Hiding in Plain Sight: Disguising Data Stealing Attacks in Federated Learning

Kostadin Garov [*1] **Dimitar I. Dimitrov** [*2] **Nikola Jovanović** [2] **Martin Vechev** [2]

Abstract

Malicious server (MS) attacks have scaled data stealing in federated learning to more challenging settings. However, concerns regarding client-side detectability of MS attacks were raised, questioning their practicality once they are publicly known. In this work, we thoroughly study the problem of detectability for the first time. We show that most prior MS attacks, which fundamentally rely on one of two key principles, are detectable by principled client-side checks. Further, we propose SEER, a novel attack framework that is less detectable by design, and able to steal user data from gradients even for large batch sizes (up to 512) and under secure aggregation. Our key insight is the use of a secret decoder, jointly trained with the shared model to disaggregate in a secret space. Our work is a promising first step towards more principled treatment of MS attacks, paving the way for realistic data stealing that can compromise user privacy in real-world deployments.

1. Introduction

A long line of work on gradient leakage attacks in federated learning (FL) has shown that even passive servers can reconstruct client data from gradients, breaking the key privacy promise of FL; we survey such work in App. A. Notably, these attacks are only applicable to naive FL deployments (Huang et al., 2021)—under realistic assumptions, they are limited to small batch sizes and no secure aggregation (Bonawitz et al., 2016). In response, recent work has argued that the honest-but-curious threat model downplays the risks to FL, as servers can be compromised or malicious. This has led to *malicious server* (MS) attacks, which yield promising results by lifting honest attacks to large batches.

Most prior MS attacks rely on one of two key underlying principles (see App. A). One attack class (Fowl et al., 2022b; Zhao et al., 2023; Boenisch et al., 2021; Zhang et al., 2023) uses model modifications to promote sparsity in dense layer gradients, enabling the use of analytical honest attacks—we refer to such attacks as *boosted analytical*. Other attacks utilize *example disaggregation* (Pasquini et al., 2022; Wen et al., 2022), reducing the effective batch size in the gradient space by restricting the gradient flow for all examples but one, allowing the use of optimization-based honest attacks.

Client-side detectability Nearly all prior work in the field (Geiping et al., 2020; Fowl et al., 2022b; Zhao et al., 2023; Boenisch et al., 2021; Fowl et al., 2022a; Pasquini et al., 2022; Wen et al., 2022; Chu et al., 2023) has raised the issue of *client-side detectability* of MS attacks, i.e., an FL client may be able to detect malicious server activity, and decide to opt out of the current or all future rounds. Despite such concerns, no attempts have been made so far to study, quantify, or improve client-side detectability of MS attacks.

This work: detecting and disguising MS attacks As our first contribution, we study the question of client-side detectability of MS attacks. In Sec. 2, we demonstrate that while boosted analytical and example disaggregation attacks pose a real threat as zero-day exploits, now that their key principles are known, *all* current (and future) attacks from these two classes are client-side detectable in a principled way, bringing into question their practicality. We observe that such limitations of prior attacks arise from their fundamental reliance on the honest attacks they lift. Namely, boosted analytical attacks always require handcrafted modifications which are *weight-space detectable*, and example disaggregation attacks rely on the success of disaggregation, which is equally evident to any party observing the gradients, i.e., it is *gradient-space detectable*. This illustrates the need for fundamentally different attack approaches.

As a first step in that direction and our second contribution, we propose a novel attack framework, SEER (Sec. 3), which recovers data from batch sizes up to 512 (Sec. 4), yet does not lift any honest attack and is by design harder to detect. Our key insights are that (i) gradient-space detection can be evaded using a *secret decoder*, disaggregating the data in a space unknown to clients, and (ii) jointly training the decoder and the shared model with SGD *avoids handcrafted*

[*]Equal contribution [1]INSAIT, Sofia University [2]ETH Zurich. Correspondence to: Kostadin Garov <kostadin.garov@insait.ai>, Dimitar I. Dimitrov <dimitar.iliev.dimitrov@inf.ethz.ch>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

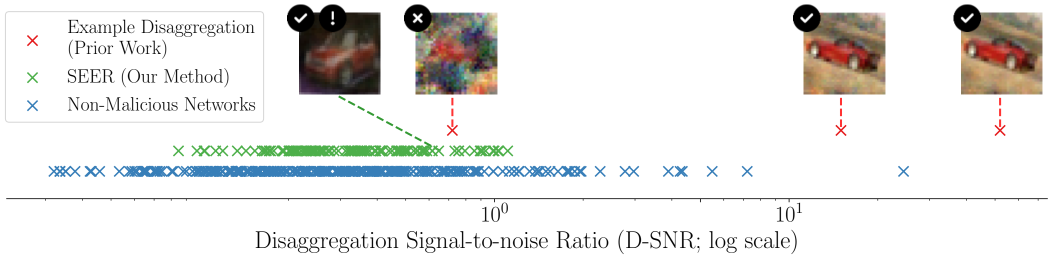

Figure 1. D-SNR (Sec. 2) of real (model, data batch) pairs. High values indicate vulnerability to data leakage, which can manifest even in non-malicious models (×). Example disaggregation attacks (×) are easily detectable as they can successfully reconstruct data (✓) only when DSNR is unusually high (note the log scale), and fail otherwise (✗). Our method, SEER (×, Sec. 3), successfully reconstructs a datapoint even when D-SNR is low (✓ !), and is thus hard to detect in the original gradient space.

modifications and allows effective reconstruction.

2. Detectability of Malicious Server Attacks

We now study the issue of client-side detectability of two MS attack classes introduced in Sec. 1, demonstrating that both are detectable using principled checks. We point out the underlying reasons for this, and discuss key desiderata that attacks should satisfy to be more practically viable.

Boosted analytical attacks Applying boosted analytical attacks to the realistic case of convolutional networks requires highly unusual *architectural modifications*, i.e., placing a large dense layer in front, making the attack obvious (Fowl et al., 2022b). The only alternative way to apply these attacks requires setting all convolutions to identity, such that the inputs are transmitted unchanged to the dense layer. As this is a pathological case that never occurs naturally, and requires handcrafted changes to most parameters (e.g., 98% of ResNet18), this approach is easily detectable by inspecting model weights (e.g., by searching for convolutional filters with a single nonzero entry, see Sec. 4). More importantly, it was shown (Fowl et al., 2022b) that relevant levels of transmission are not possible in realistic deep networks due to pooling and strides, which would further worsen by attempts to conceal the weight changes (e.g., by adding noise).

Example disaggregation attacks While the detectability of boosted analytical attacks in *weight space* was recognized in prior work, example disaggregation attacks were thought to be more promising. We argue example disaggregation attacks are also fundamentally limited, as they are efficiently detectable in the *gradient space* using a principled metric.

We now propose one such metric—*disaggregation signal-to-noise ratio (D-SNR)*. Assuming the standard cross-entropy loss $\mathcal{L}(x, y)$, a shared model with parameters $\boldsymbol{\theta}$, and a batch $D = \{(x_1, y_1), \ldots, (x_B, y_B)\}$, we define:

$$D\text{-}SNR(\boldsymbol{\theta}, D) = \max_{W \in \boldsymbol{\theta}_{lw}} \frac{\max_{i \in \{1,\ldots,b\}} \|g_i\|}{\sum_{i=1}^{b} \|g_i\| - \max_{i \in \{1,\ldots,b\}} \|g_i\|},$$

where $g_i = \frac{\partial \mathcal{L}(x_i, y_i)}{\partial W}$ and $\boldsymbol{\theta}_{lw}$ denotes the set of weights of all linear layers (dense and convolutional). Intuitively, D-SNR looks for *any* layer where the batch gradient is dominated by the gradient of a single example, implying disaggregation. While we focus on the case of disaggregating a single point, our approach can be easily generalized.

We use D-SNR to study the detectability of example disaggregation attacks in a realistic setting (see Sec. 4 for experimental details). As D-SNR is always ∞ for attacks proposed in Wen et al. (2022), we modify them in an attempt to smoothly control the strength of gradient flow restriction. Our key observation (Fig. 1, red ×), is that in all cases where the attack succeeds, D-SNR is unusually large, making the attack easily detectable. Reducing the attack strength causes a sharp drop in D-SNR, entering the range of most non-malicious networks (blue ×), i.e., the attack is undetectable. However, in all such cases the attack fails.

In rare cases (e.g., when overfitting), even natural networks can produce high D-SNR and get flagged. This behavior *is desirable*, as such networks indeed disaggregate a single example, and are thus (unintentionally) exposing sensitive user data. Thus, metrics such as D-SNR should be interpreted as detecting *vulnerability*, and not necessarily *maliciousness*.

Discussion Our results show that all prior attacks are detectable with general checks. We argue that this is caused by fundamental problems in the two attack classes once their principles are known, and can not be remedied by refinements. Any attempt to lift an honest analytical attack will inherit the limitation of being inapplicable to convolutions, and will require architectural changes or handcrafted modifications detectable in the weight space. Lifting optimization-based attacks always requires example disaggregation, which is gradient space detectable. More broadly, as all information needed to execute these attacks is in the user gradients, the server has no informational advantage, and no principled way to conceal the malicious intent. This poses the question of finding novel attack principles that better exploit the potential of the MS threat model, while targeting realistic settings (convnets, large batches, no pro-

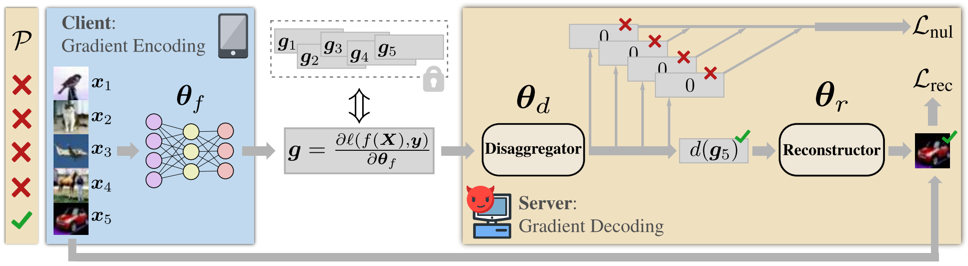
$\mathcal{P}$
Client: Gradient Encoding
x_1
x_2
x_3
x_4
x_5
θ_f
g_1 g_2 g_3 g_4 g_5
$g = \frac{\partial \ell(f(X), y)}{\partial \theta_f}$
θ_d
Disaggregator
θ_r
Reconstructor
$d(g_5)$
0
$\mathcal{L}_{\text{nul}}$
$\mathcal{L}_{\text{rec}}$
Server: Gradient Decoding

Figure 2. Overview of SEER. A client propagates a batch $\boldsymbol{X}$ (where one image satisfies the property $\mathcal{P}$ only known to the server) through the shared model f with malicious weights $\boldsymbol{\theta}_f$, and returns the aggregated gradient $\boldsymbol{g}$, hoping that aggregation protects individual images. The server steals the image that satisfies $\mathcal{P}$ by applying a secret disaggregator d to remove the impact of other images in a hidden space, followed by a secret reconstructor r. SEER is trained by jointly optimizing $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_d$, and $\boldsymbol{\theta}_r$ to minimize a combination of $\mathcal{L}_{\text{nul}}$ and $\mathcal{L}_{\text{rec}}$.

tocol changes or unrealistic FL assumptions) and explicitly considering weight and gradient space detection.

3. SEER Attack Framework

As a first step towards the above goal, we propose *Data Stealing via Secret Embedding and Reconstruction* (SEER), a novel attack framework that does not lift honest attacks, and does not use handcrafted modifications. Instead, it fine-tunes the shared model and the secret modules with SGD, which evades obvious weight space detection. Further, we avoid straightforward gradient space detection by disaggregating the data in a hidden space defined by a server-side *secret decoder*—in Fig. 1, SEER-trained networks (green ✕) have D-SNR values similar to those of natural networks. Additionally, SEER requires no architecture or protocol changes, and does not assume unrealistic knowledge of BN statistics or batch labels (Huang et al., 2021).

Mounting SEER We first describe the stealing of data with SEER once trained (Fig. 2). The client propagates their batch $(\boldsymbol{X}, \boldsymbol{y})$ of B examples $(\boldsymbol{x}_i, y_i)$ through the shared model f with parameters $\boldsymbol{\theta}_f$ sent by the server, and returns the gradient $\boldsymbol{g} = \frac{\partial \ell(f(\boldsymbol{X}), \boldsymbol{y})}{\partial \boldsymbol{\theta}_f}$ of the public loss ℓ. Due to aggregation provided by large B, $\boldsymbol{g}$ is a mixture of gradients $\boldsymbol{g}_i$ of individual examples, i.e., $\boldsymbol{g} = (1/B)\sum_{i=1}^{B} \boldsymbol{g}_i$.

Aiming to break this aggregation, the server passes $\boldsymbol{g}$ through a *secret decoder*, i.e., a *disaggregator* d, and a *reconstructor* r. If trained well, this reconstructs a client image. The key concept that enables this is the *secret property* $\mathcal{P}$ (far left in Fig. 2), which the server picks before training, keeps secret, and uses for disaggregation. While the exact choice of $\mathcal{P}$ is not essential, the goal is to pick the property such that, as often as possible, *only one example in the batch satisfies it*. Let $I_{\text{nul}} \subseteq [B] \coloneqq \{1, \dots, B\}$ denote examples that do not satisfy $\mathcal{P}$, and $I_{\text{rec}} \subseteq [B]$ those that do.

To this end, we make the observation that batch normalization (BN) enables an improvement over prior work (Fowl et al., 2022b) which relies on properties that hold *probabilistically* with probability of success $\frac{1}{e} \approx 37\%$ when the batch size goes to ∞. We define *local* properties which empirically hold *almost always* (for the case of big batches with no secure aggregation). See App. B for more details. In Fig. 2 (and our experiments) we set $\mathcal{P}$ =“the *least bright* image in the batch”, and observe (far left) that only $\boldsymbol{x}_5$ satisfies it.

The disaggregator d should ideally map $\boldsymbol{g}$ to a hidden space in which embeddings of the images that violate $\mathcal{P}$ are removed, leaving only the projection of the target image (in our example, $d(\boldsymbol{g}_5)$). This allows the reconstructor r to steal a client image $\boldsymbol{x}_5$, shown on the far right.

Training SEER SEER interprets gradient computation as a latent space encoding of client data. While poor scalability of honest attacks suggests that this encoding is often insufficient to recover user data, we observe that the MS threat model allows us to overcome this by *controlling the encoding* by fine-tuning $\boldsymbol{\theta}_f$ with SGD (no handcrafted changes!) to support downstream disaggregation and reconstruction.

The main goal of training is for d to nullify the contributions of images not satisfying $\mathcal{P}$, disaggregating the batch in a secret lower-dimensional *embedding space*, addressing the key limitation of previous disaggregation attacks. To this end, d uses a linear map $\boldsymbol{\theta}_d$, driving the gradients of I_{nul} to ≈ 0 when they are in or close to the null space of $\boldsymbol{\theta}_d$. Using the additivity of the linear map we aim to get:

$$d(\boldsymbol{g}) = d(\sum_{i=1}^{B} \boldsymbol{g}_i) = \sum_{i \in I_{\text{nul}}} d(\boldsymbol{g}_i) + \sum_{i \in I_{\text{rec}}} d(\boldsymbol{g}_i) \approx \sum_{i \in I_{\text{rec}}} d(\boldsymbol{g}_i).$$

To achieve this, we minimize the following objective:

$$\mathcal{L}_{\text{nul}} = \sum_{i \in I_{\text{nul}}} \| \, d(\boldsymbol{g}_i) \, \|_2^2.$$

To ensure that this does not also nullify the gradient of the example of interest in I_{rec}, we train the reconstructor r to reconstruct the client image that satisifes $\mathcal{P}$ from $d(\boldsymbol{g})$, i.e., the (noisy) isolated embedding of the target image gradient. We define the following reconstruction objective:

$$\mathcal{L}_{\text{rec}} = \| \, r(d(\boldsymbol{g}_{\text{rec}})) - \boldsymbol{x}_{\text{rec}} \, \|_2^2.$$

Table 1. Multi-client reconstruction on the bright and dark properties using different number of clients C on CIFAR10 and CIFAR100 datasets, for different total numbers of points. We report the percentage of correctly reconstructed images (*Rec*) and the average PSNR across all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

	CIFAR10, Bright, $C=1$			CIFAR100, Dark, $C=1$			CIFAR10, Bright, $C=4$			CIFAR10, Bright, $C=8$		
#Pts	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑
64	89.4	**32.1 ± 2.0**	27.2 ± 5.3	**97.0**	**32.8 ± 1.3**	**29.3 ± 4.1**	41.4	27.2 ± 3.7	19.3 ± 6.8	41.3	**27.2 ± 3.4**	19.5 ± 6.5
128	**94.2**	31.9 ± 1.7	**28.2 ± 4.3**	95.9	30.5 ± 1.3	26.8 ± 3.8	44.2	26.5 ± 3.1	19.4 ± 6.2	40.6	26.6 ± 3.7	19.2 ± 6.5
256	81.3	29.0 ± 2.1	24.4 ± 4.9	83.3	25.7 ± 1.3	22.2 ± 3.4	51.9	**27.3 ± 2.6**	**20.3 ± 6.2**	41.9	25.5 ± 3.2	18.6 ± 6.1
512	87.8	26.6 ± 1.8	23.2 ± 3.5	62.8	24.3 ± 1.5	20.2 ± 4.0	**52.9**	25.7 ± 2.2	20.0 ± 5.2	**51.7**	26.2 ± 3.2	**20.2 ± 5.5**

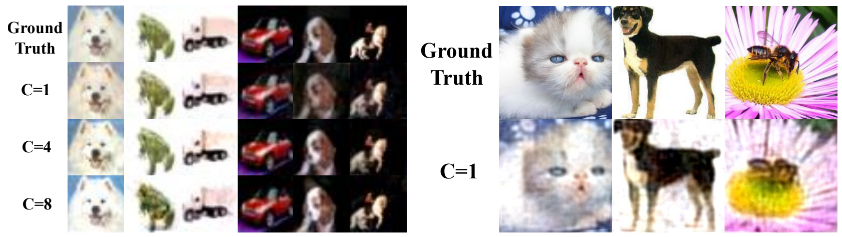

Figure 3. Results of SEER with 128 examples and different #clients C on CIFAR10 (left) and 64 examples on ResImageNet (right).

Finally, SEER trains all three components jointly to minimize the linear combination of the two losses: $\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha \cdot \mathcal{L}_{\text{nul}}$. Complete algorithms describing the training and mounting of SEER are deferred to App. F.

4. Experimental Evaluation

In all our experiments, we use ResNet18 (He et al., 2016) and consider 3 datasets—CIFAR10, CIFAR100 (Krizhevsky et al., 2009) (where we use a linear r) and Restricted ImageNet (*ResImageNet*) (Tsipras et al., 2018) (where we use a linear version of the U-Net decoder (Ronneberger et al., 2015), see App. C. We use the properties of maximal/minimal brightness (*Bright* and *Dark*) and three reconstruction quality metrics: (i) average PSNR, (ii) fraction of reconstructed images (PSNR > 19, *Rec*), and (iii) the average PSNR for the top $\frac{1}{e} \approx 37\%$ of the batch reconstructions, relevant for our multi-client experiments where we also rely on probabilistic properties as in (Fowl et al., 2022b).

Results A reduced version of our main results on single and multiple-client setups are given in Table 1. Our full results including more results with the *Dark* property and CIFAR100 follow similar trends and are deferred to App. H.1.

We make several key observations. First, in our single-client experiments, our local property allows us to reconstruct a large percentage of the batches (up to 97%), which is much higher than the theoretical limit $\frac{1}{e}$ of probabilistic disaggregation, used in prior work. Further, despite observing the expected trend that the quality of reconstruction degrades with the increased batch size B, our attack works well for wide range of batch sizes, even reconstructing up to 87.8% of client batches of size as large as 512.

Third, we compare the results on the CIFAR10 and CIFAR100 datasets and notice no discernible difference in performance except for $B = 512$, where CIFAR10 is better.

Our multi-client attack consistently obtains average PSNR > 25 on the top 37% of images, recovering them almost perfectly. We observe a slight degradation of success rate with C, as expected. We show example attacks in Fig. 3 (left), confirming visually the potency of SEER, with success rate being the only clear difference between the two settings.

In Fig. 3 (right), we show our ResImageNet results, which demonstrate that SEER scales to high-resolution images, by training a model with $B = 64$. We obtain average PSNR of 20.6 ± 3.7 and 23.8 ± 1.4 on all and the top 37% of the images, respectively, corresponding to 77% successfully attacked batches. We note that these results are significant, as reconstructing even a single ImageNet image is not possible with honest attacks without restrictive assumptions of BN statistics and label data (Huang et al., 2021).

Measuring D-SNR To produce Fig. 1 we considered 4 SEER-trained malicious models (CIFAR10, *Bright*/*Dark*, batch size 128/256), as well as 8 checkpoints made at various points during natural training, using the same initialization as used for SEER. Then, for each value of $B \in \{16, 32, 64\}$ we chose 5 random batches of size B from the training set, and 5 random batches of size B from the test set of CIFAR-10. For each batch, we computed the D-SNR on each of the 12 networks and plotted the resulting value in Fig. 1 (blue for natural and green for SEER networks). For example disaggregation attacks, we used a publicly available implementation of the attacks of (Wen et al., 2022) and modified the *multiplier* parameter to control the strength of the attack. We used the default setting where batches are chosen such that all images belong to the same class (*car* in this case). The three reconstructions of the example disaggregation attack were obtained by running the modernized variant of the attack of Geiping et al. (2020),

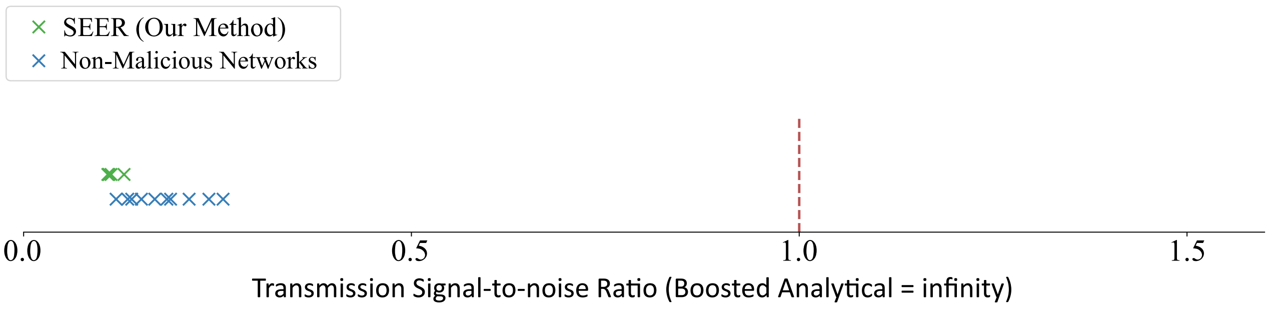

Figure 4. The transmission signal-to-noise ratio of several SEER-trained and naturally trained networks. The same metric has a value of ∞ for all boosted analytical attacks.

implemented in the Breaching framework (Geiping et al., 2023), on the disaggregated batch. Finally, to allow for better comparison, for SEER we chose to show a reconstruction from the same class (a car) that has a D-SNR slightly below the D-SNR of the leftmost example disaggregation point (0.72). To do this, we used the *Dark* property and the dark car image from Fig. 3, and sampled the other 63 points in the batch randomly from the test set until the D-SNR fell in the $[0.62, 0.72]$ range. We stopped as soon as we found such a batch and reported the reconstruction produced by SEER.

Measuring transmission As noted in Sec. 2, to be applicable to convolutional networks, boosted analytical attacks require handcrafted changes to convolutional layers that simply transmit the inputs unchanged. While, as noted above, even in the ideal case this cannot lead to good reconstructions, we illustrate the point that such change is detectable by defining a metric similar to D-SNR, which can be interpreted as a *transmission signal-to-noise ratio*, measured on the first convolutional layer. Namely, for each filter in the first convolutional layer, we divide the absolute values of the largest entry, and the sum of absolute values of all other entries.

$$TSNR(\boldsymbol{\theta}) = \max_{K \in \boldsymbol{\theta}_{c_1}} \frac{\hat{K}}{\sum_{i=1}^{h} \sum_{j=1}^{w} |K_{i,j}| - \hat{K}}$$

where $\hat{K} = \max_{i \in \{1,\ldots,h\}, j \in \{1,\ldots,w\}} |K_{i,j}|$ and $\boldsymbol{\theta}_{c_1}$ denotes the set of weights of all kernels in the first convolutional layer. Intuitively, we treat the entry with the largest absolute value as the signal, and measure how well this is transmitted by the filter. The ratio is high when the filter transmits the input unchanged, and is ∞ for the handcrafted changes used by the boosted analytical attacks. We compute this metric on the 12 networks used in Fig. 1 (see previous paragraph) and show the results in Fig. 4. Intuitively, the red line at 1.0 indicates the case where there are equal amounts of the pixel being transmitted and all other pixels. We can observe that all networks have values below 0.3, confirming that transmission is indeed unusual and not a case that ever happens naturally, implying that if boosted analytical attacks that use this technique would be able to obtain good results, they still would be easily weight space detectable.

5. Conclusion and Outlook

In this work, we studied client-side detectability of MS attacks in FL. We showed that prior attacks are detectable in a principled way, and proposed SEER, a novel attack framework that effectively steals data while being less detectable.

While SEER is a powerful attack that can harm user privacy, potentially demonstrating disparate impact (Wen et al., 2022), we believe our work opens the door to a more principled investigation of defenses, as it illustrates that techniques such as secure aggregation are not as effective as previously thought. To mitigate attacks such as SEER, prior work has discussed differential privacy methods such as DPSGD (Abadi et al., 2016), which in the global model is incompatible with the MS setting as it requires a trusted aggregator, and in the local model (Truex et al., 2020) is known to degrade utility (Wei et al., 2019). Further, cryptographic techniques such as SMPC or FHE are applicable in theory, but still largely impractical (Kairouz et al., 2019).

Thus, we believe that principled client-side detection is the most promising way forward. While SEER avoids pitfalls of prior attacks which made them easily detectable, and we see no clear ways to detect it in its current form, more sophisticated detection techniques may be able to do so. We encourage such work, and advocate for efficient and robust checks accompanied by categorical analyses of attack classes (such as in our work), as opposed to ad-hoc detection which attack refinements can easily adapt to. On the attack side, interesting future directions include applying SEER to other data modalities and model architectures, improving its computational costs, investigating its dependence on in-distribution data, and attempting to generalize reconstruction beyond a single property.

References

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *CCS*, 2016.

Balunovic, M., Dimitrov, D. I., Jovanovic, N., and Vechev, M. T. LAMP: extracting text from gradients with language model priors. In *NeurIPS*, 2022a.

Balunovic, M., Dimitrov, D. I., Staab, R., and Vechev, M. T. Bayesian framework for gradient leakage. In *ICLR*, 2022b.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. When the curious abandon honesty: Federated learning is not private. *arXiv*, 2021.

Boenisch, F., Dziedzic, A., Schuster, R., Shamsabadi, A. S., Shumailov, I., and Papernot, N. Reconstructing individual data points in federated learning hardened with differential privacy and secure aggregation. *arXiv*, 2023.

Bonawitz, K. A., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. *NIPS*, 2016.

Chu, H.-M., Geiping, J., Fowl, L. H., Goldblum, M., and Goldstein, T. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. *ICLR*, 2023.

Fowl, L., Geiping, J., Reich, S., Wen, Y., Czaja, W., Goldblum, M., and Goldstein, T. Decepticons: Corrupted transformers breach privacy in federated learning for language models. *ICLR*, 2022a.

Fowl, L. H., Geiping, J., Czaja, W., Goldblum, M., and Goldstein, T. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *ICLR*, 2022b.

Fung, C., Yoon, C. J. M., and Beschastnikh, I. The limitations of federated learning in sybil settings. In *RAID*, 2020.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *NeurIPS*, 2020.

Geiping, J., Fowl, L., and Wen, Y. Breaching - a framework for attacks against privacy in federated learning. `https://github.com/JonasGeiping/breaching`, 2023.

Geng, J., Mou, Y., Li, F., Li, Q., Beyan, O., Decker, S., and Rong, C. Towards general deep leakage in federated learning. *arXiv*, 2021.

Gupta, S., Huang, Y., Zhong, Z., Gao, T., Li, K., and Chen, D. Recovering private text in federated learning of language models. In *NeurIPS*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. Evaluating gradient inversion attacks and defenses in federated learning. In *NeurIPS*, 2021.

Jin, X., Chen, P., Hsu, C., Yu, C., and Chen, T. CAFE: catastrophic data leakage in vertical federated learning. *arXiv*, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *arXiv*, 2019.

Kariyappa, S., Guo, C., Maeng, K., Xiong, W., Suh, G. E., Qureshi, M. K., and Lee, H. S. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. *arXiv*, 2022.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lam, M., Wei, G., Brooks, D., Reddi, V. J., and Mitzenmacher, M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *ICML*, 2021.

Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

Pasquini, D., Francati, D., and Ateniese, G. Eluding secure aggregation in federated learning via model inconsistency. In *CCS*, 2022.

Phong, L. T., Aono, Y., Hayashi, T., Wang, L., and Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Secur.*, (5), 2018.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015.

Truex, S., Liu, L., Chow, K. H., Gursoy, M. E., and Wei, W. Ldp-fed: federated learning with local differential privacy. In *EdgeSys@EuroSys*, 2020.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Vero, M., Balunovic, M., Dimitrov, D. I., and Vechev, M. T. Data leakage in tabular federated learning. *ICML*, 2022.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *arXiv*, 2019.

Wen, Y., Geiping, J., Fowl, L., Goldblum, M., and Goldstein, T. Fishing for user data in large-batch federated learning via gradient magnification. In *ICML*, 2022.

Wu, H., Zhao, Z., Chen, L. Y., and van Moorsel, A. Federated learning for tabular data: Exploring potential risk to privacy. In *ISSRE*, 2022.

Wu, R., Chen, X., Guo, C., and Weinberger, K. Q. Learning to invert: Simple adaptive attacks for gradient inversion in federated learning. *arXiv*, 2021.

Ye, J., Maddi, A., Murakonda, S. K., Bindschaedler, V., and Shokri, R. Enhanced membership inference attacks against machine learning models. In *CCS*, 2022.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. See through gradients: Image batch recovery via gradinversion. In *CVPR*, 2021.

Yue, K., Jin, R., Wong, C., Baron, D., and Dai, H. Gradient obfuscation gives a false sense of security in federated learning. *arXiv*, 2022.

Zhang, S., Huang, J., Zhang, Z., and Qi, C. Compromise privacy in large-batch federated learning via malicious model parameters. In *ICA3PP*, 2023.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv*, 2020.

Zhao, J. C., Sharma, A., Elkordy, A. R., Ezzeldin, Y. H., Avestimehr, S., and Bagchi, S. Secure aggregation in federated learning is not private: Leaking user data at large scale through model modification. *arXiv*, 2023.

Zhu, J. and Blaschko, M. B. R-GAP: recursive gradient attack on privacy. In *ICLR*, 2021.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. In *NeurIPS*, 2019.

A. Related Work

In this section, we discuss prior work on gradient leakage attacks in federated learning.

Honest server attacks In the honest-but-curious setting, the existing attacks fall into three types. *Optimization-based attacks* (Zhu et al., 2019; Zhao et al., 2020; Geiping et al., 2020; Geng et al., 2021; Wu et al., 2021; Yin et al., 2021) optimize a dummy batch with SGD to match the user-provided gradient. *Analytical attacks* (Phong et al., 2018; Kariyappa et al., 2022) can recover inputs of linear layers in closed form, but are limited to batch size $B = 1$ and do not support convolutional networks. *Recursive attacks* (Zhu & Blaschko, 2021) extend analytical attacks to convolutional networks but are limited to $B \leq 5$. Several works thoroughly study these attacks (Yue et al., 2022; Balunovic et al., 2022b; Jin et al., 2021; Huang et al., 2021). Crucially, Huang et al. (2021) show that in realistic settings, where clients do not provide batchnorm statistics and labels, existing attacks are limited to $B < 32$ for low-resolution data, and fail even for $B = 1$ on high-resolution data. This implies that large B and secure aggregation (Bonawitz et al., 2016) are effective in privacy protection against honest attacks.

Malicious server (MS) attacks As discussed in Sec. 2, throughout the main paper we focus on analysing prior MS attacks through the lens of the broadly applicable classes of boosted analytical (Fowl et al., 2022b; Zhao et al., 2023; Boenisch et al., 2021; Zhang et al., 2023) and example disaggregation attacks (Wen et al., 2022; Pasquini et al., 2022). Here, we reflect on other MS attacks that study more specific or orthogonal settings. Some works (Zhao et al., 2023; Pasquini et al., 2022) require the additional ability to send different updates to different users, which was shown easy to overcome with reverse aggregation (Pasquini et al., 2022). The attack of Lam et al. (2021) focuses on the rare setting where participation side-channel data is present. While we focus on images, various attacks consider other modalities such as text (Balunovic et al., 2022a; Gupta et al., 2022; Fowl et al., 2022a; Chu et al., 2023) or tabular data (Wu et al., 2022; Vero et al., 2022). Similarly, we consider only the threat of data reconstruction—some works (Pasquini et al., 2022) study important but strictly weaker privacy notions such as membership (Ye et al., 2022) or property inference (Melis et al., 2019). Finally, another direction orthogonal to our work concerns the notably stronger threat model of sybil-based attacks (Fung et al., 2020; Boenisch et al., 2023).

B. Choosing the Property $\mathcal{P}$

In this section, we describe how we adapt our algorithm to the single and multi-clients settings by using different properties $\mathcal{P}$. In our experiments, for both variants of SEER we use properties $\mathcal{P}$ based on individual image brightnesses, inspired by Fowl et al. (2022b), who define $\mathcal{P}$ by a threshold on the *global* brightness distribution of the dataset, such that I_{rec} covers probability mass $\frac{1}{B}$. Fowl et al. (2022b) shows that this choice maximizes the probability of successful client data disaggregation $P(|I_{\text{rec}}| = 1)$, which approaches $\frac{1}{e} \approx 37\%$ as $B \to \infty$.

Single-client properties In the single-client setting, we improve upon Fowl et al. (2022b) by making the observation that using batch normalization (BN, present in most convolutional networks), allows us to choose $\mathcal{P}$ such that, empirically, $|I_{\text{rec}}| = 1$ *for nearly all batches*, for B as large as 512. This significantly improves the above probabilistic solution, as it implies that in most cases a single FL round is sufficient to steal client data. This is possible as each BN layer normalizes the distribution of its input, intertwining the computational graphs of images in the batch, which are otherwise independent. With this in mind, we define $\mathcal{P}$ with respect to the *local* distribution, e.g., as the maximal brightness in the batch. As shown in Sec. 4, we find that SEER can learn such $\mathcal{P}$ when trained with auxiliary data, almost always singling out the brightest image at attack time.

Multi-client properties We remark that when Secure Aggregation (Bonawitz et al., 2016) is applied in the multi-client setting, BN layers no longer intertwine all inputs, and inputs across different clients remain independent. This observation suggests Secure Aggregation in the presence of BN layers can be a strictly more powerful defence than using large batch sizes coming from a single client, an aspect that was overlooked in prior work (Wen et al., 2022; Fowl et al., 2022b).

To overcome this challenge, we design a more elaborate $\mathcal{P}$ for the multi-client setting that uses a combination of local and global properties of the brightness distribution. In particular, we define $\mathcal{P}$ in terms of a brightness range that we generated to be invariant to the mean and the standard deviation of a *single client* batch, while still having probability mass equal to one over the *total number* of aggregated datapoints. As finding those ranges still relies on the global distribution, our reconstruction in this setting is probabilistic with probability $\frac{1}{e}$, similarly to prior work. Next, we give detailed description of our brightness range computation.

To simplify our explanation, we focus on the particular brightness range corresponding to the most bright images, which in turn simplifies our range computation to calculating a single brightness threshold τ. To calculate τ, we use the insight that individual client gradients are still generated in the presence of BN before their aggregation. To this end, we normalize the brightnesses within individual client batches of size B for 20000 sampled client batches and use the sampled normalized brightness to generate the cumulative density function (CDF) of their empirical distribution. We then choose the threshold τ on this distribution, to maximize the probability that exactly one out of the C aggregated clients has exactly one image with normalized-brightness above the threshold. We estimate the probability of having exactly one image with normalized-brightness above the threshold as:

$$(1 - \Phi_1(\tau)) * \Phi_2(\tau) * \Phi_1(\tau)^{C-1}$$

where Φ_1 is the CDF of the top-brightness in a sampled batch, and Φ_2 is the CDF of the second highest brightness in a sampled batch. Up to a multiplicative constant, the probability equation can be intuitively rephrased as follows—for exactly one client the highest normalized brightness within its batch is above τ *and* the second highest brightness is below τ, while for the rest *all* brightnesses are below τ. To optimize the equation above for the threshold τ, we use the golden section search method—a numerical optimization technique that repeatedly divides a search interval by the golden ratio to efficiently locate the (possibly-local) extremum of a function of a single variable.

C. U-Net-based Image Reconstructor

In this section, we explain the architecture of our image reconstructor r used in our ResImageNet experiments in Sec. 4. Our architecture is inspired by the decoder portion of an U-Net (Ronneberger et al., 2015), which has been demonstrated to be a memory efficient architecture for generating images.

We show our architecture in Fig. 5. In the figure, $\boldsymbol{g}$ depicts our model's gradient subsampled randomly so that only 3% of its entries are kept (See App. E.1). There are two main differences between our r in Fig. 5 and the original 6 layer U-Net architecture. First, we use no activation functions, thus creating a (sparse) linear reconstructor function r. This allows us, similarly to our CIFAR10/100 experiments, to combine r and d into single linear layer, whose bias becomes the target for our filtered out inputs $\boldsymbol{X}_{\text{nul}}$. Second, as our architecture does not include U-Net-style encoder, the U-connections of our reconstructor are substituted by pairs of linear layers with bottleneck in the middle applied on $\boldsymbol{g}$. In Fig. 5, we depict the bottleneck sizes and transposed convolution sizes, as well as the intermediate output sizes of the different layers. The bottlenecks ensure that the memory efficiency of our method is preserved and are inspired by the intuition that the U-connections only need to provide high frequency content which can live in a much lower-dimensional subspace. Finally, we note that for the purpose of pretraining, we used the first 3 channels of the third transposed convolution layer as our CIFAR10 output and that our transposed convolution stack produces images of size 264×264 which we then center-crop to produce our ImageNet-sized final output.

D. Efficiently Computing $\mathcal{L}_{\text{nul}}$

In this section, we detail why the disaggregation loss $\mathcal{L}_{\text{nul}}$, as presented in Sec. 3, is inefficient to compute and how we approximate it to alleviate this issue. Computing $\mathcal{L}_{\text{nul}}$ directly for large batch sizes B takes a lot of memory due to the need to store $\boldsymbol{g}_i$ for all i in the large set I_{nul}. Note that the reason for this is that we want to enforce all of the individual gradient $\boldsymbol{g}_i$ to fall in the null space of θ_d separately. In practice, to save memory we enforce the same condition by computing the surrogate:

$$\widehat{\mathcal{L}}_{\text{nul}} = \| \frac{1}{|I_{\text{nul}}|} \sum_{i \in I_{\text{nul}}} d(\boldsymbol{g}_i) \|_2^2 + \| d(\boldsymbol{g}_{j \sim I_{\text{nul}}}) \|_2^2,$$

where the first part of the equation nullifies the mean gradient, and the second part nullifies a different randomly chosen gradient at every SGD step. In practice, this achieves similar results to the original loss $\mathcal{L}_{\text{null}}$ in that over time all gradients in I_{nul} go to 0.

E. Additional Implementation Details

In this section, we provide a few additional implementation details about our solution.

E.1. Subsampled Gradients

In order to save memory and computation, we use only part of the entries in our full model gradient $\boldsymbol{g}$ to construct our intermediate disaggregation space $\mathbb{R}^{n_d}$. In particular, we randomly sample 0.1% of the gradient entries of each of the model's parameters while ensuring that at least 8400 entries per parameter are sampled for our CIFAR10/100 experiments, and 2% and at least 9800 entries for our ResImageNet experiments. This results in 1.6% of the total gradient entries for CIFAR10/100 and 3.0% for ResImageNet. We theorize that we are able to reconstruct nearly perfectly with such small percent of the gradient entries because there is large redundancy in the information different gradient entries provide.

E.2. Trainset Data Augmentation

For the purpose of training our encoder-decoder framework, we observed data augmentation of our auxiliary dataset is crucial, especially for large batch sizes B. We theorize that the reason for this is the lack of diversity in the reconstruction samples $\boldsymbol{X}_{\text{rec}}$. In particular, as B grows, an increasingly smaller set of images are selected to be the brightest or darkest of any batch sampled from the training set. To this end when sampling our training batches, for

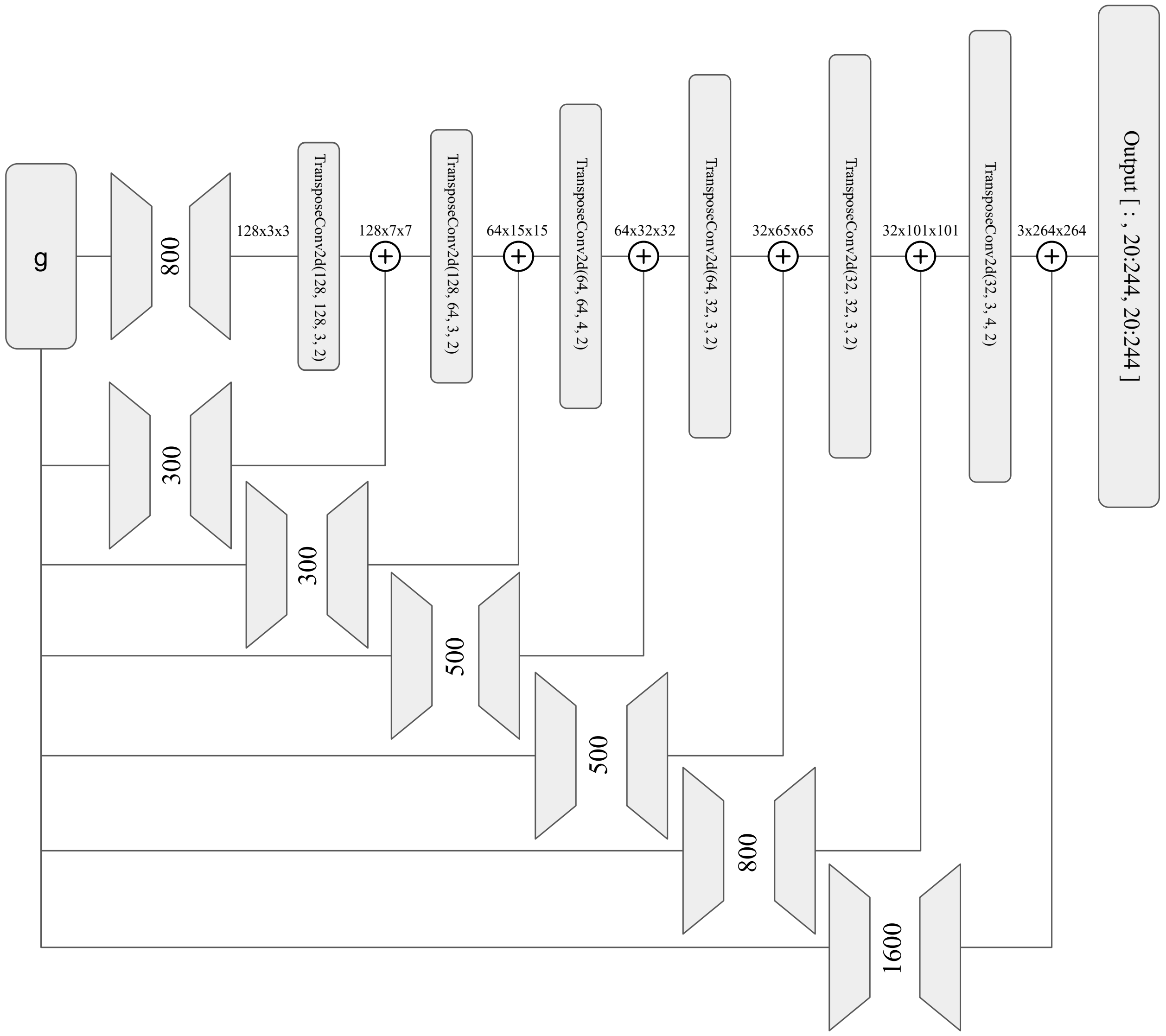
g
800
128x3x3
TransposeConv2d(128, 128, 3, 2)
128x7x7
TransposeConv2d(128, 64, 3, 2)
64x15x15
TransposeConv2d(64, 64, 4, 2)
64x32x32
TransposeConv2d(64, 32, 3, 2)
32x65x65
TransposeConv2d(32, 32, 3, 2)
32x101x101
TransposeConv2d(32, 3, 4, 2)
3x264x264
Output [: , 20:244, 20:244]
300
300
500
500
800
1600

Figure 5. The architecture of our U-Net-based reconstructor r used in our ResImageNet experiments. Here $\boldsymbol{g}$ represents the randomly subsampled model gradient and the output is the resulting reconstructed image.

CIFAR10/100 we first apply random ColorJitter with brightness, contrast, saturation and hue parameters 0.2, 0.1, 0.1, and 0.05, respectively, followed by random horizontal and vertical flips, and random rotation at $N * 90 + \epsilon$ degrees, where N is a random integer and ϵ is chosen uniformly at random on $[-5, 5]$. For ResImageNet, we additionally do a random cropping of the original image to the desired size of 224×224 before the other augmentations.

E.3. Trainset Batch Augmentation

When dealing with multiple clients, as detailed in App. B, we adopt a probabilistic approach. As noted in Sec. 3, the probability of attack success with a perfect threshold τ is $\frac{1}{e}$ in the limit of the number of images being aggregated. As we need SEER to successfully reconstruct images only in the $\frac{1}{e}$ fraction of the securely-aggregated batches, where we are expected to successfully disaggregate, we avoid training on the rest $1 - \frac{1}{e} > \frac{1}{2}$ fraction of the securely-aggregated batches, which can act as a strong noise factor during training and prevent convergence. This in turn results in rejecting training on a big portion of our sampled client batches.

To address this sample inefficiency, we use batch augmentation during training to transform the client batches to ones with a desired brightness distribution. The batch augmentation simply consists of adjusting the brightnesses of individual images within each batch. We do two types of batch augmentations based on two different target distributions—one where it contains precisely one image in the batch with brightness above the threshold τ and another, where pre-

Algorithm 1 The training procedure of SEER

```
1:  function TRAINSEER($f$, $\ell$, $B$, $\mathcal{X}$, $\mathcal{Y}$)
2:      Choose $\mathcal{P}$, initialize $d$ and $r$
3:      while not converged do
4:          $\boldsymbol{X}, \boldsymbol{y} \leftarrow \{\boldsymbol{x}_i, \boldsymbol{y}_i \sim (\mathcal{X}, \mathcal{Y}) \mid i \in [B]\}$
5:          $I_{\text{nul}}, I_{\text{rec}} \leftarrow \mathcal{P}(\boldsymbol{X}, \boldsymbol{y})$
6:          $\boldsymbol{X}_{\text{nul}}, \boldsymbol{y}_{\text{nul}} \leftarrow \boldsymbol{X}[I_{\text{nul}}], \boldsymbol{y}[I_{\text{nul}}]$
7:          $\boldsymbol{X}_{\text{rec}}, \boldsymbol{y}_{\text{rec}} \leftarrow \boldsymbol{X}[I_{\text{rec}}], \boldsymbol{y}[I_{\text{rec}}]$
8:          $\boldsymbol{g}_{\text{nul}}, \boldsymbol{g}_{\text{rec}} \leftarrow \text{BP}(f, \ell, \boldsymbol{X}_{\text{nul}}, \boldsymbol{X}_{\text{rec}}, \boldsymbol{y}_{\text{nul}}, \boldsymbol{y}_{\text{rec}})$
9:          $\mathcal{L}_{\text{nul}} \leftarrow \| d(\boldsymbol{g}_{\text{nul}}) \|_2^2$
10:         $\mathcal{L}_{\text{rec}} \leftarrow \| r(d(\boldsymbol{g}_{\text{rec}})) - \boldsymbol{X}_{\text{rec}} \|_2^2$
11:         $\mathcal{L} \leftarrow \mathcal{L}_{\text{rec}} + \alpha \cdot \mathcal{L}_{\text{nul}}$
12:         $\boldsymbol{\theta}_m \leftarrow \boldsymbol{\theta}_m - \gamma_m \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_m}, \forall m \in \{f, d, r\}$
13:     end while
14:     return $f, d, r$
15: end function
16: function BP($f$, $\ell$, $\boldsymbol{X}_{\text{nul}}$, $\boldsymbol{X}_{\text{rec}}$, $\boldsymbol{y}_{\text{nul}}$, $\boldsymbol{y}_{\text{rec}}$)
17:     $[\![\boldsymbol{l}_{\text{nul}}; \boldsymbol{l}_{\text{rec}}]\!] \leftarrow \ell(f([\![\boldsymbol{X}_{\text{nul}}; \boldsymbol{X}_{\text{rec}}]\!]), [\![\boldsymbol{y}_{\text{nul}}; \boldsymbol{y}_{\text{rec}}]\!])$
18:     return $\frac{\partial \boldsymbol{l}_{\text{nul}}}{\partial \boldsymbol{\theta}_f}, \frac{\partial \boldsymbol{l}_{\text{rec}}}{\partial \boldsymbol{\theta}_f}$
19: end function
```

cisely zero images in the batch have brightness above the threshold τ. We alternate the two augmentations at each step of the training procedure. To achieve the distributions, we adjust the all image brightness within a batch using a heuristic method. The method first adjusts the brightness of the most bright images (least bright images in the case of the dark image property) such that they land on the desired side of the threshold τ. However, as after adjusting the image the brightnesses within the batch are no longer normalized, we then renormalize the batch resulting in a new batch brightness distribution. If the new distribution is as desired we stop, otherwise we iterate until convergence.

F. End-to-end Attack Description & Discussion

Algorithm 1 describes the training of SEER. We train on client-sized batches (see App. H.5 for a related study) sampled from our auxiliary data (Line 4). Based on $\mathcal{P}$ we select the index sets I_{nul} and I_{rec} (Line 5), representing the examples we aim to disaggregate. Then, we simulate the client updates $\boldsymbol{g}_{\text{rec}}$ and $\boldsymbol{g}_{\text{nul}}$ computed on the full batch $\boldsymbol{X}$ (Line 8), and use them to compute our optimization objective (Line 11). We minimize the objective by jointly training f, d, and r using SGD (Line 12).

Mounting SEER once the malicious weights θ_f have been trained using Algorithm 1 is simple, as we illustrate in Algorithm 2. The server, during an FL round, sends the client the malicious model f (Line 2), and receives the gradient update $\boldsymbol{g}$. Then, it applies its secret disaggregator d and reconstructor r (Line 3), to obtain $\boldsymbol{x}_{\text{stolen}}$, the reconstructed private example from the client batch.

Algorithm 2 Mounting SEER

```
1: function MOUNTSEER($f$, $d$, $r$)
2:     $\boldsymbol{g} \leftarrow$GETCLIENTUPDATE($f$)
3:     $\boldsymbol{x}_{\text{stolen}} \leftarrow r(d(\boldsymbol{g}))$
4:     return $\boldsymbol{x}_{\text{stolen}}$
5: end function
```

G. Hyperparameters

In this section, we provide more details about the exact hyperparameters used in our experiments in Sec. 4. We implemented SEER in Pytorch 1.13. Throughout our experiments we used the Adam optimizer with learning rate of 0.0001. To stabilize our training convergence we adopt gradient accumulation and, thus, updated our modules' parameters only once every 10 gradient steps. For CIFAR10/100 we trained between 500 and 1000 epochs, where an epoch is defined to be 1000 sampled batches from our trainset. Having gradient accumulation set to 10, this amounts to 100 gradient descent steps per epoch. For ResImageNet, we trained for 370 epochs instead, with 400 gradient descent steps per epoch.

For faster convergence and better balance in the optimized objective $\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha \cdot \mathcal{L}_{\text{nul}}$ we adopted a schedule for the hyperparameter α, following an exponential curve of the epoch κ. The schedule is defined as: $\alpha(\kappa) = min(|B|, 2^{\beta(\kappa)})$, where $\beta(\kappa) = \frac{(K-\kappa)\beta_0 + \kappa\beta_1}{K}$ linearly interpolates between β_0 and β_1 across the total number of epochs K with (β_0, β_1) set to $(-2, log_2|B|)$. For ResImageNet we set (β_0, β_1) to $(-5, 5.3)$ to allow for better reconstruction earlier in the training process.

H. Additional SEER Experiments

In this section we provide additional SEER experiments, not included in the main body due to space constraints.

H.1. Extended Single-Client Experiments

In Table 2, we present the extended version of our CIFAR10 and CIFAR100 single-client experiments, first presented in Sec. 4. We observe similar trends as in the original experiments. For example, we observe that CIFAR10 and CIFAR100 performances are similar up to $B = 256$ and that there isn't major difference in performance between the most bright and most dark image properties.

H.2. Extended Multi-Client Experiments

In this section, we present and discuss the results of our extended secure aggregation experiments. The results are shown in Table 3 for $C = 4$ and $C = 8$ clients for the *Bright* and *Dark* properties, with batch sizes such that the total number of datapoints is the same as in Table 2.

Table 2. Single-client reconstruction on the *Bright* and *Dark* properties from batches of different sizes B. We report the percentage of well-reconstructed images (*Rec*), the average PSNR and its standard deviation across all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

	CIFAR10, Dark			CIFAR10, Bright			CIFAR100, Dark			CIFAR100, Bright		
B	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑
64	82.0	**32.0 ± 1.6**	**26.6 ± 6.6**	89.4	**32.1 ± 2.0**	27.2 ± 5.3	**97.0**	**32.8 ± 1.3**	**29.3 ± 4.1**	**95.6**	**32.2 ± 1.5**	**28.2 ± 4.3**
128	74.5	29.5 ± 1.4	23.6 ± 6.2	**94.2**	31.9 ± 1.7	**28.2 ± 4.3**	95.9	30.5 ± 1.3	26.8 ± 3.8	94.7	30.0 ± 1.3	26.5 ± 3.7
256	67.6	26.6 ± 1.7	21.6 ± 4.8	81.3	29.0 ± 2.1	24.4 ± 4.9	83.3	25.7 ± 1.3	22.2 ± 3.4	82.2	25.4 ± 1.1	22.1 ± 3.4
512	**82.3**	26.8 ± 1.3	22.8 ± 4.0	87.8	26.6 ± 1.8	23.2 ± 3.5	62.8	24.3 ± 1.5	20.2 ± 4.0	38.4	21.6 ± 1.4	17.4 ± 3.9

We observe similar trends to those observed in our main experiments shown in Sec. 4. In particular, as before, our attack consistently obtains image reconstructions with high average PSNR > 25 on the top 37% of images, i.e., it recovers all images, that can be recovered due to the probabilistic nature of our property $\mathcal{P}$ in this setting, almost perfectly. Compared to Table 2, the attack probability of reconstruction degrades with C, confirming our intuition (see App. B) that secure aggregation provides additional protection in the presence of BN, compared to simply using large batches. Despite this, we observe that the obtained success probability is still substantially higher than 37%. We suspect this is due to the model learning a restricted version of our local property reconstruction for each individual client. We provide further comparison of our secure aggregation and single-batch variants of SEER in App. H.3.

Even more surprisingly, we notice that both reconstruction quality and the percentage of batches reconstructed rise with the number of aggregated points. Experimentally, for smaller batch sizes we observe that this is caused by sampling too many images above the threshold τ, defining $\mathcal{P}$, and thus failing to disaggregate as often. Therefore, we believe that the observed improvements for large number of aggregated points is caused by better estimation of τ on the additional samples contained in larger batches.

H.3. Comparison between Properties $\mathcal{P}$ based on Global and Local Ditributions

In this section, we compare our two SEER variants, based on local and global properties $\mathcal{P}$ respectively. We mount both variants of SEER on gradients coming from a single client with batch size $B = 128$ on CIFAR10. We note that both methods are well defined in this setting and either one can successfully reconstruct data from the client batches, however the global one is probabilistic in nature.

The results are depicted in Table 4. While both methods successfully reconstruct the majority of client batches, we clearly see the benefits of using the local property $\mathcal{P}$. In particular, the results in Table 4 suggest that the local distribution approach reconstructs up to 1.75 times more images, while also producing higher PSNR values not only on the full set of reconstructed batches but also on the top 37% of them. This motivates the need of our single-client attack, that demonstrates that secure aggregation provides additional protection to individual clients.

H.4. Additional Property $\mathcal{P}$

This section, demonstrates that we are not restricted to brightness properties only. We experiment with a property that is invariant to the mean of a datapoint's tensor representation. The property is computed through a linear combination with coefficients $(2, -1, -1)$ along the RGB color-channel dimension and summing the other dimensions. The higher the property response computed on an image, the redder the image appears to human perception. Experiments are conducted in a single client setting with varying batch sizes: $B \in \{64, 128, 256, 512\}$ and reconstruction quality is evaluated using the PSNR metric, as outlined in Table 5. Comparative analysis of the experimental results against other properties - bright and dark - reveals that the "redness" property performs as well as the alternative properties. This suggests that more sophisticated properties, such as arbitrary linear combinations on the entire tensor representation of an image may yield satisfactory outcomes. This would potentially enable an attacker to query and target specific types of images for stealing by maximizing their response to attacker-defined kernels.

H.5. Robustness to B

In this section, we demonstrate that attack parameters θ_f generated by SEER for a particular client batch size B can work to a large extend for batch sizes close to the original one, thus relaxing the requirement that the exact client batch size B is known during the crafting of the malicious model f. In particular, in Table 6 we show the effect of applying our single-client attack trained on $B = 128$ on CIFAR10 using the *Bright* image property for clients with varying batch sizes B_{test}. We observe that while, as expected, SEER performs best when $B_{\text{test}} = B$, both the success rate and the quality of reconstruction on clients with batch sizes even $2\times$ larger than the trained one remain very good. We note that Table 6 suggests that underestimating the client batch size

Table 3. Multi-client reconstructions on the *Bright* and *Dark* properties using different number of clients C on CIFAR10, for different total numbers of aggregated points. We report the percentage of correctly reconstructed images (*Rec*) and the average PSNR and its standard deviation across all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

	$C = 4$, Dark			$C = 4$, Bright			$C = 8$, Dark			$C = 8$, Bright		
#Pts	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑
64	50.2	27.9 ± 3.0	20.2 ± 6.8	41.4	27.2 ± 3.7	19.3 ± 6.8	43.0	26.5 ± 3.7	19.0 ± 6.7	41.3	$\mathbf{27.2 \pm 3.4}$	19.5 ± 6.5
128	51.3	28.4 ± 3.0	20.7 ± 6.7	44.2	26.5 ± 3.1	19.4 ± 6.2	43.4	27.2 ± 3.8	19.1 ± 7.2	40.6	26.6 ± 3.7	19.2 ± 6.5
256	50.9	29.8 ± 2.4	21.2 ± 7.4	51.9	$\mathbf{27.3 \pm 2.6}$	$\mathbf{20.3 \pm 6.2}$	51.7	27.3 ± 2.9	20.5 ± 6.0	41.9	25.5 ± 3.2	18.6 ± 6.1
512	**61.3**	$\mathbf{29.9 \pm 2.6}$	$\mathbf{21.8 \pm 7.2}$	**52.9**	25.7 ± 2.2	20.0 ± 5.2	**56.3**	$\mathbf{29.0 \pm 2.8}$	$\mathbf{21.4 \pm 6.8}$	**51.7**	26.2 ± 3.2	$\mathbf{20.2 \pm 5.5}$

Table 4. Single-client reconstruction on the *Bright* and *Dark* properties from batches of size $B = 128$ on CIFAR10 using our Single-client variant (*Local*) and our Secure Aggregation variant with $C = 1$ (*Global*). We report the percentage of well-reconstructed images (*Rec*), the average PSNR and its standard deviation across all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

	CIFAR10, Bright			CIFAR10, Dark		
$\mathcal{P}$	Rec (%)	PSNR Top ↑	PSNR All ↑	Rec (%)	PSNR Top ↑	PSNR All ↑
Global	54.4	27.0 ± 1.8	20.6 ± 5.8	61.9	27.7 ± 2.2	21.1 ± 6.1
Local	**94.2**	$\mathbf{31.9 \pm 1.7}$	$\mathbf{28.2 \pm 4.3}$	**74.5**	$\mathbf{29.5 \pm 1.4}$	$\mathbf{23.6 \pm 6.2}$

B_{test} during the training of f is better than overestimating it, as the reconstruction performance on $B_{\text{test}} = 256$ is significantly better than on $B_{\text{test}} = 64$. This is mostly caused by d filtering out all images in client batches smaller than the batches used at train-time, resulting in the removal of all the client data.

Table 5. Single-client reconstruction of red property samples on CIFAR10 on networks trained with different batch sizes and tested on the corresponding batch size. We report the percentage of well-reconstructed images (*Rec*), the average PSNR and its standard deviation on all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

B	Rec (%)	PSNR Top ↑	PSNR All ↑
64	87.3	30.4 ± 1.1	26.5 ± 5.2
128	93.5	31.1 ± 1.2	27.8 ± 4.1
256	**94.7**	$\mathbf{31.3 \pm 1.0}$	$\mathbf{28.0 \pm 4.0}$
512	94.4	30.0 ± 1.2	26.6 ± 3.8

Table 6. Single-client reconstruction on the bright property on CIFAR10 on a network trained with batch size $B = 128$ and tested for various client batch sizes B_{test}. We report the percentage of well-reconstructed images (*Rec*), the average PSNR and its standard deviation on all reconstructions (*PSNR All*), and across the top 37% images (*PSNR Top*).

B_{test}	Rec (%)	PSNR Top ↑	PSNR All ↑
64	42.0	21.5 ± 1.8	18.5 ± 2.9
96	87.4	30.6 ± 1.3	26.3 ± 4.9
128	**94.2**	$\mathbf{31.9 \pm 1.7}$	$\mathbf{28.2 \pm 4.3}$
192	86.3	30.8 ± 2.4	25.8 ± 5.1
256	67.5	28.0 ± 2.8	22.4 ± 5.1