Towards Understanding the Risks of Gradient Inversion in Federated Learning

Ali Hatamizadeh
NVIDIA
Hongxu Yin
NVIDIA
Pavlo Molchanov
NVIDIA
Andriy Myronenko
Nvidia (United States)
Wenqi Li
NVIDIA https://orcid.org/0000-0003-1081-2830
Prerna Dogra
NVIDIA
Andrew Feng
NVIDIA
Mona Flores
NVIDIA https://orcid.org/0000-0002-7362-3044
Jan Kautz
NVIDIA
Daguang Xu
NVIDIA, Santa Clara, CA,
Holger Roth (hroth@nvidia.com)
NVIDIA https://orcid.org/0000-0002-3662-8743

Article

Keywords:

Posted Date: December 14th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1147182/v1

Towards Understanding the Risks of Gradient Inversion in Federated Learning

Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Prerna Dogra, Andrew Feng, Mona G. Flores, Jan Kautz, Daguang Xu, Holger R. Roth
NVIDIA

Abstract
Federated learning (FL) allows the collaborative training of AI models without needing to share raw data. This capability makes it especially interesting for healthcare applications where patient and data privacy is of utmost concern. However, recent works on the inversion of deep neural networks from model gradients raised concerns about the security of FL in preventing the leakage of training data. In this work, we show that these attacks presented in the literature are impractical in real FL use-cases and provide a new baseline attack that works for more realistic scenarios where the clients' training involves updating the Batch Normalization (BN) statistics. Furthermore, we present new ways to measure and visualize potential data leakage in FL. Our work is a step towards establishing reproducible methods of measuring data leakage in FL and could help determine the optimal tradeoffs between privacy-preserving techniques, such as differential privacy, and model accuracy based on quantifiable metrics.

INTRODUCTION

Federated learning (FL) allows the collaborative training of AI models without the need to share raw data[1]. Participating client sites only share model weights or their updates with a centralized server and therefore, the underlying data privacy is being preserved[2]. This makes FL especially attractive for healthcare applications where patient privacy is of utmost concern. Several real-world applications of FL in medical imaging have shown its potential to improve the model performance and generalizability of AI models by allowing them to learn from large and diverse patient populations, which could include variations in scanner equipment and acquisition protocols, often spanning different countries, continents, and patient populations[3–7]. FL has attracted a lot of attention from researchers[8], who study how to deal effectively with heterogeneous (or non-i.i.d.) data[9,10], address fairness and bias concerns[11], improve the efficiency of client-server communication[12], and more[13].

But how true is the assumption that FL preserves privacy? Several efforts have demonstrated the possibility of recovering the underlying training data (e.g., high-resolution images and corresponding labels) by exploiting shared model gradients (i.e., the model updates sent to the server during FL)[14–16]. This attack scenario, known as the "gradient inversion" attack, could happen on both the server- and client-side during FL training. In typical FL scenarios for healthcare, the server-side gradient inversion attack assumes that all the clients trust each other[1] while the server is "honest but curious" (Bonawitz et al. 2017). This means the server follows the FL procedure faithfully but might be compromised and attempts to reconstruct some raw data from the clients (see Figure 1, Methods).

In the "cross-silo" scenario of FL[8], where an AI model is trained on a relatively small number of client sites with large amounts of data, one can assume that each client is running a relatively large number of training iterations on their local data before sending the model updates to the server. This is in contrast to the "cross-device" setting with many hundreds or thousands of clients, each with relatively small amounts of data, local compute resources, and unreliable network connections which might be dropping in and out at any time during training. In healthcare, there are different ways the data could be organized across institutes. In the so-called "horizontal" FL,

each institute has data from different patients who share common attributes for a certain modeling task. On the other hand, “vertical” FL allows institutes which share a common set of patients but with different attributes to collaborate. For instance, bloodwork results and imaging data of the same patients could be possessed by two different participating institutes[17] but used to build a common AI model.

In this work, we consider the common FL setting in which the participating clients share model updates ΔW, after training on their local data, to a central server that aggregates them, updates the global model, and sends back the updated global model to each client to continue the local training (see Figure 1). This procedure is continued until the model converges. It is easy to see that this FL setting is equivalent to the well-known federated averaging (FedAvg) algorithm[2]. Although the raw training data is never shared in FL, there is a concern that the shared model updates between the server and clients may still leak some training data[14]. While the server receives model updates from each client, a client computes model updates from the global models, consisting of updates from all clients, in different rounds of FL. In this work, we focus on the server-side gradient inversion attacks and investigate their applicability to FL in medical image classification scenarios. However, we note that the client-side gradient inversion could be treated in the same way but would involve the aggregation of more gradients from other clients.

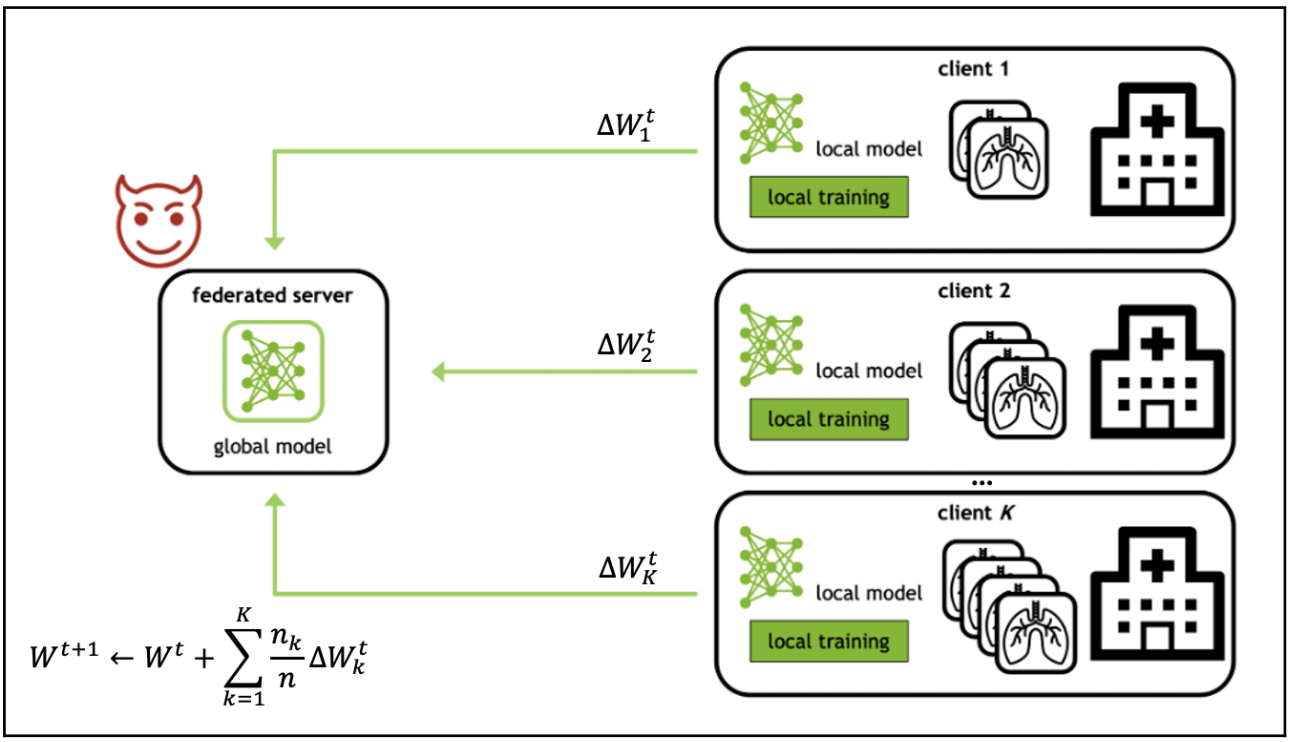

Figure 1 ***The server-side gradient inversion attack scenario in FL.*** *An attack scenario in FL. In a typical FL setup for healthcare applications,* ***K*** *participating sites (often hospitals or research institutions) keep their data locally while training a common global model. This model is distributed by a centralized federated server to initialize the current round of local training at each site. After the local training is completed, all sites or a subset of the sites send model updates, ΔW with respect to the global model, to the server, which aggregates them by performing a weighted average and then uses the aggregate to update the global model for the next round of FL $(t + 1)$. Here, n_k and n stand for the number of images at client k and the total number of images across clients, respectively. This iterative procedure continues until convergence of the models. To study the gradient inversion attack, we assume an “honest but curious” (Bonawitz et al. 2017) scenario where the server performs the FL computations faithfully but could be compromised by a potential malicious actor who could gain access to the training configurations and model updates from the participating sites and attempt an inversion attack (see Methods).*

We investigate the risks of data leakage in gradient inversion attacks in practical FL scenarios using a real-world horizontal FL system to record the model updates. Unlike previous efforts in this area[14,18] which assume fixed Batch Normalization[19] (BN) statistics (i.e., evaluation mode), our work is the first to utilize model updates (i.e., gradients) that are generated by updating BN statistics during training, commonly known as “training mode” in frameworks like PyTorch or TensorFlow. Therefore, our work is directly applicable to real FL use-cases where BN is often used. Specifically, we propose a novel gradient inversion algorithm for estimating the running statistics of BN layers (i.e., running mean and variance) to match the gradient updates, and as a result, extract prior knowledge from intermediate feature distributions.

To further understand the risk of gradient inversion in FL, we propose several ways to measure and visualize the potential of data leakage. In addition, we introduce a comprehensive study of realistic FL scenarios consisting of clients training with different amounts of data and batch sizes, different differential privacy settings and utilize them to evaluate our proposed gradient inversion attack. We use our findings to make recommendations for more secure and privacy-preserving FL settings.

Our work is inspired by recent advances in gradient inversion of deep neural networks, which also raised concerns about privacy in FL[14], and especially for healthcare applications[18]. One of the earliest works on this topic was Deep Gradient Leakage (DLG)[20], which only relied on the information of shared gradients for their inversion attack. The main idea of their proposed algorithm is to match the gradients between the synthesized and real data (see Methods). However, this approach has several drawbacks which limit its use in practical FL scenarios. First, it only works for low-resolution images (32x32 or 64x64). Second, it needs a second order optimizer (e.g., L-BFGS) for the reconstruction algorithm. Such an optimizer is computationally expensive and cannot be easily applied to scenarios with high-resolution images or larger batch sizes. The work by Geiping et al.[14] attempts to address these issues and align itself with a more practical FL scenario by proposing a cosine similarity loss function to match the synthesized and ground truth gradients. This work uses first-order based optimizers such as Adam to make it more suitable for high resolution images and larger batch sizes. Geiping et al.[14] showed also that high-resolution images could be reconstructed from model updates sent in FL from small batch sizes, with access to the model gradients for one training iteration. Most recently, Yin et al.[21] further improved the image fidelity and visual realism of gradient inverted reconstructions by matching fixed running statistics of BN layers. To effectively localize objects in the training images, they use different optimization seeds to simultaneously reconstruct several images that can be registered to a consensus image from all recovered images. Kaissis et al.[18] applied the framework of Geiping et al. to medical image classification scenarios and analyzed its applicability to reconstructing training images from FL clients. However, in their analysis, clients were trained only in the evaluation mode without updating the BN statistics, which is uncommon in modern machine learning.

Despite remarkable insights, one common assumption by prior work is the fixed BN statistics given the most common task of single-batch gradient inversion. However, in practical FL scenarios, such statistics vary inevitably when multiple batches are jointly used by a client within one training epoch for the weight update calculation. As the gradient inversion problem is a second-order optimization by nature that relies on a static underlying forward pass, any shifts in BN statistics can result in accumulated errors across batches and baffle the efficacy of prior attacks, as we show later. By carefully tracing the momentum mechanism behind BN updates, this work demonstrates the first successful attempt of gradient inversion in a multi-batch scenario with updated BN statistics as in real-world FL applications. In addition, a comprehensive analysis is provided as guidance to enhance FL security against gradient inversion attacks. To the best of our knowledge, our work is the first to show how a gradient inversion attack might be successful when clients are sending model updates while updating their BN statistics, as commonly done in real-world FL scenarios.

RESULTS

FL scenario. We simulate eight clients using different numbers of training images and local batch sizes in a cross-silo FL scenario with horizontal data partitioning for a chest X-ray image classification task. Each client trains an ImageNet-pretrained[23] ResNet-18[22] for one epoch using a stochastic gradient descent (SGD) optimizer without momentum and a step-wise learning rate decay and sends the model updates to the server. The dataset consists of publicly available chest X-ray images from normal patients and patients who tested positive for COVID-19[24,25]. Table 1 shows a summary of the local training batch size and number of training images that were used for each client. Each client has its own validation set of 200 cases that will be used to select the best global model based on the average validation accuracy. The selected global model is finally evaluated on the testing set. Except for client 9, each client training set contains the same number of normal and COVID-positive cases in order to simulate clients with balanced class distributions. Client 9 shares its only training image (a normal case) and its validation set with client 1. As a result, client 9 is at the highest risk of data leakage as it is sending model updates based on a single image and a batch size of one and will be referred to as the "high-risk client". We intentionally utilize relatively small batch sizes in order to study the effect of our new inversion attack on clients with varying amounts of training data (and therefore, local training iterations). Therefore, all clients investigated in this study could be considered of higher risk than in real-world FL scenarios where clients use larger amounts of training data and batch sizes (see Discussion).

Table 1 *Dataset used for simulating federated learning.*

client	batch size	# of training data	# of validation data	# of testing data
client 1	4	8	200	1382
client 2	4	32	200	-
client 3	4	128	200	-
client 4	4	512	200	-
client 5	8	8	200	-
client 6	8	32	200	-
client 7	8	128	200	-
client 8	8	512	200	-
client 9 ("high-risk")	1	1	200	-
	Sum	1360	1600	1382

Limitations of previous attack scenarios. We utilize an extended version of the GradInversion algorithm[21] to invert the training images of each client separately based on its model updates sent during FL. For details on the inversion attack, see Methods. Prior works[14,18] did not consider clients updating the statistics of BN layers which are commonly used in modern classification models with deep convolutional neural networks (CNNs)[19,22]. Therefore, the clients in these efforts only updated model weights while executing the training with fixed BN statistics. Here, we show that when the attacked clients update BN layers during training, an additional BN loss is essential for the attack to succeed. The running mean and variance of the BN layers need to be updated during the optimization of the inversion attack as these statistics inevitably change during training. Since a gradient inversion attack is a second-order optimization problem, the accumulated errors in

estimating the BN statistics by assuming fixed values hinder successful reconstructions of training data (see Figure 2 (b)). In addition, it is critical to initialize the network of inversion attack with the same global model weights that are used to initialize the client's network in a particular FL round to ensure the exact simulation of local training steps within the attack (see Figure 2 (c)).

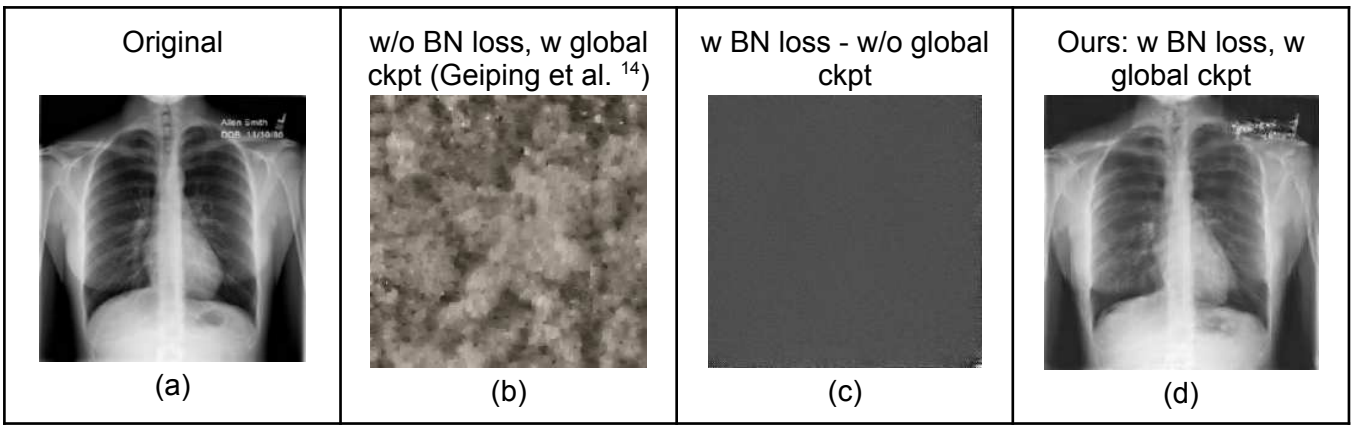

Figure 2 ***Success factors of the inversion attack.*** *Here, the "high-risk" client 9 sends updates based on one iteration and one image in round 10 of an FL experiment. The original training image is shown for comparison (a). The inversion technique by Geiping et al.[14] does not estimate updated BN statistics during training and therefore cannot handle clients with BN layers in their architectures (b). The attack also fails if not initialized with the current global model checkpoint (ckpt) (c). Both the current global checkpoint and proper accounting for updated BN statistics during training are necessary for the attack to succeed (d). The patient's name and date of birth in the original image are randomly generated to illustrate the difficulty of inverting detailed textual information by the gradient inversion attack.*

Measuring data leakage by inverting gradients. Assuming the attacker has access to the full information needed for a successful data recovery as shown in Figure 2 (d), we attempt to invert any model updates coming from any of the clients in different rounds of FL. Figure 3 illustrates the effect of the number of training images, batch size, and local iterations on the success of a gradient inversion attack in an earlier or later round of FL. A larger number of local training iterations in clients with larger training sets detrimentally impacts the quality of reconstructions due to accumulated errors in estimating the updated BN statistics. In addition, reconstructions from later rounds of FL (e.g., round 90 of 100) have improved image quality in terms of image fidelity and capturing the details of anatomical structures. This observation might be attributed to the global model having learned more effective feature representations during the later stages of the FL training and outliers or individual images in those training sets that cause higher losses in later stages of training and therefore contribute to more recognizable reconstructions.

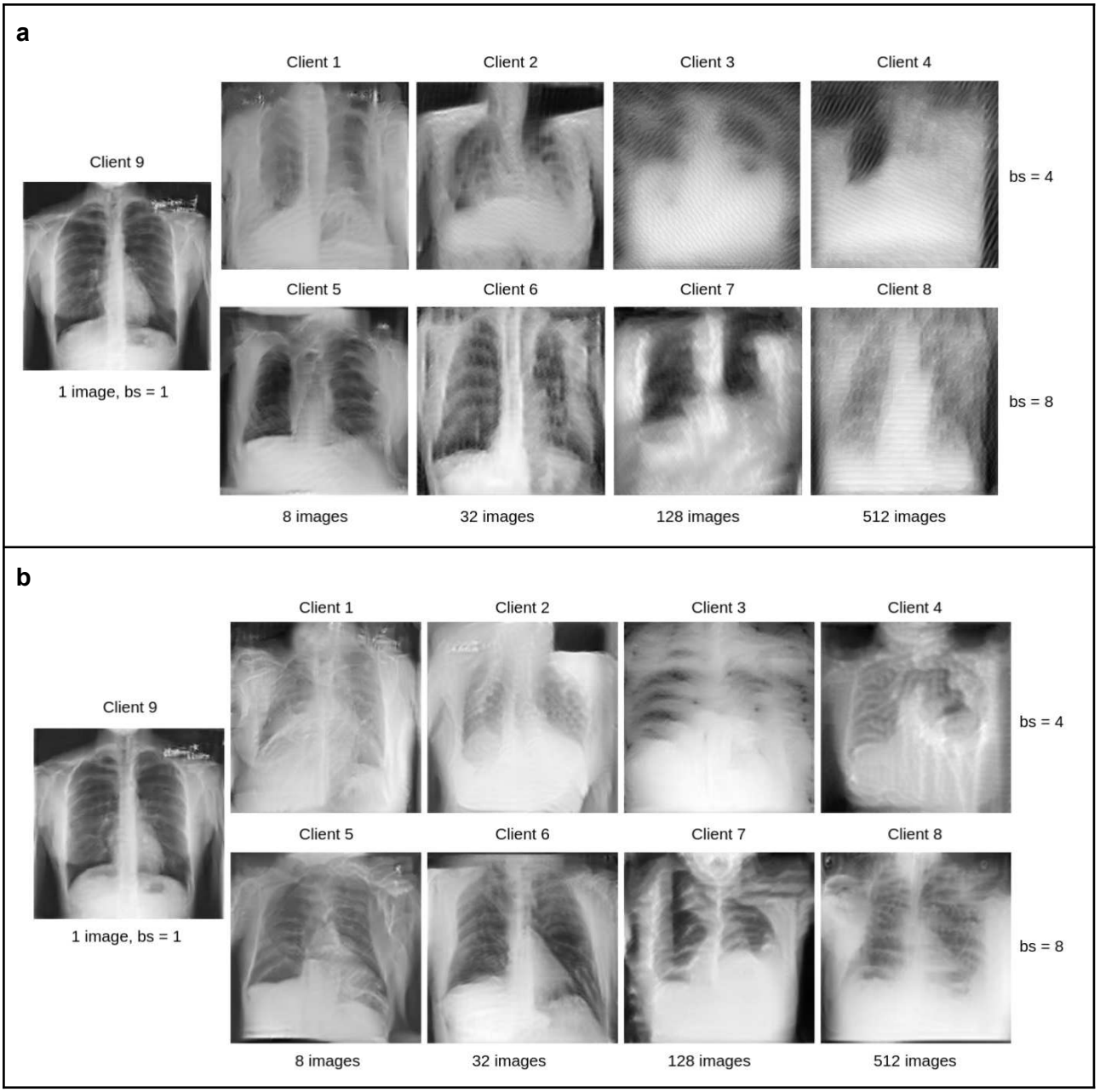

Figure 3 ***Reconstructions from different FL rounds.*** *Effect of the number of training images, batch size, and local iterations on the inversion attack early on in FL training (a, round 10) and late in training (b, round 90).*

The effect of training batch sizes and local number of iterations is further quantified in Figure 5 (a) and (b), respectively. Here, we utilize the global model from round 90 for the attack and train on a client with increasing dataset sizes. In the first setting, we set the number of local iterations to one but keep increasing the batch sizes. In the second setting, we set the number of batch sizes to one but increase the local number of iterations. For each setting, we add new images to the client's training set while measuring the impact of batch size and local number of iterations by using the same original image for reference. As shown in Figure 5 (a) and (b), increasing both the number of batch size and local number of iterations are detrimental to the success of the attack. In particular, increasing the local number of iterations negatively impacts the reconstructions as the estimation of BN statistics becomes less accurate when the client is iterating over different local training images (see METHODS). To quantify the data leakage between the original image and the reconstructions, we use Structural SIMilarity index (SSIM), a pixel-wise metric for measuring image similarity in a more intuitive and interpretable manner compared to other commonly used metrics such as root-mean-squared error or peak signal-to-noise ratio[26].

Mitigating inversion through differential privacy (DP). A straightforward approach to avoid data recovery by a server-side gradient inversion attack is the addition of random noise to the model updates before they are sent to the server[27]. Here, we explore a simple DP protocol that adds calibrated Gaussian noise, with zero mean, based on the gradient magnitudes of the model updates (see METHODS). We evaluate this technique on client 9, which is the high-risk client, using model updates that are sent at round 90. As shown in Figure 4 (c), the image reconstruction quality, as measured by the SSIM between original and recovered images, expectedly degrades as more noise is added to model updates from the client.

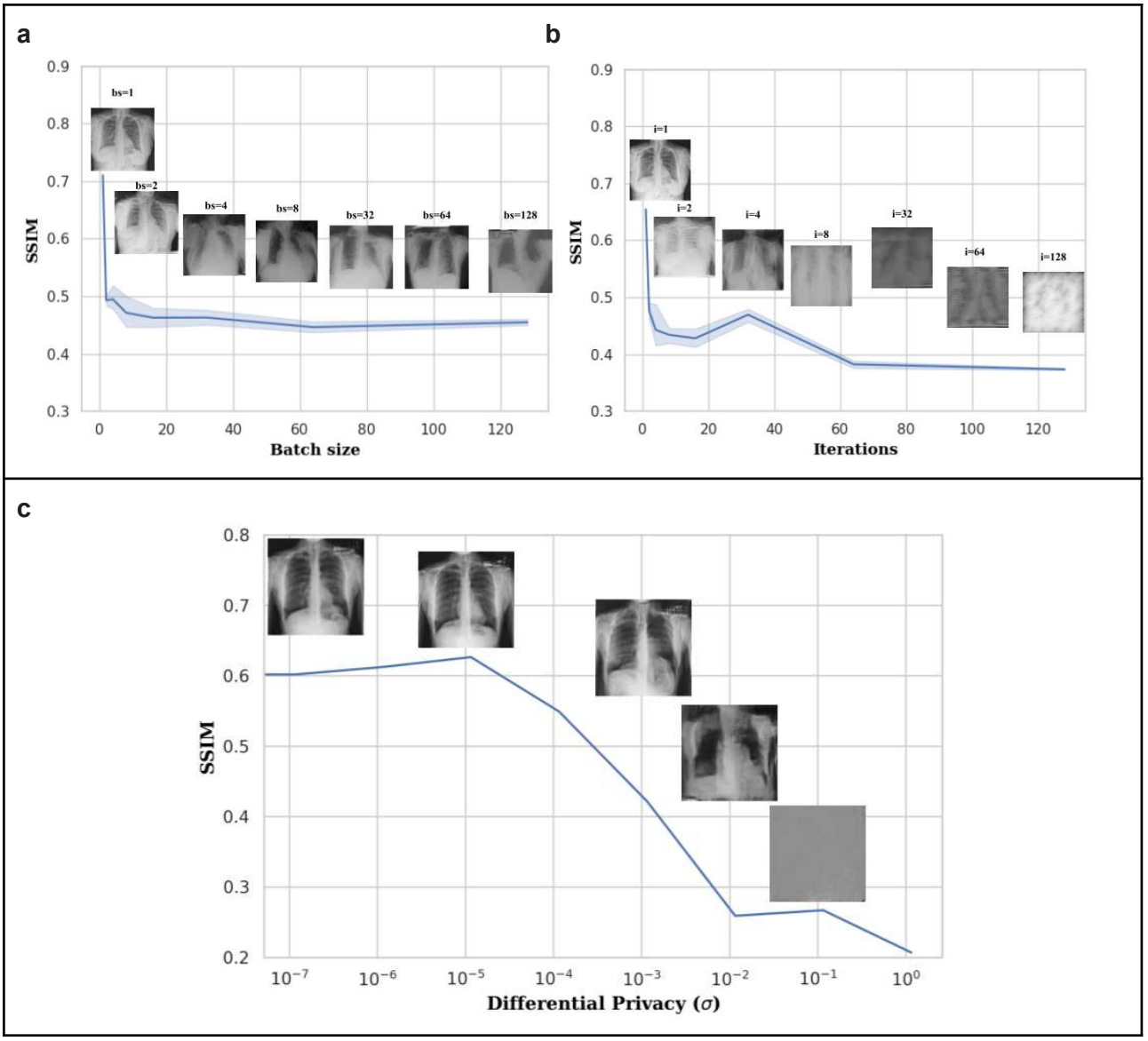

Figure 5 ***Quantifying data leakage.*** *The impact of batch size (a) and local number of iterations (b) to the success of the inversion attack. 95% confidence interval bands are shown for each client. (c) The "high-risk" client 9 is shown with varying amounts of noise added to the model updates before they are sent to the server. The SSIM value of the reconstructed inversions are shown for reference.*

The SSIM value between the prior which is used to initialize the gradient inversion attack (see METHODS) and the original image is 0.37 (see Figure 6). This can be considered as a lower bound of this metric during the attack. In addition, as the noise level increases, reconstructions from model updates with added DP become closer to the prior image and less similar to the original training image. This indicates that a privacy setting (σ) allowing only a reconstruction with a SSIM value equal to or less than that between the original image and the prior could be considered "safe". For instance, as shown in Figure 5, this behavior can be observed for the high-risk client 9 at FL round 90 and noise added with $\sigma = 10^{-2}$.

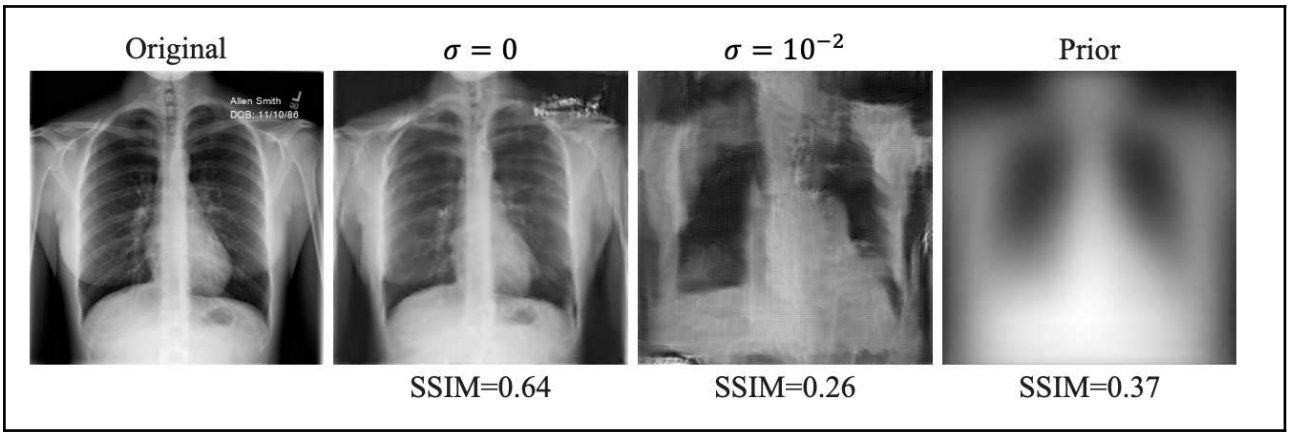

Figure 6 ***Structural similarity (SSIM).*** *SSIM can be used to measure both the similarity between an original image and reconstruction or the similarity between the prior image used in the attack and the original image. An SSIM value below the one compared with the prior could indicate that no useful information was leaked or recoverable by the attack. Note: the patient's name and date of birth in the original image are randomly generated.*

Quantification of data leakage. We further visualize the degree of data leakage across different clients in FL with respect to different levels of privacy preservation through added Gaussian noise. Comparing data leakage across clients with different images is not straightforward. This is due to the variation in the reference images used to compute the SSIM, which renders the absolute values of the similarity metrics between the reconstruction and the training images incomparable. Therefore, we utilize a relative percentage change with respect to a reference point to compute a more comparable quantitative metric across different clients. We define a new metric denoted as "Relative Data Leakage Value" (RDLV) to measure the relative percentage change of the similarity metric (SSIM) for reconstructions from different clients' model updates. Given a training image T_i, the prior used in the gradient inversion attack P and the corresponding reconstruction I_i, RDLV is defined as:

$$RDLV = 100 \times \frac{SSIM(T_i, I_i) - SSIM(T_i, P)}{SSIM(T_i, P)} \quad (1)$$

Figure 6 illustrates the RDLV based on different differential privacy settings for each client. Each line shows the average central tendency and a 95% confidence interval (bands shown in the plot) estimated using bootstrapping (with 1000 trials) on reconstructions that achieve the highest SSIM with any of the original training images during different rounds of FL. As observed, only the high-risk client 9, which sends model updates from training one iteration on one image, leaks the training data up to a certain amount of noise which corresponds to a positive value of RDLV (up to around $\sigma_0 \geq 25$ for most FL rounds in this case). A positive percentage change indicates that the reconstructions are more similar to the original images than the prior used in the gradient inversion attack. In addition, clients with more than one training image do not leak data in this regime within the confidence interval as their reconstructions are less similar than the prior used in the attack. As

expected, the global model accuracy drops as more noise is added to the model updates from clients during FL.

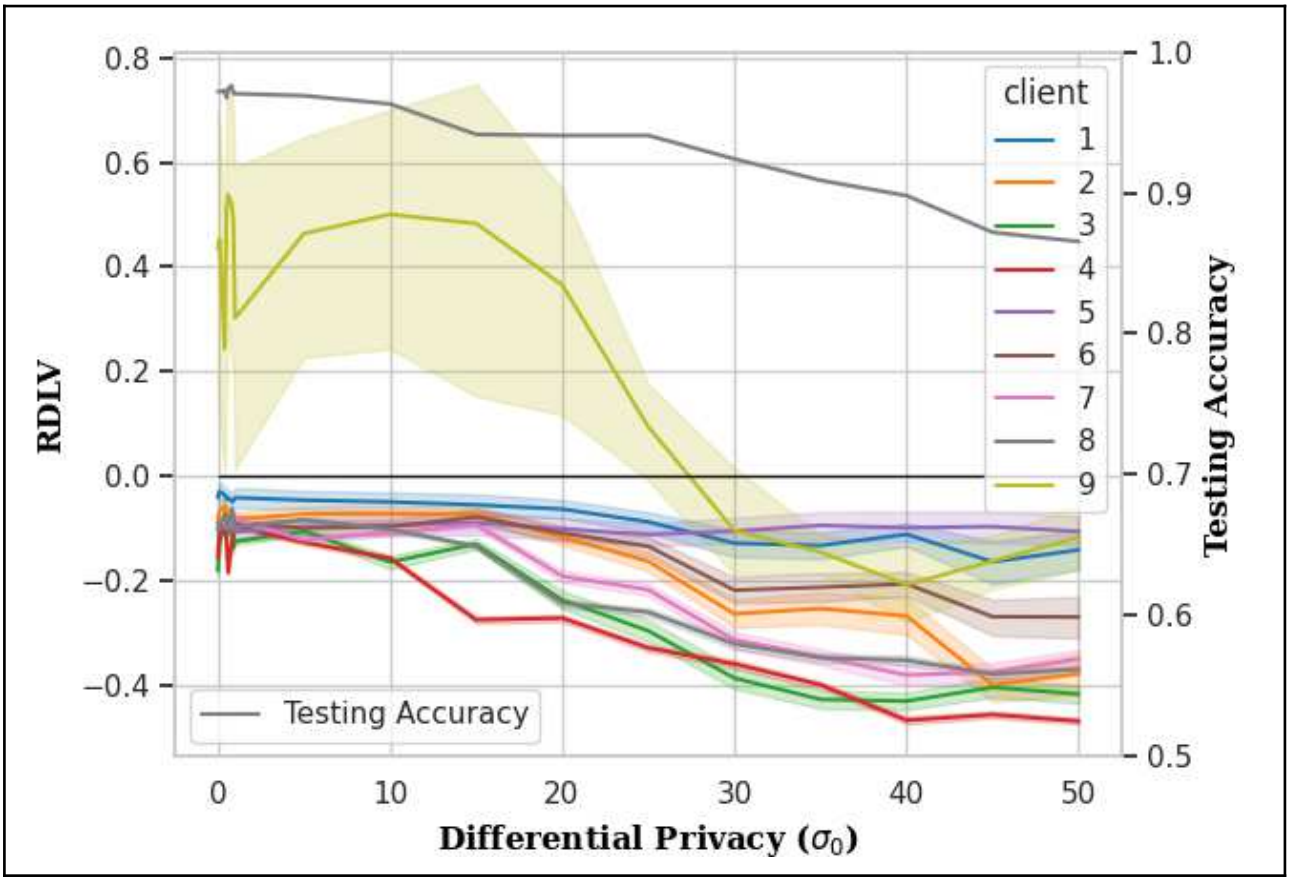

Figure 7 ***Quantification of data leakage.*** *"Relative Data Leakage Value" (RDLV) for each client compared to the prior used in the gradient inversion attacks for different DP settings (σ_0). Higher σ_0 values add more noise to obfuscate the model updates sent by the clients. The testing accuracy of the best global model is shown for comparison. 95% confidence interval bands are shown for each client.*

Deep embedding-based visualizations. Alternative ways to measure the similarity between reconstructed images and training images are based on deep feature embeddings. We use an off-the-shelf ImageNet-pretrained ResNet-18[22] to extract 512-dimensional feature embeddings from training and reconstructed images (see Methods for more details). In Figure 8, we visualize the deep feature embeddings of training images and reconstructions using t-SNE[28]. As shown, with higher values of Gaussian noise for privacy preservation, the clusters of original images and reconstructions further drift apart as the similarity between the original training images and any of the reconstructions decrease significantly.

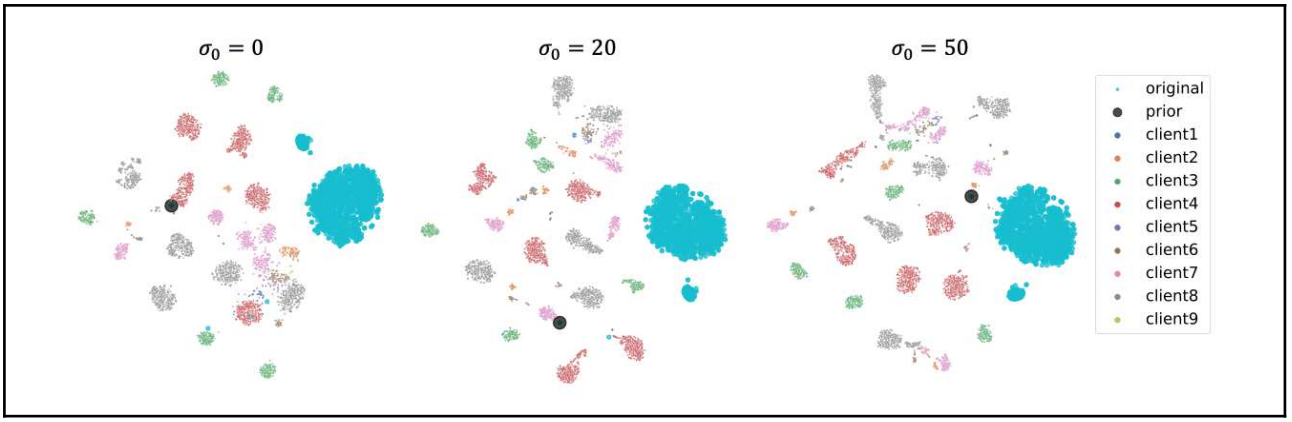

Figure 8 ***Visualizing data leakage.*** *Two-dimensional T-SNE[28] embeddings of deep features extracted from original training images and reconstructions at different DP levels as indicated by* σ_0 .

Data leakage as captured by membership inference attacks. We furthermore adjusted the *Image Identifiability Precision* (IIP) proposed in Yin et al.[21] to the FL scenario. Our specific approach is outlined in Methods. IIP measures the amount of identifiable information leaked in the gradient inversion attack by counting the closest exact matches between the reconstructions from the membership attack and the original training images for each client. As shown in Figure 9, the high-risk client 9 (updates sent based on 1 image and 1 iteration) attains the highest IIP score of 1.0 up to a Gaussian noise level σ_0 value of ~20. This indicates that the attacker is able to determine the exact image that participated in FL training for that specific client. Furthermore, all other clients already achieve significantly lower values of the IIP score (e.g., <0.1) with much lower Gaussian noise levels σ_0. We also show the cosine similarity (dotted lines) between the reconstructions and original images identifying exact members of the training set at each client based on ImageNet-pretrained ResNet-18[22] feature embeddings. As expected, cosine similarity values also decrease with higher Gaussian noise levels.

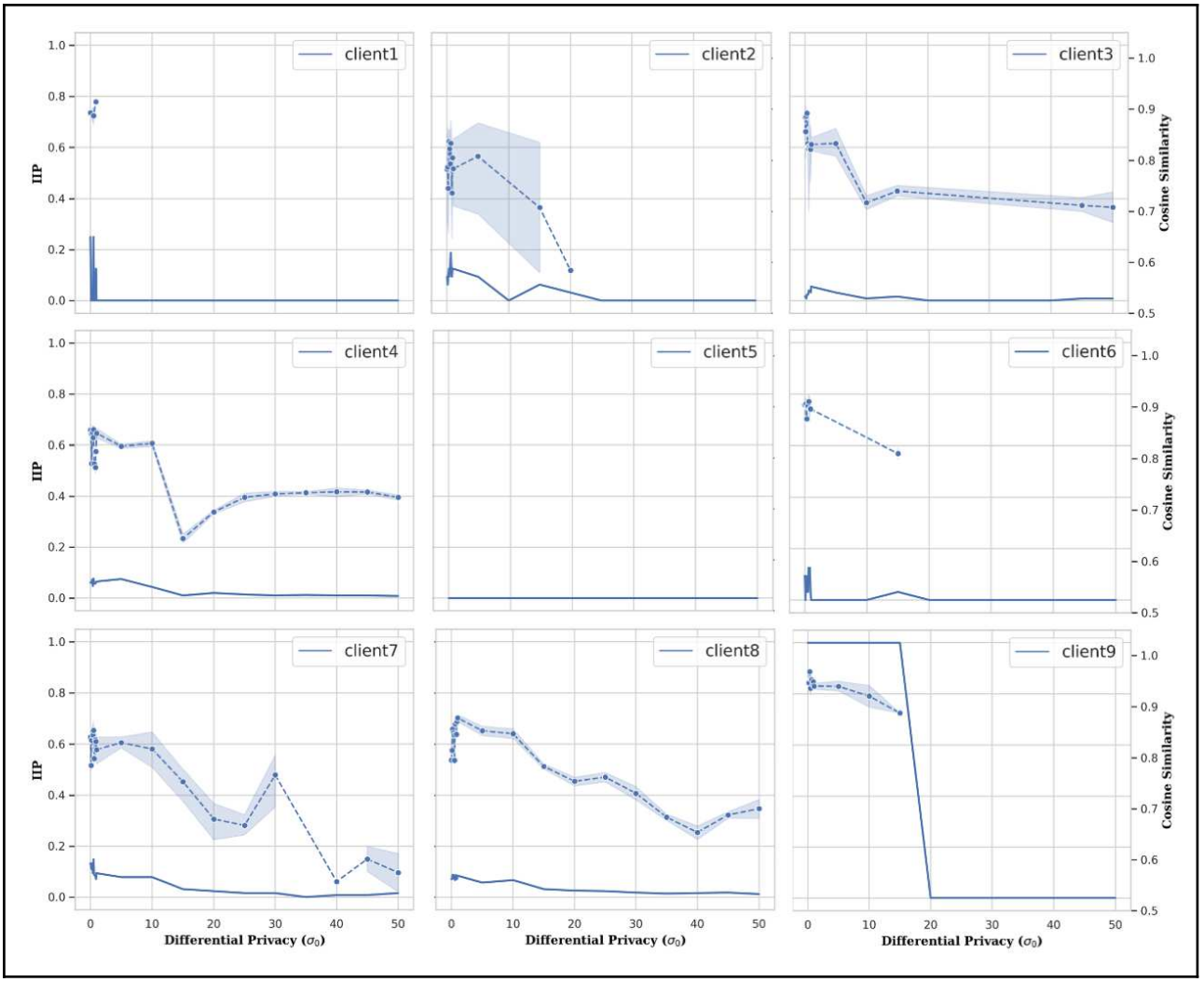

Figure 9 | ***Image Identifiability Precision (IIP).*** *IIP score (solid lines) measures how many reconstructions could be used to identify the exact members of the training images that are used during FL for a specific client using different differential privacy settings (σ_0). We also show the cosine similarity (dotted lines) of feature embeddings between the reconstructions and original images used to identify the exact members for all clients. Only the cosine similarity of matched exact members is shown (IIP > 0). Both IIP and cosine similarities reduce with higher Gaussian noise levels and are typically lower for clients with larger amounts of training images.*

DISCUSSION

Table 2 | **Security conclusions.**

What is insecure?	**What is "more" secure?**
↓ small training sets	↑ large training sets
↓ updates from small number of iterations	↑ updates from a large number of iterations over different images
↓ small batch sizes	↑ large batch sizes

In this work, we have introduced a new way to measure the data leakage in horizontal FL for chest X-ray image classification and proposed several ways to quantify and visualize it. Overall, we conclude that FL security is multifaceted. Clients which send updates based on a small number of training images are at a higher risk of leaking data than the clients sending updates based on larger training sets as well as higher numbers of local training iterations. Furthermore, larger batch sizes make reconstruction via gradient inversion attacks more difficult, a finding that is in line with previous literature[14,16,21]. Table 2 summarizes our overall conclusions based on the findings shown in this study. While sending model updates using a single training image is at a high risk of data leakage, clients that utilize more realistic settings, such as larger number of training images and large number of local iterations as common in real-world FL scenarios, are not at a high risk of leaking identifiable information. Specifically, gradient inversion attacks for clients that iterate over several images before sending model updates are harder due to accumulated errors in the estimation of BN statistics. In this scenario, the attack is executed using only an approximation of the BN statistics (see Figure 3 and METHODS).

In addition, reconstructions from model updates that are intercepted in later stages of FL training exhibit higher image quality in terms of image fidelity and capturing the fine-grained details of anatomical structure. This improved image generation capability could be attributed to the improved feature learning capability of the global model in later rounds of training. However, these images still do not bear any identifiable similarities to the individual training images. As shown in Figure 3, the reconstructed images start to resemble real chest X-ray images, but none of them could be found in the original training set. As a result, such reconstructions do not constitute a significant security breach.

Additionally, there are several potential mitigations against the server-side gradient inversion attacks that could be easily incorporated into modern FL systems:

1) If the attacker does not have access to the current state of the global model with respect to which client's updates are computed, then the attack becomes unsuccessful (see Figure 2). We believe that even only removing the access to the state of the last layer can render the attack ineffective. Specifically, this could be achieved efficiently by utilizing cryptographic techniques such as homomorphic encryption to allow secure aggregation of model updates [29]. This ensures that the server never holds the complete unencrypted model to avoid a potential gradient inversion attack while adding negligible computing requirements.

2) If the attacker does not have access to the model architecture, attempting to match the gradients will become very challenging. The clients could be sending only numerical arrays of model updates to be aggregated on the server without the server ever holding the information about the current model architecture or training procedure. While theoretically, the attacker could infer common architectures from the size of the parameter arrays, in practice, it would become extremely difficult, if not impossible, to deduce the exact architecture, especially for custom architectures often used in healthcare applications.

3) For models that are trained with BN, not sending the BN statistics to the server causes the gradient inversion attack to fail (see Figure 2). Techniques such as FedBN[30,31] could reduce the risk of server-side attacks by keeping personalized BN statistics specific to each client. One negative aspect of this approach is the lack of a common global model after FL training since only an ensemble of personalized client models would be available.

In addition, DP techniques typically reduce the risks of data leakage in many scenarios. In this work, we explored a simple variation of the Gaussian DP mechanism to add calibrated noise to the model updates before sending them to the server. Even this simple approach showed to be effective in reducing the risks of gradient inversion attacks. However, a noticeable degradation in model performance was observed with higher Gaussian noise levels. As DP has shown to be an effective approach in reducing data leakage for gradient inversion attacks, future research efforts are needed to explore the optimal tradeoff between various DP mechanisms and model performance.

Other techniques such as secure authentication, trusted execution environment, and secure communication using for example, SSL encryption, and cryptographic methodologies such as homomorphic encryption[29] or secure multi-party computation[32] to completely encrypt the exchanged updates could be used to ensure a secure FL. As previously discussed, targeted encryption of the most vulnerable parts of the model (e.g., the last linear layer) might be enough to ensure the security of training while only introducing a minimal computational overhead.

We also identified some limitations of our study. For instance, we only studied modern CNN architectures such as ResNet-18 that are commonly used in image classification tasks. We did not consider other network architectures such as those used for image segmentation or natural language processing in our analysis. Furthermore, we limited this study to clients using SGD as the training optimizer. More complicated optimization schemes such as FedOpt [33] might require further considerations. Moreover, using the previously discussed RDLV regime (positive percentage change, see Figure 7), some reconstructed images did not fall within the 95% confidence interval, and as a result, could indicate “data leakage”. However, qualitative comparisons did not reveal any data leakage or individually identifiable information in those reconstructions. Specifically, these reconstructions have a low IIP score and do not bear close similarities to the original training images. However, they have some qualities that cause a relatively high SSIM metric. With such cases, similarity metrics that are computed based on deep feature embeddings might provide more plausible values for comparison.

Future work might identify the vulnerable populations such as outliers or individual images that might cause higher losses in later stages of FL training causing higher SSIM or IIP values. Furthermore, other applications such as 2D/3D segmentation and 3D classification along with novel model architectures should be investigated. Other attack scenarios like model inversion after FL training, as well as backdoor and membership attacks, should be further studied[34].

In conclusion, we presented a new way to measure and visualize data leakage in real-world FL scenarios. Without resorting to contrived settings for enabling the gradient inversion attack[14,16], we studied a practical scenario in which model updates from clients were sent by accounting for updated BN statistics (i.e., training mode). We conclude that previous efforts likely exaggerated the risks of gradient inversion attacks in FL, and that our work builds a foundation towards a better understanding of the real risks of such attacks in real-world FL applications. In most cases, it becomes very difficult for an attacker to recover any training images, as common FL scenarios involve model updates from sufficiently large numbers of batch sizes, training images, and local iterations that detrimentally impact the success of current gradient inversion attacks. Furthermore, we listed several effective protection methods that could be easily incorporated into any FL system to render existing attacks futile.

We believe that our work is a step towards establishing reproducible methods of measuring data leakage in FL and therefore builds a framework for finding an optimal tradeoff between privacy and accuracy based on quantifiable metrics.

METHODS

Inversion attacks. Today's gradient inversion attacks follow a common principle. The main idea of their proposed algorithm is to match the gradients between the trainable input data and real data. If the attack has access to the state of the global model W and a client's model update ΔW, it can try to optimize the input image (usually randomly initialized) to produce a gradient that matches the observed model updates (see Figure 10).

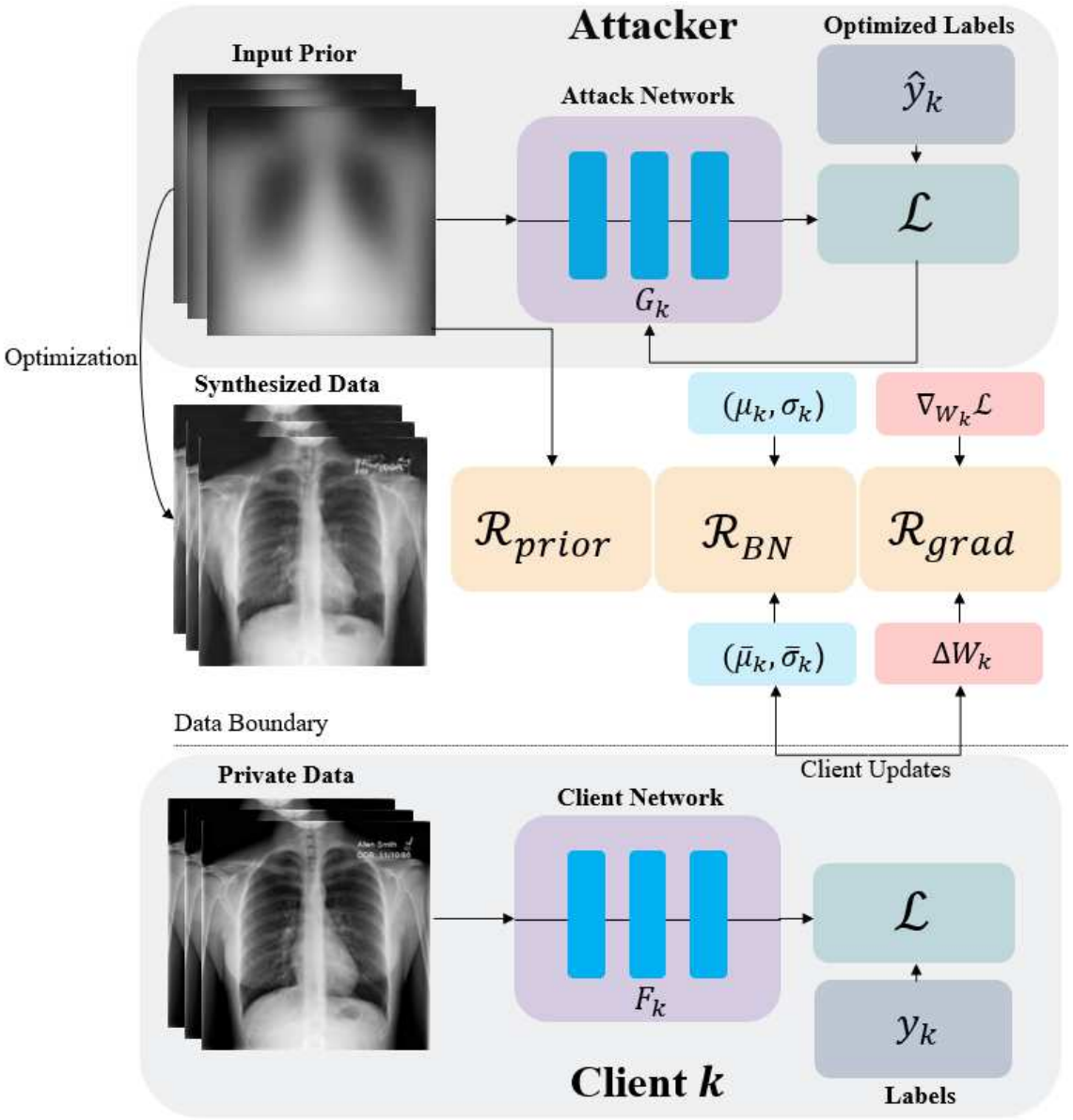

Figure 10 ***Overview of the attack model.*** *A malicious actor ("attacker") recovers private training data via updates sent from a client by optimizing an input prior. The objective aims to simulate the training by matching the intercepted model updates and BN statistics from a client. Additional image prior losses are added to enhance the quality of recovered images.*

Practical Gradient Inversion Attack. Although previous efforts have shown the possibility of reconstructing training data by matching the gradients and optimizing the randomly initialized input images, these attacks have been conducted in contrived settings. Prior methods simplified the attack scenario by sending model updates with fixed BN statistics. However, in realistic FL scenarios, these statistics change during training with multiple batches during an epoch of training before sending the updates to the server. Since the gradient inversion is formulated as a second-order optimization task, accumulation of error due to discrepancies in the estimation of BN statistics leads to suboptimal solutions. As a result, matching the gradients in practical use cases is extremely difficult, if not impossible, when taking these considerations into account, especially when the training takes more than one iteration. In this work, we study a real-world attack scenario in which the intercepted model updates from the clients are sent while updating the BN statistics.

In networks that leverage BN layers[19] in their architecture, during training, mini-batches of input data x are first normalized by the batch-wise mean μ and variance σ and transformed by an affinity mapping with trainable values β and γ according to $\frac{x-\mu(x)}{\sqrt{\sigma(x)+\epsilon}} \times \gamma + \beta$. It is customary to update running statistics of μ and σ during training according to $\overline{u}_{new} = (1 - \eta_{BN}) \times \overline{u} + \eta_{BN} \times u_{new}$ where η_{BN} is the momentum, $\overline{u}_{new}$, $\overline{u}$ denote the new and current estimated statistics, and u_{new} denotes the new update, respectively.

The BN layers contain a strong prior in the forms of tracked running statistics of the training data and must be effectively leveraged in gradient-based attack scenarios. During evaluation (i.e., inference), the estimated running mean and variance are used in lieu of computing on-the-fly batch-wise statistics.

Furthermore, previous efforts[14–16,35] have not taken into account the effect of running statistics in the BN layers and only utilized gradient updates that were sent by assuming fixed BN statistics. As a result, matching the gradient updates that are sent during the training phase is not possible since the running statistics from BN layers affect the gradient computation.

In our attack scenario, we consider the server as “honest but curious”[36]. Hence all global models, configuration files, local model updates and training BN statistics that exist on the server or are exchanged between the server and client can be considered as compromised for a potential gradient inversion attack.

Figure 10 illustrates an overview of our attack model. We formally define the attack scenario for one server-client configuration which can be extended to all participating clients in FL. Consider client k having local model F_k with weights W_k, the model updates ΔW_k are computed with respect to mini-batches from the local training images x_k and their corresponding labels y_k. In order to reconstruct the private data, we formulate an optimization scheme wherein trainable input images $\hat{x}_k$ and the corresponding labels $\hat{y}_k$ are employed to match the client’s updates by minimizing $argmin\,(\mathcal{R}_{grad}(\hat{x}_k;\, W_k, \Delta W_k) + \mathcal{R}_{BN}(\hat{x}_k) + \mathcal{R}_{prior}(\hat{x}_k))$; in which $\mathcal{R}_{grad}$ is the gradient matching loss between the intercepted gradient and gradients computed using $\hat{x}_k$ and $\hat{y}_k$ by enforcing $\sum_l || \nabla_{W_k^{(l)}} \mathcal{L}(\hat{x}_k, \hat{y}_k) - \Delta W_k^{(l)} ||_2$ for each layer l in both client network F_k and attack network G_k. We initialize the attack network G_k, which is used for computing the matching gradient, by the global weights in the server and update it with the model weight differences $\Delta W_k^{(l)}$ that are sent

from the client k. In addition, we update the intercepted BN updates from the client by computing the running statistics with batch-wise mean μ_k and variance σ_k and initialize the BN layers in the attack network with these updated statistics. We estimate the BN running statistics of each mini-batch by $\overline{\mu}_{new} = (1 - \eta_{BN}) \times \overline{\mu} + \eta_{BN} \times \mu_{new}$ and $\overline{\sigma}_{new} = (1 - \eta_{BN}) \times \overline{\sigma} + \eta_{BN} \times \sigma_{new}$ where $\overline{\mu}$ and $\overline{\sigma}$ denote running mean and variance respectively, and η_{BN} is set to 0.1. These considerations ensure that the attack model exactly simulates the client training procedure by matching the BN statistics during training, hence making it applicable to practical FL scenarios. Note that if the number of local iterations increases, the estimation of running statistics becomes less accurate and hence it becomes harder to properly match the gradients. This effect is illustrated in Figure 3 where we show the effect of the number of training images, batch size, and local iterations on the gradient inversion attack.

We employ $\mathcal{R}_{BN}$ to enforce matching the distributions of BN statistics that are obtained from the trainable input $\hat{x}_k$ and intercepted from the client updates according to $\sum_l ||\mu_k(\hat{x}_k)^{(l)} - \overline{\mu}_k^{(l)}||_2 + \sum_l ||\sigma_k(\hat{x}_k)^{(l)} - \overline{\sigma}_k^{(l)}||_2$ for each layer l in attack and client networks. We account for image prior losses by minimizing $\mathcal{R}_{prior}$, which includes the standard total variation and l_2 norm losses [21].

In addition, we jointly optimize to obtain the corresponding labels from a client's data by using a trainable label $\hat{y}_k$ and minimize a loss function such as cross-entropy that is used for supervising the classification networks.

For initializing the trainable parameters, and without loss of generality, we initialize $\hat{x}_k$ from a domain-specific prior (e.g., chest X-ray images) for faster convergence and avoiding sub-optimal solutions. However, we randomly initialize $\hat{y}_k$ from a uniform distribution in the interval between 0 and 1.

Differential privacy (DP). Most commonly, DP mechanisms add some form of calibrated noise to the data in order to obfuscate the contributions of individual data entries, i.e., patients in healthcare applications[37]. There are many variations of how one can add this noise, like the Gaussian mechanism or Laplacian mechanism[38]. Other, more sophisticated methods like the sparse vector technique (SVT) combine the addition of noise with value clipping and/or removal of parts of the data[39]. DP has been successfully integrated into the training of deep neural networks[40–42] and used for FL applications[18,27,43,44]. DP can make strong guarantees about the theoretical effectiveness of protecting against individual privacy loss but is difficult to tune to keep the model accuracy at an acceptable level[45].

Our work intentionally utilized a simple Gaussian mechanism to explore DP and to establish an upper bound for successful inversion attacks in the FL scenario. The intention was to illustrate how simply adding Gaussian noise to the model updates could already reduce the success of the inversion attack but at the cost of decreasing model performance.

At each round of FL, we compute the k^{th} percentile of the absolute model update values and multiply this value by σ_0, a tuneable hyperparameter that can be specified by the user and controls the overall amount of noise added to the model updates. The resulting σ is then used to sample

from a Gaussian distribution $N(0, \sigma)$ with zero-mean. Therefore, the model updates become $\widehat{\Delta W} = \Delta W + N(0, \sigma)$, where$\sigma = percentile(|\Delta W|, k)\sigma_0$, before they are sent to the central server for aggregation. In our experiments, we use $k = 95$, which was chosen to reduce the impact of outliers in the gradient computation, which in turn would cause too high an amount of noise and hinder the model convergence.

As this DP protocol happens locally on each client before the updates are sent to the server, this approach is commonly referred to as local DP[38]. In scenarios where only the protection against model inversion after FL training is of concern (e.g., when employing cryptographic techniques such as HE to make the FL aggregation step itself secure against inversion), "global" or "central" DP could be utilized to protect against model inversion after training is completed[46].

Future work could explore more sophisticated DP methods and how they impact the quality of inversion attacks and their impact on subsequent model inversion attacks.

Image Identifiability Precision

Image Identifiability Precision[15] is designed for measuring the image-specific features between the reconstructed images and original training data. In its original formulation, the metric is computed for a fixed number of randomly selected images (e.g., 256) by employing the gradient-inversion attacks for each batch of data. The deep feature embeddings of reconstructions, as well as the entire pool of training images are computed using a pretrained network (e.g., ResNet-18 trained on ImageNet). Furthermore, with a clustering algorithm (e.g., K-Nearest Neighbors), the closest training image in the feature embedding space (as measured by cosine similarity) of each reconstructed image is selected and fetched back as an exact match if it matches the original image. The IIP score is then computed in as $IIP = \frac{m}{N}$, where m is the number of unique exact matches according to the described procedure. Here, N denotes the size of the training pool (e.g., 256). In this work, we use a modified version of the IIP metric adjusted to the FL scenario in which the IIP score is computed for the entire training and validation images of all clients as opposed to the randomly selected samples. A reconstruction is counted as an exact match if the closest match originates from the attacked client. As a result, a diverse population of training images can be taken into consideration for IIP calculations.

Data availability

The data used for training, validation, and testing data is available at:
https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

Code availability

The Clara Train SDK 4.0 is used for local model training and evaluation:
https://ngc.nvidia.com/catalog/containers/nvidia:clara-train-sdk

NVIDIA FLARE is used to record model updates in a simulated FL setting:
https://nvidia.github.io/NVFlare

The inversion software will be made available upon acceptance.

References

1. Rieke, N. *et al.* The future of digital health with federated learning. *NPJ Digit Med* **3**, 119

(2020).

2. McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. in (eds. Singh, A. & Zhu, J.) vol. 54 1273–1282 (PMLR, 2017).
3. Roth, H. R. *et al.* Federated Learning for Breast Density Classification: A Real-World Implementation: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings. in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning* (eds. Albarqouni, S. et al.) vol. 12444 181–191 (Springer International Publishing, 2020).
4. Sheller, M. J. *et al.* Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Sci. Rep.* **10**, 12598 (2020).
5. Remedios, S. W., Butman, J. A., Landman, B. A. & Pham, D. L. Federated Gradient Averaging for Multi-Site Training with Momentum-Based Optimizers. in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning* 170–180 (Springer International Publishing, 2020).
6. Sarma, K. V. *et al.* Federated learning improves site performance in multicenter deep learning without data sharing. *Journal of the American Medical Informatics Association* vol. 28 1259–1264 (2021).
7. Dayan, I. *et al.* Federated learning for predicting clinical outcomes in patients with COVID-19. *Nat. Med.* **27**, 1735–1743 (2021).
8. Kairouz, P. *et al. Advances and Open Problems in Federated Learning*. (2021).
9. Li, T. *et al.* Federated Optimization in Heterogeneous Networks. *arXiv [cs.LG]* (2018).
10. Karimireddy, S. P. *et al.* SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. in *Proceedings of the 37th International Conference on Machine Learning* (eds. Iii, H. D. & Singh, A.) vol. 119 5132–5143 (PMLR, 2020).
11. Zhou, Z. *et al.* Towards Fair Federated Learning. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021) doi:10.1145/3447548.3470814.
12. Rothchild, D. *et al.* FetchSGD: Communication-Efficient Federated Learning with Sketching. in *International Conference on Machine Learning* 8253–8265 (PMLR, 2020).
13. Li, T., Sahu, A. K., Talwalkar, A. & Smith, V. Federated Learning: Challenges, Methods, and

Future Directions. *IEEE Signal Processing Magazine* vol. 37 50–60 (2020).

14. Geiping, J., Bauermeister, H., Dröge, H. & Moeller, M. Inverting Gradients -- How easy is it to break privacy in federated learning? (2020).
15. Yin, H. *et al.* See through Gradients: Image Batch Recovery via GradInversion. (2021).
16. Kaissis, G. *et al.* End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nature Machine Intelligence* vol. 3 473–484 (2021).
17. Yang, Q., Liu, Y., Chen, T. & Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **10**, 1–19 (2019).
18. Kaissis, G. *et al.* End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nature Machine Intelligence* **3**, 473–484 (2021).
19. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. (2015).
20. Zhu, L., Liu, Z. & Han, S. Deep Leakage from Gradients. in *Advances in Neural Information Processing Systems 32* (eds. Wallach, H. et al.) 14774–14784 (Curran Associates, Inc., 2019).
21. Yin, H. *et al.* See Through Gradients: Image Batch Recovery via GradInversion. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* 16337–16346 (2021).
22. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) doi:10.1109/cvpr.2016.90.
23. Deng, J. *et al.* ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009) doi:10.1109/cvprw.2009.5206848.
24. Chowdhury, M. E. H. *et al.* Can AI Help in Screening Viral and COVID-19 Pneumonia? *IEEE Access* vol. 8 132665–132676 (2020).
25. Rahman, T. *et al.* Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *Comput. Biol. Med.* **132**, 104319 (2021).
26. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* **13**, 600–612 (2004).
27. Li, W. *et al.* Privacy-Preserving Federated Brain Tumour Segmentation. *Machine Learning in*

Medical Imaging 133–141 (2019) doi:10.1007/978-3-030-32692-0_16.

28. van der Maaten, L. & Hinton, G. Visualizing Data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
29. Zhang, C. *et al.* BatchCrypt: Efficient Homomorphic Encryption for Cross-Silo Federated Learning. in *2020 {USENIX} Annual Technical Conference ({USENIX} {ATC} 20)* 493–506 (2020).
30. Li, X., Jiang, M., Zhang, X., Kamp, M. & Dou, Q. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. (2021).
31. Andreux, M., du Terrail, J. O., Beguier, C. & Tramel, E. W. Siloed Federated Learning for Multi-centric Histopathology Datasets. *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning* 129–139 (2020) doi:10.1007/978-3-030-60548-3_13.
32. Kaissis, G. A., Makowski, M. R., Rückert, D. & Braren, R. F. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence* vol. 2 305–311 (2020).
33. Reddi, S. *et al.* Adaptive Federated Optimization. (2020).
34. Usynin, D. *et al.* Adversarial interference and its mitigations in privacy-preserving collaborative machine learning. *Nature Machine Intelligence* **3**, 749–758 (2021).
35. Zhu, L. & Han, S. Deep Leakage from Gradients. *Lecture Notes in Computer Science* 17–31 (2020) doi:10.1007/978-3-030-63076-8_2.
36. Bonawitz, K. *et al.* Practical Secure Aggregation for Privacy-Preserving Machine Learning. in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* 1175–1191 (Association for Computing Machinery, 2017).
37. Dwork, C., McSherry, F., Nissim, K. & Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. in *Theory of Cryptography* 265–284 (Springer, Berlin, Heidelberg, 2006).
38. Yang, M., Lyu, L., Zhao, J., Zhu, T. & Lam, K.-Y. Local Differential Privacy and Its Applications: A Comprehensive Survey. (2020).
39. Lyu, M., Su, D. & Li, N. Understanding the Sparse Vector Technique for Differential Privacy. (2016).
40. Ziller, A. *et al.* Medical imaging deep learning with differential privacy. *Sci. Rep.* **11**, 13524 (2021).

41. Abadi, M. *et al.* Deep Learning with Differential Privacy. (2016) doi:10.1145/2976749.2978318.
42. Wu, B. *et al.* P3SGD: Patient Privacy Preserving SGD for Regularizing Deep CNNs in Pathological Image Classification. in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* 2099–2108 (2019).
43. Liang, Z., Wang, B., Gu, Q., Osher, S. & Yao, Y. Differentially Private Federated Learning with Laplacian Smoothing. (2020).
44. al., R.-B. et. Federated Learning and Differential Privacy: Software tools analysis, the Sherpa.ai FL framework and methodological guidelines for preserving data privacy. *Inf. Fusion* **64**, 270–292 (2020).
45. Bagdasaryan, E., Poursaeed, O. & Shmatikov, V. Differential Privacy Has Disparate Impact on Model Accuracy. *Adv. Neural Inf. Process. Syst.* **32**, (2019).
46. Naseri, M., Hayes, J. & De Cristofaro, E. Local and Central Differential Privacy for Robustness and Privacy in Federated Learning. (2020).