Regularizing Gradient Reconstruction Attacks in Federated Averaging

Pierre JOBIC Aurélien MAYOUE Sara TUCCI-PIERGIOVANNI
Université Paris-Saclay, CEA List, F-91120, Palaiseau, France

Résumé – Pour répondre aux besoins grandissants d'accès aux données et de confidentialité, l'apprentissage fédéré est de plus en plus considéré. Il permet à plusieurs clients d'apprendre un modèle de façon collaborative sous la supervision d'un serveur central mais sans jamais partager leurs données. Par construction, l'apprentissage fédéré apparait comme une solution pour assurer la confidentialité des données mais pas celle du modèle dont les paramètres sont échangés entre les clients et le serveur tout au long du processus d'apprentissage. Des attaques récentes, telles que les attaques par reconstruction de gradient (GRAs), ont ainsi montré la faisabilité de reconstruire les données d'entraînement des participants. Néanmoins, la majorité de la littérature s'intéresse à FedSGD, une version simplifiée et non réaliste de l'apprentissage fédéré où chaque client réalise une seule itération de descente de gradient stochastique à chaque tour. Dans un scénario plus réaliste, les clients calculent plusieurs étapes de descente de gradient (FedAvg) avant de partager leur modèle au serveur. Ce protocole ajoute des étapes de calculs intermédiaires, inconnues de l'attaquant, rendant ainsi les GRAs moins efficaces. Dans cet article, nous introduisons une nouvelle régularisation qui rend les GRAs plus efficaces dans le cadre de FedAvg. Notre discussion est étayée par des expériences dans un contexte de vision par ordinateur.

Abstract – Federated Learning (FL) has gained prominence as a decentralized and privacy-preserving paradigm that enables multiple clients to collaboratively train a machine learning model under the supervision of a central server but without sharing their data. By design, FL is a solution for data privacy but not model privacy. Recent attacks, such as gradient reconstruction attacks (GRAs) have precisely shown privacy issues when an attacker knows the model parameters sent by a client to the server. In the literature, these privacy issues are mainly explored when clients compute a single gradient descent step on their data (FedSGD). In a more realistic scenario, clients compute several gradient descent steps (FedAvg). This protocol adds intermediate computation steps, which are unknown from the attacker, thus making GRAs less successful. In this paper, we introduce a new regularizer that makes GRAs more efficient under FedAvg. Our discussion is supported with experiments in computer vision.

1 Introduction

Federated Learning (FL) [9] addresses privacy concerns by keeping training data on user devices and sharing only locally-computed updates of a parametric statistical model, such as neural networks. Federated Stochastic Gradient Descent (FedSGD) [9] involves clients computing a full-batch gradient descent and sharing it with the server, but it incurs high communication overhead [9, 2]. Federated Averaging (FedAvg) [9] reduces communication rounds by having clients perform multiple stochastic gradient descent (SGD) steps before sending updates, thus accelerating learning and decreasing communication costs.

Despite FL's design to protect training data, recent attacks on FedSGD [12, 3, 11] and FedAvg [1, 4, 10, 8, 5] have inferred sensitive information from model updates. Gradient Reconstruction Attacks (GRAs) exploit these updates to infer client data passively and undetectably.

Following the recent work to design attack for FedAvg, we propose a new regularizer that can enhance all GRAs on FedAvg. Thus our **main contributions** are:

— A novel regularizer that can boost all GRAs in the context of FedAvg.
— Extensive experiments to compare our method with state-of-the-art attack [1].

2 Background and Related Work

In FL, the FedAvg [9] is a procedure to collaboratively train a machine learning model, which computes as follows:

1. **Clients Selection**: In each round t, a subset S_t of clients is selected. Each client k has a local dataset (X_k, Y_k).
2. **Communication in**: In each round t, the server sends to each selected client the global model θ_t.
3. **Model Optimization**: After the communication step, each client updates the global model θ_t by minimizing a loss function $\mathcal{L}$ with its dataset. Let n, β, η, ϵ be the dataset size, the batch size, the learning rate and the number of epochs respectively. Then each client updates the model with a stochastic gradient descent (SGD), $\forall e \in [\![1, \epsilon]\!], \forall u \in [\![1, \frac{n}{\beta}]\!]$:

$$\begin{cases} \theta_e^{u+1} = \theta_e^u - \eta \cdot \nabla_\theta \mathcal{L}(\theta_e^u; X_e^u, Y_e^u) \\ \theta_{e+1}^1 = \theta_e^{\frac{n}{\beta}+1} \end{cases} ; \text{ and } \theta_1^1 = \theta_t$$

   Where X_e^u, Y_e^u is the batch of data at step u and epoch e. We introduce the notations $U \doteq \frac{n}{\beta}$ and $\theta_\epsilon^U \doteq \textbf{FedAvg}(\beta, \epsilon, \eta, \theta_t, X, Y)$ for this algorithm.
4. **Communication out**: After model optimization at round t, each client sends its model updates $\Delta\theta_t = \theta_\epsilon^U - \theta_t$ back to the server.
5. **Aggregation**: The server aggregates these updates to compute new global parameters:

$$\theta_{t+1} \leftarrow \theta_t + \frac{1}{|S_t|} \sum_{k \in S_t} \textbf{FedAvg}(n, \beta, \epsilon, \theta_t, X_k, Y_k) - \theta_t$$

6. **Communcation rounds**: The step 1. to 5. are repeated for a number of communications rounds T.

In the specific case of FedSGD, each client performs only one epoch ($\epsilon = 1$) and uses the entire dataset as a single batch ($\beta = n$). This results in a single update step per round.

Threat Model In this paper we consider the case of a **honest-but-curious** server, which has access to the client's communication θ_ϵ^U and θ_t and the model architecture, i.e. a **white-box** model. The hyperparameters (HP) of the clients η, β, n are known from the attacker.

Gradient Reconstruction Attacks (GRAs) In spite of the privacy-by-design architecture of FL to protect clients' training data, it is still possible for an attacker to reconstruct crucial information on these training data. GRAs are the simplest attack to implement yet powerful. In order to attack, GRAs will construct data points $\tilde{X}, \tilde{Y}$ which will mimic the observed parameters update $\theta_t, \theta_\epsilon^U$. Namely:

1. **Initialization**: initialize $(\tilde{X}, \tilde{Y})$ (often sampled from random normal distribution)
2. **Optimization**: Repeat until convergence:
   — **Simulation**: $\tilde{\theta} = \textbf{Sim}(\beta, \epsilon, \eta, \theta_t, \tilde{X}, \tilde{Y})$ where **Sim** is any optimization algorithm on the parameters θ (e.g. **FedAvg** as in step 3 of dotted list).
   — **Gradient Descent**: any gradient descent algorithm on $(\tilde{X}, \tilde{Y})$ by minimizing a loss function $\textbf{Loss}(\theta_\epsilon^U, \theta_t; \tilde{\theta}, \tilde{X}, \tilde{Y})$.

All the trick is to choose **Sim** and **Loss** wisely.

Loss functions can be decomposed in two terms:

$$\textbf{Loss} = \textbf{dist}(\theta_\epsilon^U, \tilde{\theta}) + \textbf{reg}(\theta_\epsilon^U, \theta_t; \tilde{\theta}, \tilde{X}, \tilde{Y}) \tag{1}$$

Where **dist** is a distance function, such as euclidean distance [12] or cosine similarity [3]. This equation (with or without the **reg**) is not convex, there is not a global minimum but a lot of local minima, but only a few are interesting. That is why, a regularization term **reg** is often added to select the solutions of interest.

For instance, [3] use the Total Variation (TV) which characterizes the smoothness of an image. This (positive) metric is often very low on real images. It allows $(\tilde{X}, \tilde{Y})$ to converge to more realistic images.

GRAs in the litterature GRAs have mainly been designed for the FedSGD cases, where the clients communicate only the gradients. However in a more realistic scenario, clients use FedAvg. As stated by [1], GRAs are harder on FedAvg because of the unknown intermediate local updates θ_e^u that are computed by the clients. To address this, [4, 10] propose to attack FedAvg by simulating FedSGD. More concretely, $\tilde{\theta} = \textbf{Sim}(\beta, \epsilon, \eta, \theta_t, \tilde{X}, \tilde{Y}) = \text{FedAvg}(n, 1, U \cdot \epsilon \cdot \eta, \theta_t, \tilde{X}, \tilde{Y})$. Note that the learning rate in the simulation is $U \cdot \epsilon \cdot \eta$ to compensate the single update step that the attack performs compared to the $U \cdot \epsilon$ steps that the client performs.

It is also possible to attack FedAvg by simulating **FedAvg**: $\tilde{\theta} = \textbf{Sim}(\beta, \epsilon, \eta, \theta_t, \tilde{X}, \tilde{Y}) = \text{FedAvg}(n, \epsilon, \eta, \theta_t, \tilde{X}, \tilde{Y})$. Simulating **FedAvg** to attack FedAvg is what [3] proposed in a slightly different context.

Then [1] attacked FedAvg by introducing a new **Sim** in accordance with a new **reg**. The new **Sim**, which we call **FedAvg-Epoch**, introduces new dummy sets $(\tilde{X}_e, \tilde{Y}_e)_{e \in [\![1,\epsilon]\!]}$ for each simulated epochs. Each dummy sets are fed into the simulation **FedAvg** sequentially. To be specific, $\forall e \in [\![1, \epsilon-1]\!]$

$$\begin{cases} \tilde{\theta}_{e+1} = \textbf{FedAvg}(n, 1, \eta, \tilde{\theta}_e, \tilde{X}_e, \tilde{Y}_e) \\ \tilde{\theta}_1 = \theta_t \end{cases}$$

This simulation returns $\tilde{\theta}_\epsilon$, such that we introduce the notation:

$$\tilde{\theta}_\epsilon \doteq \textbf{FedAvg-Epoch}(n, \epsilon, \eta, \theta_t, (\tilde{X}_e, \tilde{Y}_e)_{e \in [\![1,\epsilon]\!]})$$

And the gradient descent of the GRAs is done on all of the dummy sets $(\tilde{X}_e, \tilde{Y}_e)_{e \in [\![1,\epsilon]\!]}$. This dummy sets are introduced to add more flexibility in the attack.

In addition, each dummy set should represent the same data, because the attacked client uses the same dataset between each epoch. Thus [1], introduce a regularization term to use this information. This regularization enforces the different dummy sets to share a common summary statistic $\mathbf{S}(\tilde{X}_e, \tilde{X}_e)$ (for instance the mean image of $\tilde{X}_e$). Their regularization is simply the pair-wise distance of this common summary statistic:

$$\mathcal{L}_{inv} = \frac{1}{\epsilon^2} \sum_{i=1}^{\epsilon} \sum_{j=1}^{\epsilon} \textbf{dist}(\mathbf{S}(\tilde{X}_i, \tilde{X}_i), \mathbf{S}(\tilde{X}_j, \tilde{X}_j))$$

At the end of their attack, [1] propose the following reconstructed data points:

$$\tilde{X}, \tilde{Y} = \textbf{Aggregate}((\tilde{X}_e, \tilde{Y}_e)_{e \in [\![1,\epsilon]\!]})$$

Where **Aggregate** combines the images that are similar between the different dummy sets. In the end, [1]'s attack is:
— **Sim**: the **Fedavg-Epoch** simulation
— **dist**: the **cossim** (cosine similarity, as in [3])
— **reg**: the combination of the TV and their $\mathcal{L}_{inv}$

Later, [8] improved the aggregation step by removing outliers in the sets $(\tilde{X}_e)_{e \in [1,..,\epsilon]}$.

3 Methods

In this paper, we introduce a novel regularization term designed to enhance any GRA that mimics a FedAvg-like simulating scheme. Our key insight is that the intermediate client parameters at the end of each epoch $(\theta_e^U)_{e \in [\![1,\epsilon]\!]}$, should lie close to a straight line. This is based on the assumption that the loss curvature should not vary significantly between two nearby points, ensuring that the gradient directions align towards the same descent path. Consequently, the parameters should descend along the gradient descent direction, which ideally forms a straight line.

To illustrate this concept, consider Fig.1, where the intermediate step θ_1^3 of the client (in black) is relatively close to the line passing through θ_1^1 and θ_2^3. Since these last two points are known to the attacker, the straight black dashed line is also known. Therefore, we propose the following regularization term:

$$\textbf{Line} = \lambda \sum_{i=1}^{\epsilon-1} \textbf{cossim}((\tilde{\theta}_{i+1}^U - \tilde{\theta}_i^U), \theta_\epsilon^U - \theta_t) \tag{2}$$

Here, λ is a scaling parameter to weight the impact of this regularization term. In this context, the vectors are $(\tilde{\theta}_{i+1}^U - \tilde{\theta}_i^U)$, corresponding to the vectors passing by the first and last points of each epoch in the parameter space. Minimizing

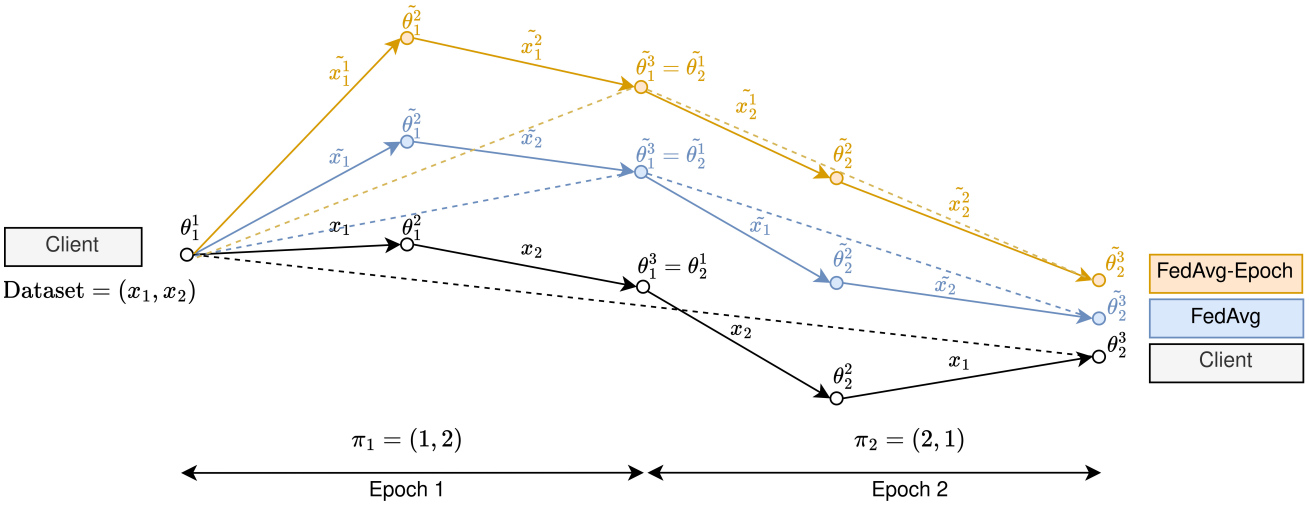

Figure 1 – Overview of the client's FedAvg optimization with ($\epsilon = 2, n = |(x_1, x_2)| = 2, \beta = 1, U = \frac{n}{\beta} = 2$) and the attacks simulating **FedAvg** and **FedAvg-Epoch**. The model's parameters at epoch $e \in [1, \cdots, \epsilon]$ and step $u \in [1, \cdots, U]$ is denoted θ_e^u. θ_1^1 is the received model parameters from the server. $\theta_\epsilon^U = \theta_3^2$ are the parameters send back to the server by the client. The dataset is shuffled between each epoch, represented by the permutation π.

this regularization term helps ensure that the intermediate steps $\tilde{\theta}_{i+1}^U$ remain close to the straight line passing through θ_t and θ_ϵ^U. In Fig.1, this is depicted by the blue and yellow dashed lines (for the **FedAvg-Epoch** and **FedAvg** simulations, respectively), which should be parallel to the black dashed line.

4 Experiments and Results

Table 1 – Effectiveness of the attacks, with the PSNR metric, for different values of regularization scale λ (from 10^{-3} to 10). The model is CNN2 and the dataset is CIFAR100. The client's FedAvg HPs are a dataset size of 32, a minibatch size of 4, 5 epochs and a learning rate of $4 \cdot 10^{-3}$.

λ	0.001	0.01	0.1	1.	10
FedAvg	24.18	24.22	24.32	**25.01**	24.72
FedAvg-Epoch	21.82	21.78	21.86	22.93	**23.51**

Considered attacks We wanted to have a fair comparison with state-of-the-art method (i.e. [1]), so that we used their attack. We also used a **FedAvg** simulation and the same loss as [1]. Our attacks simply add our **Line** regularization, to both previous attacks, in the **Loss** term with the scaling parameter λ. This is what we call our attack. Notice that in Tab.1, there is no $\mathcal{L}_{inv}$ regularization compared to attacks in Tab.2.

Reproducibility In order to have robust results, all the metrics reported are the average of at least 25 independent and identical distributed experiments.

Datasets We use CIFAR100 [6], MNIST [7] to have different levels of complexity in the images we attack.

Models The models are a Simple CNN (as in [9]) which we call CNN1, a Deeper CNN (as in [1]) which we call CNN2, and a simple MLP called Linear.

On the Hyperparameter (HP) λ On Tab.1, the Peak-Signal-Noise-Ratio (PSNR) values of the images reconstructed with **FedAvg** and **FedAvg-Epoch** simulations are shown. We can see that from 10^{-3} to 10^{-1} the regularization has little to no impact on the performances of the attacks. Which is expected because the distance term in eq.1 and our regularization **Line** both compute the **cossim**, thus have the same order of magnitude. In addition, we can see an improvement of 0.69 to 1.07 dB on the PSNR comparing $\lambda = 0.1$ and $\lambda = 1.0$, which proves the effectiveness of our regularization term. PSNR scale is logarithmic, so gaining 1. dB is a notable improvement. For $\lambda = 10$, the enhancement is not unanimous, thus we consider $\lambda = 1.$ for the next of the experiments. It is interesting to note that it helps even more the **FedAvg-Epoch** simulation, which has indeed more variables (the sets $(X_e, Y_e)_{e \in [\![1,\epsilon]\!]}$) to optimize, thus the regularization term **Line** is lowering the freedom of these variables and helping the outcome of the attack.

Table 2 – PSNR of the reconstructed images comparing [1]'s attack (**FedAvg-Epoch** + $\mathcal{L}_{inv}$) versus ours (**FedAvg-Epoch** + $\mathcal{L}_{inv}$ + **Line**) for different number of local epoch (ϵ), mini batch size (β), and total number of local updates $U \cdot \epsilon$. The dataset size is fixed to 32. The model is CNN2 and the dataset is CIFAR100. The GAP column is the difference between our attack and [1]'s attack.

$U \cdot \epsilon$	β	ϵ	Attack	PSNR	GAP
20	8	5	[1]	21.85	1.32
			Us	23.17	
40	4	5	[1]	22.37	0.63
			Us	23.00	
80	2	5	[1]	20.97	0.85
			Us	21.82	
	4	10	[1]	20.17	1.39
			Us	21.56	

On the Dataset Size (DS), Datasets and Models On Fig.2, the PSNR of the different attacks are shown under an increasing DS for a variety of models and datasets. As expected, the PSNR is decreasing (for any attack) while the DS is increasing, because the quantity of information to reconstruct (the images) is increasing while the quantity of information available (the gradients) stay the same. Our attack (shown with solid lines)

is almost always better compared to its counterpart (dashed lines), up to a big margin (3dB). Our regularization fails only for the **FedAvg** attack carried out on the CNN1 with MNIST. We have no explanation for this phenomena. In addition, the regularization is less effective when the DS is higher. This is due to the fact that our assumption ("the gradient path is close to a straight line") is more realistic when the number of data points is low. Because it increases the number of optimization steps, leading to less straight gradient paths.

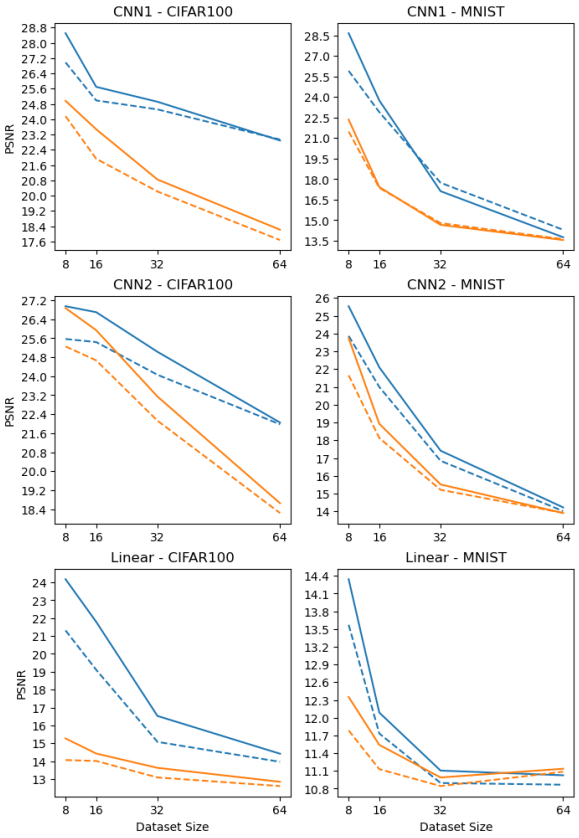

Figure 2 – PSNR of the reconstructed images coming from the attacks in function of the client's DS for different models and datasets. The blue, orange lines are **FedAvg**, **FedAvg-Epoch** simulation respectively. The solid, dashed lines are with or without our regularization respectively. Thus the orange dashed line is [1]'s attack. The client's FedAvg HP are a mini-batch size of 4, 5 epochs and a learning rate of $4 \cdot 10^{-3}$.

On the number of local epochs (ϵ) and the mini batch size (β) On Tab.2, the comparison between [1]'s attack and ours (i.e. the same with the regularization **Line**) for different ϵ and β is provided. Our attack is always better than [1] which proves the effectiveness of our regularization. In addition, at a fixed number of epochs (and dataset size), our regularization is more effective when the batch size is higher (1.32dB). Looking at the four last rows of the Tab.2, it seems that our attack is more effective when the number of local epoch is higher (improvement of 1.39dB vs. 0.85dB). Indeed, our regularization might restrict the optimization's freedom of the dummy sets $(X_e, Y_e)_{e \in [\![1,\epsilon]\!]}$ such that the intermediate parameters $\tilde{\theta}_e^U$ are close to the line $\theta_\epsilon^U - \theta_1^1$

5 Conclusion

FedAvg is harder to attack than FedSGD but is a more realistic scenario. The work of [1] proposes an effective attack against FedAvg. In this paper, we add a regularization to their attack that helps the convergence of the attack. The idea of the regularization is to help the simulation of FedAvg by enforcing the intermediate steps to stay close to a line. This line is given by the direction of the client's gradients. Throughout a variety of experiments, we prove the effectiveness of our regularization for different scenarios. Our regularization is applicable to any attack that is designed for FedAvg.

Acknowledgement

This work has benefited from French State aid managed by the Agence Nationale de la Recherche (ANR) under France 2030 program with the reference ANR-23-PEIA-005 (REDEEM project).

References

[1] Dimitar Iliev Dimitrov, Mislav Balunovic, Nikola Konstantinov, and Martin Vechev. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022.

[2] Peter Kairouz et al. Advances and Open Problems in Federated Learning, March 2021. arXiv:1912.04977 [cs, stat].

[3] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting Gradients – How easy is it to break privacy in federated learning?, September 2020. arXiv:2003.14053 [cs].

[4] Jiahui Geng, Yongli Mou, Feifei Li, Qing Li, Oya Beyan, Stefan Decker, and Chunming Rong. Towards General Deep Leakage in Federated Learning, January 2022. arXiv:2110.09074 [cs].

[5] Pierre Jobic, Aurélien Mayoue, Sara Tucci-Piergiovanni, and François Terrier. Extending the scope of gradient reconstruction attacks in federated averaging. In *Proceedings of the 2024 ACM Workshop on Information Hiding and Multimedia Security*, IHMMSec '24, page 235–246, New York, NY, USA, 2024. Association for Computing Machinery.

[6] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009.

[7] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[8] Bowen Li, Hanlin Gu, Ruoxin Chen, Jie Li, Chentao Wu, Na Ruan, Xueming Si, and Lixin Fan. Temporal Gradient Inversion Attacks with Robust Optimization, June 2023. arXiv:2306.07883 [cs].

[9] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data, February 2017.

[10] Jin Xu, Chi Hong, Jiyue Huang, Lydia Y. Chen, and Jérémie Decouchant. Agic: Approximate gradient inversion attack on federated learning, 2022.

[11] Hongxu Yin Hongxu, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through Gradients: Image Batch Recovery via GradInversion, April 2021. arXiv:2104.07586 [cs].

[12] Ligeng Zhu, Zhijian Liu, and Song Han. Deep Leakage from Gradients, December 2019. arXiv:1906.08935 [cs, stat].