Inverting Gradients Report

Smriti Das
smritidas@umass.edu
UMass Amherst
Amherst, Massachusetts, USA

ABSTRACT

The proliferation of mobile devices and the sources of data in the past few years has paved way for decentralized training of neural networks. Federated Learning is this decentralized way in which the data is located at the source locally and only the model updates are sent to the global model located at remote server [4]. One of the several benefits of this is method of training is user data privacy because no data is being shared by the users. But the important question is whether privacy can be compromised and the original data can be recovered by obtaining the updates sent to the server. There are research works that show it is not possible to break privacy by only using the updates while there are others that prove the opposite. These works make certain assumptions, and the key is to gauge how realistic are the assumptions to validate the theoretical arguments on whether or not Federated Learning assures privacy. In this report, I analyze if Federated Learning really does not guarantee privacy because the data can be reconstructed by inverting the gradient information sent to the server for model updates by implementing (code) the proposed cost function by [2].

CCS CONCEPTS

• **Machine Learning** → **Federated Learning**; • **Security** → *Data Privacy*; • **Networks** → Network reliability.

KEYWORDS

data-sets, neural networks, euclidean loss, optimization techniques, cosine similarity, cost function, threat model, privacy, honest-but-curious server

ACM Reference Format:
Smriti Das. 2022. Inverting Gradients Report. In *Proceedings of Class Report (Report'22).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Deep Learning(DL), a subset of Machine Learning(ML), has been producing some stirring results in recent times for industrial application as well as in several research domains. This was only possible because of the modern time hardware's ability to process large amount of data. The source of all this data is distributed across variety of platforms across industries. For some industries like healthcare, obtaining patient data is a difficult because people may not give consent for several reasons. This might hamper the chance of capitalizing on the usefulness of DL models in healthcare. In general the data from multiple sources are collected and stored in a central server which is accessible to train the DL models, that certainly creates possibility of data leakage.

Report'22, December 2022, UMass Amherst, USA
©

In Federated Learning (FL), the participating devices or clients form a *federation* to collaboratively train a global model [3] as illustrated in Fig. (1). The devices download the current global model, each client with their local data-set sends the gradients or parameter updates back to the server where the server updates the global model using one of the several FL paradigms. This setting can help improve the model training by leveraging data which could not be otherwise shared for the centralized model training because of its sensitive nature. This is where ensuring privacy becomes important.

The general idea of training a neural network includes optimizing parameters θ of a neural network using a loss function L over the training set x_i, y_i, where x_i are features and y_i are labels and N is the total number of data points in the training set. The goal to minimize the overall loss and get a set of parameters θ for which that minimization occurs can be expressed as (1) [2]. Loss function computes the difference between current predictions given by the model and actual labels for the training set.

$$min_\theta \sum_{i=1}^{N} L_\theta(x_i, y_i) \tag{1}$$

In Federated SGD setting, each client has their own data-set, the calculated gradients by each client is sent to the server and the parameter update that takes place at the server can be expressed as (2) [2], where N training data points are distributed amongst any M number of clients and α is learning rate. Once the update has been done at server, the θ_{k+1} is sent to all the clients. The main idea in this case is - the clients share gradients, $L_{\theta^k}(x_i, y_i)$, for model update.

$$\theta_{k+1} = \theta_k \; - \; \alpha \sum_{i=1}^{N} \nabla_\theta L_{\theta^k}(x_i, y_i) \tag{2}$$

Federated Averaging is another paradigm in which the clients perform several parameter updates locally using (3) [4], send the parameter updates to the server where the averaging takes place and the averaged parameter update (4)[4] is shared back with the clients. The value of $\frac{n_m}{n}$ represents the fraction of data points at *m th* client. This paradigm can help hide the information about data.

$$\theta_{k+1}^{m} = \theta_k \; - \; \beta \nabla_\theta L_{\theta^k}(x_i, y_i) \tag{3}$$

$$\theta_{k+1} = \sum_{m=1}^{M} \frac{n_m}{n} \theta_{k+1}^{m} \tag{4}$$

$$\frac{1}{t} \sum_{i=1}^{t} \nabla_\theta L_{\theta^k}(x_i, y_i) \tag{5}$$

FL has several advantages as mentioned by [4] but privacy is by far

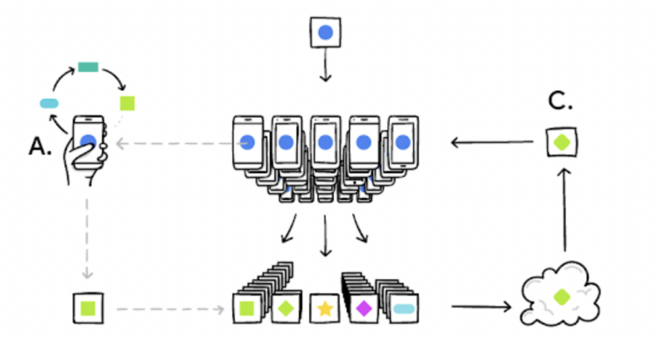

Figure 1: Federated Learning outline

the most important one because this allows for access to more data and expanded Machine Learning applications because of access to sensitive data for model training. Centralized data storage as required in common settings requires data to be always available in the data center and even anonymity can not protect the user identity because information can be joined from multiple databases to infer important details. Given the use of Machine Learning has been increasing in medical, bio-tech, chemical industries - privacy is becoming increasingly important.

There are several works which show that FL allows for data privacy given the client never shares data in a FL setting rather only gradients or parameter updates are shared. But [2] argue that gradients and parameter updates also contain enough information to allow for reconstruction of original data which was used for the update. Their analysis is based on single image and multi-image recovery from the gradient information. They also analyze the recovery of images from the parameter updates as in case of federated averaging. Although, their analysis is limited to image data recovery.

We train DL models for several purposes and the most common technique while training is back propagation. The other purposes may include use in Natural Language Processing where we use texts, audio waves for training voice assistants and DL attacks for Side Channel Analysis which uses power signals. And the models trained for the aforementioned purposes may use back propagation and gradient updates. So, the method that proves that it is possible to break privacy in FL setting by using gradient information should be successful for different input data types.

FL is not limited to training image classification models. The data can be signals and text as well. In this paper I perform the same analysis as done by [2] but on the signal data. The threat model defined is same as [2] where honest-but-curious server is trying to recover the input data from the gradient or parameter updates obtained from different clients. There is *no* interference with the FL training process and the assumption is that the server carries out the recovery process such that the regular collaborative training is not impacted in any way. No modification to the global parameters and architecture is allowed in order to improve the possibility or quality of recovery. Network latency and the client settings is ignored. The main focus is only on the reconstruction algorithm.

In the next few sections I discuss about the previous works on the data recovery using gradients. Next, I detail the theoretical analysis of how the data can be recovered from gradients in fully connected networks no matter the location of fully connected network within the architecture. Finally, I discuss the experiments, results and outcomes for the image as well as power signal data where I implement deep as well as shallow architectures in trained and untrained model. Empirical result show it is not feasible to generate signal input data from the gradients.

2 RELATED WORK

As mentioned in [main] there have been works on information recovery using gradients but the assumption of training a shallow model is not fitting in the real world scenarios. The architecture of model being trained plays a role in deciding whether gradients carry enough information for recovering data consequently if privacy is a reasonable supposition. As noted in [main] earliest works of recovering an image from gradient information by used shallow models with large fully-connected layers to create a representation of the images and use GANs on the representation to improve the image. Few follow up works showed the multi-image recovery from averaged gradients is also possible. All the works rely heavily on the limited parameters or shallow networks which makes the adversary's task easier, this violates the threat model for which I analyze the information reconstruction. Works on deeper network architecture failed to produce useful results.

The formulation for recovery by [6],[8] used euclidean matching optimization which solves for x based on the cost function as mentioned in (6)[2] which is solved using L-BFGS solver. In this approach, to find the minimum difference between the gradient of the actual image and the image which will be the output, supposedly the same image as input image, we need to solve for the higher order derivatives of the loss function which would be difficult because of the discontinuity of higher order derivatives due to non-linear activation functions.

$$argmin_x||\nabla_\theta L_\theta(x,y) - \nabla_\theta L_\theta(x*,y)||^2 \tag{6}$$

In the [2] paper their work depends on the similarity of gradients - the detailed explanation of which is presented in the next section - and not minimizing the difference between gradients. They use cosine similarity to calculate and find x based on the equation (7) [2]. Here total variation (TV) is the term which compensates for the noise in the image. The TV term has been found to be useful for images recovery. TV is a mathematical concept which is largely used in the context of denoising the data, especially image and signal data. In my case I am not changing anything in the cost function becuase the denoising term would be useful for signals as well. I apply the proposed similarity cost function for signals to check if the proposition by [2] holds for data other than images. Their work has proven that gradient inversion to recreate images is easier

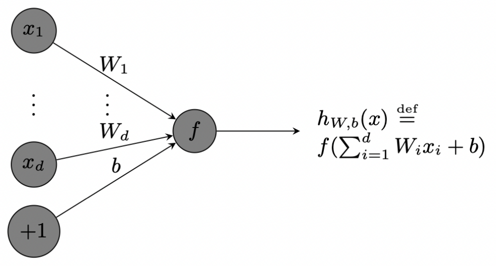
x_1
W_1
W_d
x_d
b
$+1$
f
$h_{W,b}(x) \overset{\text{def}}{=} f(\sum_{i=1}^{d} W_i x_i + b)$

Figure 2: Single Neuron.

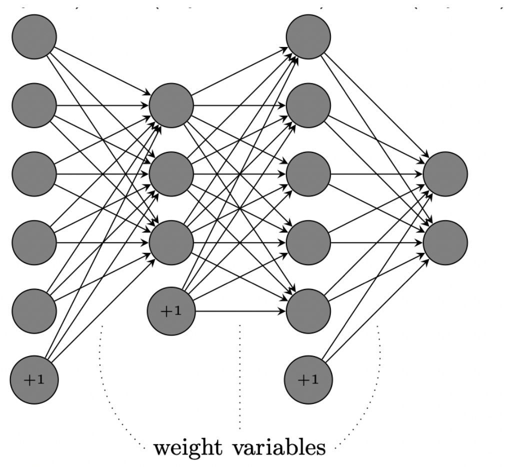

Figure 3: Fully Connected Layers

than using visual representation and parameters to recreate images.

There have been several works to retrieve the information from the middle layers to get chunks of information about the overall image but not the actual data using local update information. Even though it could be said that these partial information doesn't necessarily reflect the actual data hat the adversary is looking for but it is still solving sub-problems. My analysis will not focus on the retrieval of information from intermediate layer. A recent work after [2] by [7] focuses on how the data diversity can have an impact on gradient inversion after federated averaging and claim that the full image recovery is possible for batch-averaged gradient descent for deep neural networks by adding a regularization term to the cost and multi-seed optimization.

3 THEORETICAL ANALYSIS

Let us analyze the theory of gradient inversion to understand how are gradients carrying enough information about the original data to be able to recover the original data. Take an example of one neuron as shown in Fig. (2) [5]. $x_i (1 \leq i \leq d)$ are the real number input to the neurons, $W_i (1 \leq i \leq d)$ are the weights, f is the non-linear activation function, y is the actual output and b is bias. Loss is the difference between the actual output and predicted output $h_{W,b}(x)$.

$$h_{W,b}(x) = f(\textstyle\sum_{i=1}^{d} W_i x_i + b) \tag{7}$$

$$L(W, b, x, y) = (h_{W,b}(x) - y)^2 \tag{8}$$

$$\frac{\delta L(W,b,x,y)}{\delta W_k} = 2(h_{W,b}(x) - y)\, f'(\textstyle\sum_{i=1}^{d} W_i x_i + b) x_k \tag{9}$$

$$\frac{\delta L(W,b,x,y)}{\delta b} = 2(h_{W,b}(x) - y)\, f'(\textstyle\sum_{i=1}^{d} W_i x_i + b) \tag{10}$$

When trying to recover an image from a gradient means we are trying to get $x \in R^n$ from $\nabla_\theta L_\theta(x, y) \in R^p$ meaning we are wither reducing the dimensions or increasing the dimensions of a vectors available to us which is a difficult task because either we need to remove and manipulate the values in such a way that useful information remains or add values to the current vector which helps generate new useful information from current information. This task is somehow easier than expected in the fully connected layers. We know that the gradients are shared with the server. Hence from (9)[5] and (10)[5] we can see that the ratio of 2 gradients gives us the kth input feature with some proportionality constant. Similarly we can find all the input features. It is easy to see for one neuron but the same "proportional" result can be generalized for general neural networks shown in Fig. (3) [5] with generalization as in example 3 of [5] as mentioned in (11)[5] and (12)[5]. Also, the proposition 3.1 in the [2] follows from the [5] which claims any input to the fully connected layer can be reconstructed if the derivative of L is known for the output of that layer and at least one of the derivatives is non-zero, this is independent of the position of the fully connected layer.

$$\frac{\delta L(W,b,x,y)}{\delta W_{ik}^{(1)}} = \xi_i x_k + \lambda W_{ik}^{(1)} \tag{11}$$

$$\frac{\delta L(W,b,x,y)}{\delta b_i^{(1)}} = \xi_i \tag{12}$$

Most of the neural networks have fully connected layers as the last layers but not the first layers connected to the input. From the propositions mentioned above the input to fully-connected layers could be reconstructed but not the inputs directly. Even though the visual representation from intermediate layers in architectures like CNN for images could give information about the actual image, the work of [2] focuses on actual input regeneration. Gradient contains information regarding the impact of one data point on the prediction model. So, the similarity of the current input x^* gradient and for some input x's gradient can be calculated to find the input that gives most similar gradient as the original input gradient, meaning it has the same effect on the model as the current input. This can be implemented using (13)[2].

$$argmin_{x \in [0,1]^n} [1 - \frac{<\nabla_\theta L_\theta(x,y), \nabla_\theta L_\theta(x*,y)>}{||<\nabla_\theta L_\theta(x,y)||||\nabla_\theta L_\theta(x*,y)||}] + \alpha \cdot TV \tag{13}$$

The above cost function as proposed in [2] for numerical reconstruction takes into account the limitations posed by early implementations that worked for simpler architecture which isn't realistic. The gradient can be considered as magnitude and direction independently. As observed in [2] the magnitude defines the local state of optimality during training and direction captures the change in current model with different data points. The overall cost function is not helping to find the image that fits the gradient of actual data but is trying to identify the image which can bring the same change in the model as the actual image. TV is the total variation factor which in image processing helps to remove the noise from an image. The factor of "noise" can be observed when dividing (11) and (12). Although it is not noise in reality but during reconstruction, an additional term with $W_{ik}^{(1)}$ or regularization can act as noise to the data.

4 EXPERIMENTS AND RESULTS

The main focus in this report is to implement the algorithm or cost function of the [2] paper to check if FL ensures privacy when only gradients are shared for updating the model. This experimental setting includes steps the honest but curious server performs after receiving the gradient information to extract the input data information. The implementation ignores the client server setting entirely. One data point from data set is chosen, the gradient for the data point is calculated for selected model architecture. This is then fed into the reconstruction algorithm to obtain the recovered data by the algorithm. In the multi-image setting, multiple data points, say N data points are randomly chosen which reflect the case where each client is sending one data point gradients in case of Federated SGD, this is implemented using loop instead of client server setting.

I use the same proof of concept as given in [2]. First I implement the reconstruction algorithm as explained in [2] in the single image single gradient setting on trained and untrained network. This reflects a setting in which all the devices use same global model be it trained or untrained, and use same image to calculate the gradient information keeping the model architecture same. Then all the devices will send the same gradient information back to the server. I repeat this experiment for 2 data-sets - FashionMNIST and tinyAES(STM32F3).

Further the architectures used for the analysis of FashionMNIST are: *Shallow Neural Network*, with one input, one hidden and one output layers with ReLU activations and logits. The second architecture is *Fashion Net* which contains one input, one hidden and one output fully connected layers and ReLU activation function but with 0.2 dropout before output for regularization. Third architecture is *ConvNet* which contains two 2d-convolution layers, then max-pooling, two fully connected layers and finally the output layer with 0.2 dropout. The accuracy of *Fashion Net* is 87% and *ConvNet* is 91.58%. The loss function used everywhere is Cross Entropy Loss and optimizer is SGD for all the results shown here, the results vary slightly but not by much for using Adam optimizer. I used both trained and untrained network here because we can [2] observe that when the model is trained and if an image from training set is the image generating gradient, i.e. the user is sending the same image as in training set to update the model, the is a chance that all the gradients can be zero which is the case we don't want. In the untrained model's case the weights are randomly initialized and there is a very small chance that all the gradients sent will be zero.

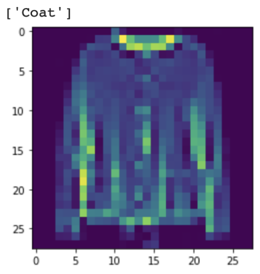

Figure 4: Input 'Coat' image from FashionMNIST data-set for which gradients are calculated. This is the image I am trying to recover using inverting gradients algorithm

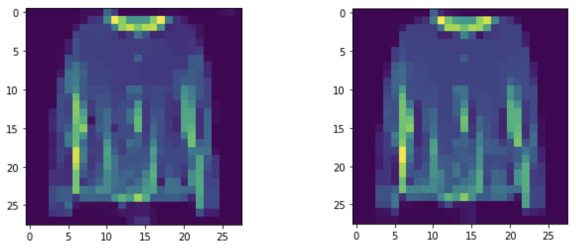

Figure 5: Recreated 'Coat' images for Shallow Neural Network for my algorithm implementation (on left) and the original algorithm implementation (on right)

The architecture I have used for experimenting on the power signal is the same as *Shallow Neural Network* but the actual implementation of deep learning SCA uses ResNet as mentioned by [1]. Because here I am verifying if we can actually recover data from gradients or not, I started with a shallow architecture.

4.1 Datasets

The first data set I used for experimentation is FashionMNIST data set for image as input test. This dataset contains 70,000 examples, each is 28*28 with associated labels from 10 classes. For training I used 60,000 data points with validation split of 20% and for testing I used 10,000 data points. The other data-set I used is TinyAES. This data set contains power signals for SCA of a cryptographic device implementing AES. This dataset is used to perform deep learning attack for obtaining the AES key implemented by the device. To be specific, STM32F3 is the dataset and can be found [1]. This is the first data-set collected on ChipWhisperer-Lite device with a cryptographic board performing AES-128 encryption. Each power data-point has 80,000 samples from which we can infer the number of rounds of AES. For my experiments I have only considered 11th byte data and only 1000 sample points of each data point.

4.2 Result Analysis

The images recovered by shallow untrained networks in Fig. (5) are the closest to the actual image. As the network gets complex, for example in the case on Fashion Network in Fig. (6), (8) as well as

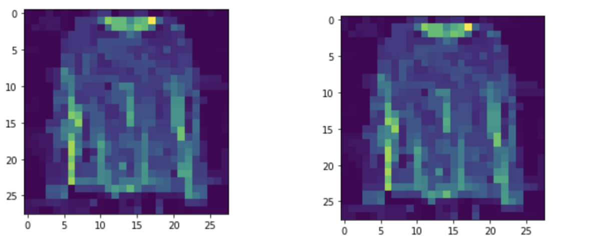
Figure 6: Recreated 'Coat' images for untrained Fashion Net for my algorithm implementation (on left) and the original algorithm implementation (on right)

ConvNet the recovered images in Fig. (7), (9) are not close to the original image. In case of Fashion Network the untrained model Fig. (6) gave not so clear output but the trained model recovered an image as shown in Fig. (8), which looks better in quality to a human but the result is not close to the original image, that means it failed to achieve the goal. As I slightly increased the architecture complexity, the results generated by trained as well as untrained model did not give any useful output. This shows that the [2] paper's claim that their proposed algorithm can handle the increased model complexity of the model as well as regularization included, as in real world scenario may not be true under certain conditions like the quality of the data, in this case image, that is being recovered using gradients. The quality of images that was used to present the results in [2] even for deeper architectures was significantly better than the images used in my experiments, choosing the bad quality images was intentional.

In the other experiment I use one power signal as shown in Fig. (10) from the tinyAES data-set and use it's gradient to reconstruct the signal back. The output I got was not satisfactory. I used both, the original algorithm as given by [2] along with my implementation of their cost function for the analysis. The outputs for both the implementations are similar but entirely different from the original input as illustrated in Fig. (11). As mentioned earlier I used a shallow network for this experiment because it is easier to reconstruct data using gradients in the shallow networks. But the result may be true for images, it certainly is not true for signals. I did not implement the federated averaging algorithm because we already know that there are less chances of successfully recreating images in case of federated averaging as compared to Federated SGD case. And given the results I got for federated SGD in case of images containing less information and signals, I suppose that the implementing federated averaging will not produce satisfactory results. Thus, making the case that privacy can be ensured in FL in most cases with the strict threat model considered for this implementation, the code for which can be referenced here.

5 CONCLUSION

Since the model being trained on the server can be anything - images, texts or voice. Saying that FL does not provide privacy would be wrong. There are multiple nuances and conditions under which FL can or can not assure privacy. In the above analysis we found that the images could be recovered to an extent, thus compromising the data privacy, which is the key feature of FL. But for data other than images the method adopted to retrieve the original data information was not close to breaking the privacy because useful information about the original data could not be gathered. One can argue that

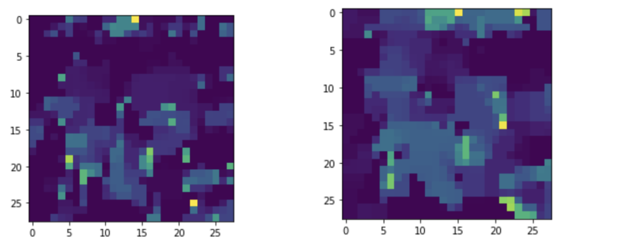
Figure 7: Recreated 'Coat' images for untrained ConvNet for my algorithm implementation (on left) and the original algorithm implementation (on right)

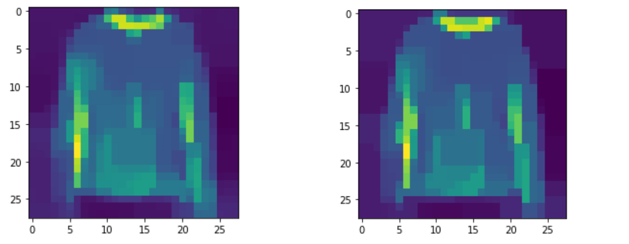
Figure 8: Recreated 'Coat' images for trained Fashion Network for my algorithm implementation (on left) and the original algorithm implementation (on right)

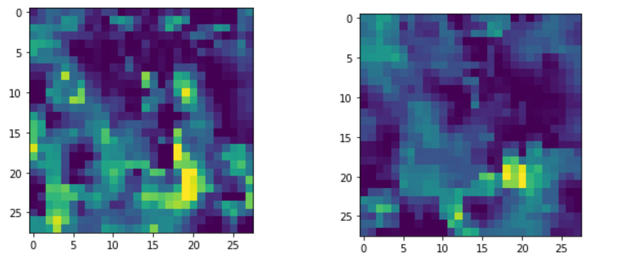
Figure 9: Recreated 'Coat' images for trained ConvNet for my algorithm implementation (on left) and the original algorithm implementation (on right)

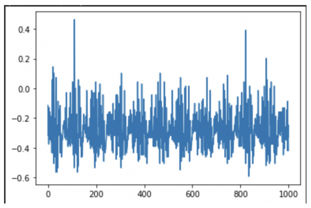
Figure 10: Input power signal for which the gradient is calculated

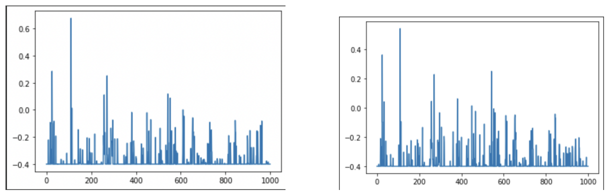

Figure 11: Recreated signal for untrained Neural Network for my algorithm implementation (on left) and the original algorithm implementation (on right)

there could be multiple reasons for that, the most important being the [2] proposed cost function being specific for images and not other kinds of data. The future works can focus on coming up with the cost functions which can possibly accommodate different data types or coming up with different cost functions or variations of the cost function as proposed by[main] for different types of data. Also, we already know that federated averaging obscures data. So, unless the data privacy offered by Federated Averaging can be compromised we should continue to find a way to do so.

REFERENCES

[1] Elie Bursztein et al. *SCAAML: Side Channel Attacks Assisted with Machine Learning*. 2019. URL: https://github.com/google/scaaml.

[2] Jonas Geiping et al. "Inverting Gradients - How easy is it to break privacy in federated learning?" In: *CoRR* abs/2003.14053 (Sept. 2020). DOI: 10.48550/ARXIV.2003.14053. URL: https://arxiv.org/abs/2003.14053.

[3] Brendan McMahan and Daniel Ramage. *Stats and Analysis*. Apr. 2017. URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[4] H. Brendan McMahan et al. "Federated Learning of Deep Networks using Model Averaging". In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629.

[5] Le Trieu Phong et al. "Privacy-Preserving Deep Learning via Additively Homomorphic Encryption". In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[6] Zhibo Wang et al. "Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning". In: *CoRR* abs/1812.00535 (2018). arXiv: 1812.00535. URL: http://arxiv.org/abs/1812.00535.

[7] Hongxu Yin et al. "See through Gradients: Image Batch Recovery via GradInversion". In: *CoRR* abs/2104.07586 (2021). arXiv: 2104.07586. URL: https://arxiv.org/abs/2104.07586.

[8] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. "iDLG: Improved Deep Leakage from Gradients". In: *CoRR* abs/2001.02610 (2020). arXiv: 2001.02610. URL: http://arxiv.org/abs/2001.02610.