this paper, we consider a valid layer in a neural network as either a fully connected layer or a convolutional layer, which contain trainable multiplicative weights and bias. A batch normalization layer also contains trainable parameters but as shown

Algorithm 1 Layer-wise Pruning by a Local User

Step 1: Compute the gradients $\partial W_{\mathbf{x}}$.
Step 2: Find the averaged gradient magnitudes $\mathbf{g}$ as in (3).
Step 3: Find the sorted averaged gradient magnitudes $\mathbf{g}^{\text{sort}}$.
Step 4: Train the model for K epochs and zero-out T layers' gradients. Then go back to Step 1.

in [7], the attacker can learn the normalization terms during reconstruction. Therefore, we do not consider this type of layer for pruning.

Define $N = \{n_1, n_2, ..., n_L\}$ as the number of elements of each valid layer and $\partial D_{\mathbf{x}} = \{\partial D^1_{\mathbf{x}}, \partial D^2_{\mathbf{x}}, ..., \partial D^L_{\mathbf{x}}\}$ as the set of gradients for each layer, where $\partial D^i_{\mathbf{x}} \subset \partial W_{\mathbf{x}}$ for $i \in \{1, 2, ..., L\}$. The average gradient magnitudes $\mathbf{g} = [g_1, g_2, ..., g_L]$ of each layer is found by

$$g_i = \frac{1}{n_i} ||\partial D^i_{\mathbf{x}}||_1. \tag{3}$$

Depending on the number of training mini-batches that a local device has, this averaged gradient magnitudes can be further evaluated by averaging over all of the training mini-batches. Based on the average gradient magnitudes of each layer, the local device can choose the layer to be pruned depending on the requirement of classification accuracy and privacy protection capability. In our method, the layers with the smallest gradient magnitudes are pruned. We use $\mathbf{g}^{\text{sort}} = [g_1^{\text{sort}}, g_2^{\text{sort}}, ..., g_L^{\text{sort}}]$ with $g_1^{\text{sort}} < g_2^{\text{sort}} < ... < g_L^{\text{sort}}$ to indicate the sorted averaged gradient magnitudes for each layer and $\mathbf{h} = \{h_1, h_2, ..., h_L\}$ as the corresponding set of indices in $\partial W_{\mathbf{x}}$ for the sorted layers. As a result, suppose there are T layers selected for pruning, the layer-wise pruning method has

$$\partial \hat{D}^i_{\mathbf{x}} = \begin{cases} \mathbf{0}_{n_i}, & \text{for } i \in \{h_1, h_2, ..., h_T\}, \\ \partial D^i_{\mathbf{x}}, & \text{for } i \in \{h_{T+1}, h_{T+2}, ..., h_L\}. \end{cases} \tag{4}$$

Moreover, we expect the final trained model's test accuracy to be not influenced by the privacy protection technique. Therefore, we propose to alternate the layers to be pruned for every K training epochs. In other words, during the model update, which has several training epochs, the set of layers to be pruned will be changed after every K training epochs. For simplicity, we increase the pruning index by 1 if the set of layers to be pruned is the same after K training epochs. For example, let $\mathbf{c}^{K-1} = \{h_1, h_2, ..., h_T\}$ be the set of layers to be pruned at the $K-1$th training epoch. If $\mathbf{c}^K = \mathbf{c}^{K-1}$, we replace h_1 by h_{T+1} in $\mathbf{c}^K$, so that $\mathbf{c}^K = \{h_2, h_3, ..., h_{T+1}\}$. As a result, for the next K training epochs, from Kth to $2K-1$th epoch, the set of layers to be pruned are not the same as the preceding K training epochs.

Algorithm 2 Sequential Update by an Attacker

Step 1: Observe the $M+1$th set of gradients and updated models.
Step 2: Select an image from the M reconstructed images as the initialized dummy image $\hat{\mathbf{x}}$ by (6).
Step 3: Construct a multi-loss objective function $\mathcal{L}$ as in (5).
Step 4: Find the gradients $\partial \mathcal{L}/\partial \hat{\mathbf{x}}$.
Step 5: Perform stochastic gradient descent to optimize the trainable dummy image and save the results.
Step 6: Wait until the next set of gradients. Then, go back to Step 1.

B. Sequential update

1) Multi-loss objective function: Previous section introduces a privacy protection method from the perspective of a local device. In this section, we propose a sequential update method for the attacker to reconstruct the source images with improved quality. We assume that the attacker has full knowledge of the gradients shared by the local device during the model update. Let $\partial W_{\mathbf{x}_i}$ be the ith shared gradients of training batch $\mathbf{x}$ on a model $\mathcal{M}_i$. Suppose the attacker has obtained and saved M sets of shared gradients during the training epochs. A multi-loss objective function can be constructed by

$$\mathcal{L} = \sum_{i=1}^{M} \gamma^{i-1} \left(1 - \frac{\langle \partial W_{\mathbf{x}_i}, \partial W_{\hat{\mathbf{x}}} \rangle}{||\partial W_{\mathbf{x}_i}||_2 ||\partial W_{\hat{\mathbf{x}}}||_2}\right) + \alpha \text{TV}(\hat{\mathbf{x}}), \tag{5}$$

where γ is a scaling factor to control the contribution of each saved shared gradients of the ith model $\mathcal{M}_i$.

2) Dummy image initialization: On top of the multi-loss objective function, we propose to adjust the initialization of dummy image $\hat{\mathbf{x}}$ by selecting the saved reconstructed image which has the smallest mean-square error to all the saved set of gradients. Suppose there are $M-1$ reconstructed images from the previous saved sets of models and gradients. We use $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2, ..., \hat{\mathbf{x}}^{M-1}\}$

to indicate all the $M-1$ reconstructed images from model $\mathcal{M}_1$ to $\mathcal{M}_{M-1}$ and $\partial W_{\hat{\mathbf{x}}_j^i}$ represents the gradients of the ith reconstructed image from the jth saved model. Then, we propose to initialize the dummy image by

$$\hat{\mathbf{x}} = \underset{\hat{\mathbf{x}}^i}{\operatorname{argmin}} \left\{ \sum_{j=1}^{M-1} ||\partial W_{\mathbf{x}_j} - \partial W_{\hat{\mathbf{x}}_j^i}||_2 \right\}, \tag{6}$$

where $i \in \{1, 2, ..., M-1\}$ refers to the index of previously saved models. Note that the attacker is assumed to have saved access to the previous $M-1$ models. For each reconstructed image $\hat{\mathbf{x}}^i$, the attacker can generate $M-1$ sets of gradients from all the saved models.

All the previous proposed methods for data leakage protection and image reconstruction are summarized in Algorithm 1 and 2.

IV. Numerical Results

In this section, we evaluate the model test accuracy and attacker reconstruction performance across the model training progress.

We use SSIM [10] to measure the accuracy of image reconstruction by the attacker. To evaluate the attacker reconstruction performance, we match each reconstructed image to the source mini-batch by:

$$\hat{x}_i = \hat{x}_{\operatorname{argmax}\{\text{SSIM}(x_i, \hat{x}_j) \mid j=1,2,...,B\}}. \tag{7}$$

This is due to the fact that with a mini-batch of size greater than 1, which might contain images that have the same labels, the reconstructed images might not follow the original order as in the source data. Therefore, when we measure the average SSIM, we match each reconstructed image to one of the ground-truth images, which has the largest SSIM.

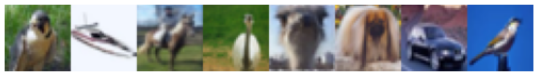
In our experiments, ResNet18 is employed as the neural network model for image classification, and CIFAR-10 is the data set. There is no pre-processing on the source images before training or testing. We use 5 different batches with a size of 8 to evaluate the attacker's reconstruction performance, where the average SSIM across the 5 batches is recorded. The gradients of a local device are assumed to be shared after every 5 epochs of training. In total, there are 40 training epochs for the local device. In Figs. 2 and 3, "EW" and "LW" represent the conventional element-wise threshold-based pruning and layer-wise pruning, respectively. Moreover, we use "Seq" to represent the sequential update method that is introduced in Section III-B. In comparison, "Static" represents the reconstruction method without saving all of the previous models and shared gradients. The attacker is assumed to know only the model and shared gradients from the current training epoch. To demonstrate the overall test accuracy and reconstruction performances during model update, in Table I, "AUC of TA" and "AUC of SSIM" refer to the area under the curves (AUC) of test accuracy (TA) and SSIM, respectively. "Final TA" refers the final model's test accuracy on classifying the CIFAR-10 test data set.

TABLE I: Test accuracy and SSIM during the training of a ResNet18

Pruning Methods	AUC of TA	AUC of SSIM	Final TA
No pruning	0.79	0.73	82.5
LW $T=1$	0.80	0.54	82.3
LW $T=2$	0.80	0.35	83.1
EW $p=0.21$	0.79	0.70	81.9
EW $p=0.43$	0.79	0.67	82.0

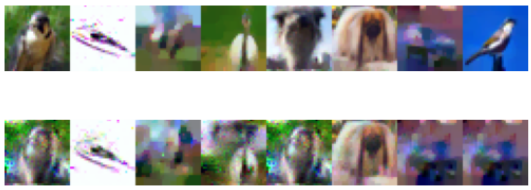
In Fig. 2, there are 4 curves with different pruning parameters shown, layer-wise pruning with $T=1$ and $T=2$ and element-wise pruning with $p=0.21$ and $p=0.43$. We choose $p=0.21$ and $p=0.43$ to have the same number of pruned parameters as the layer-wise pruning with $T=1$ and $T=2$, respectively. We observe that both element-wise and layer-wise pruning methods can achieve a close performance to the trained model without pruning. Moreover, in Fig. 3, we can observe that the SSIM of layer-wise pruning is much smaller than the element-wise pruning method at all the training epochs. Without the proposed sequential update method, the reconstruction performance by the attacker is poor. As analysed in Table I, the total AUC of SSIM of layer-wise pruning is 0.35 and 0.54 for $T=1$ and $T=2$, respectively. These values are much smaller than the AUC of SSIM of element-wise pruning, which has 0.7 and 0.67 for $p=0.21$ and $p=0.43$. Note that the AUC of SSIM of layer-wise pruning with $T=1$ is much smaller than the element-wise pruning with $p=0.43$, which has twice number of pruned parameters.

On top of the test accuracy and reconstruction performance, in Fig. 1, we show an example of reconstructed image in comparison between element-wise pruning with $p=0.43$ and layer-wise pruning with $T=2$. We observe that with the implementation of sequential update, the image becomes more and more accurate with the increase of training epoch for the layer-wise pruning. The SSIM with $T=2$ eventually reaches a SSIM of 0.3, where the attacker is hard to identify all the images. While for an element-wise pruning, all the 8 images are relatively accurate to be identified.

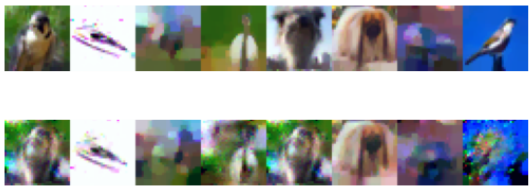


(a) Original test images

(b) Training epoch index: 0. EW pruning with $p = 0.43$ has a SSIM of 0.48. LW pruning with $T = 2$ has a SSIM of 0.13.



(c) Training epoch index: 20. EW pruning with $p = 0.43$ has a SSIM of 0.79. LW pruning with $T = 2$ has a SSIM of 0.28.



(d) Training epoch index: 40. EW pruning with $p = 0.43$ has a SSIM of 0.75. LW pruning with $T = 2$ has a SSIM of 0.30.

Fig. 1: Examples of attacker's reconstructed images using the sequential update method. The figures above refer to the reconstruction results with layer-wise pruning of $T = 2$. The figures below refer to the element-wise threshold-based pruning with $p = 0.43$, which has the same number of pruned parameters as the layer-wise pruning.

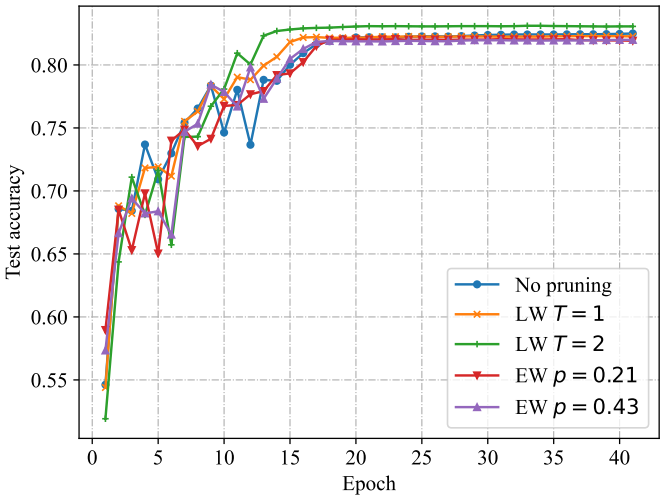


Fig. 2: Test accuracy during 40 epochs of training

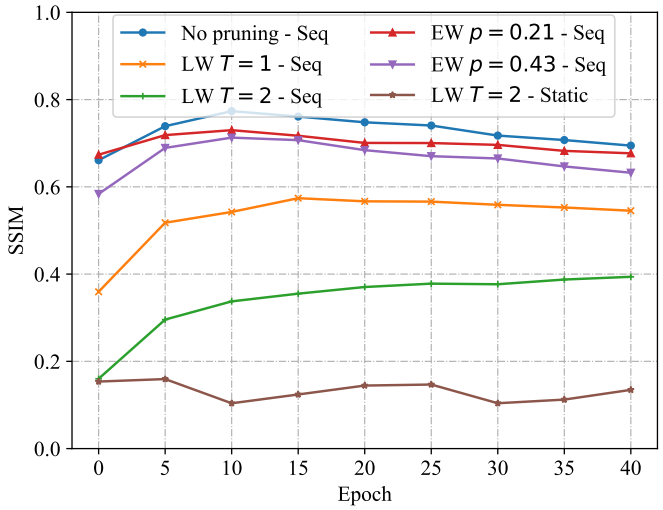


Fig. 3: Averaged SSIM of reconstructed images.

V. Conclusion

In this paper, we studied a data leakage problem in a federated learning system. We proposed a privacy protection method for a local device by zeroing out the full layer's gradients before sharing the gradients to the central server. By comparing to the element-wise threshold-based pruning method, we show that layer-wise pruning can achieve a much better protection on the source data. On top of that, we assume that an attacker might have the full knowledge of shared gradients during the model update. As a result, the attacker can sequentially combine all the models and shared gradients' information at each training epoch to achieve a better reconstruction performance on the source data.

References

[1] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv: 1610.05492*, 2016.

[2] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[3] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," *arXiv: 2001.02610*, 2020.

[4] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients – how easy is it to break privacy in federated learning?" *arXiv: 2003.14053*, 2020.

[5] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[6] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," 2021.

[7] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," *arXiv: 2112.00059*, 2021.

[8] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," *arXiv:2004.10397*, 2020.

[9] E. Candes, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *arXiv: 0503066*, 2005.

[10] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conf. Sig., Sys. Comp., 2003*, vol. 2, 2003, pp. 1398–1402 Vol.2.