SRATTA: Sample Re-ATTribution Attack of Secure Aggregation in Federated Learning.

Tanguy Marchand [1] **Regis Loeb**[*1] **Ulysse Marteau-Ferey**[*1] **Jean Ogier du Terrail**[*1] **Arthur Pignet**[*1]

Abstract

We consider a cross-silo federated learning (FL) setting where a machine learning model with a fully connected first layer is trained between different clients and a central server using FedAvg, and where the aggregation step can be performed with secure aggregation (SA). We present SRATTA an attack relying only on aggregated models which, under realistic assumptions, (i) recovers data samples from the different clients, and (ii) groups data samples coming from the same client together. While sample recovery has already been explored in an FL setting, the ability to group samples per client, despite the use of SA, is novel. This poses a significant unforeseen security threat to FL and effectively breaks SA. We show that SRATTA is both theoretically grounded and can be used in practice on realistic models and datasets. We also propose counter-measures, and claim that clients should play an active role to guarantee their privacy during training.

1. Introduction and Background

Federated learning (FL) (Shokri & Shmatikov, 2015; McMahan et al., 2017) has been introduced as a *privacy-preserving* technique to train a model on multiple data sources or clients. The applications of FL now span multiple domains from medicine (Ogier du Terrail et al., 2023) to finance (Long et al., 2020), giving ways to unlock new sources of data while remaining privacy-preserving (Zheng et al., 2022). As the adoption of this new technology grows, so do concerns about the limitations of FL regarding its actual privacy guarantees (Kairouz et al., 2021). Indeed, it has now become clear that naïve gradient sharing is vulnerable to data-reconstruction (Zhu et al., 2019), membership (Shokri et al., 2017), and property inference (Ganju et al., 2018) attacks, which endangers privacy.

Various mechanisms to increase the privacy provided by FL have been explored by the literature, in particular differential privacy (DP) and secure aggregation (SA) protocols. Despite the absolute security guarantees they provide (Dwork et al., 2014), current DP algorithms might impact the models' performances (El Ouadrhiri & Abdelhadi, 2022). On the other hand, SA is a cryptographic protocol that *"allow[s] a collection of mutually distrust parties, each holding a private value, to collaboratively compute the sum of those values without revealing the values themselves"* (Bonawitz et al., 2017). It can therefore be used to average model updates during FL training, to hide individual contributions from clients without impacting model performance. Such techniques are being used in production (Heyndrickx et al., 2022) as an effective way to protect user data. By hiding individual contributions, the effect of SA is twofold: (i) it virtually increases the number of accumulated gradients in one update, making gradient and update-based attacks more difficult and (ii) it prevents reconstructed samples to be attributed to specific clients.

Recently, the efficiency of SA to prevent reconstruction attacks has been questioned, as gradient attacks (Zhu et al., 2019) can recover samples from large batches of raw gradients (Yin et al., 2021). In the FL setting, recent attacks (Geiping et al., 2020; Xu et al., 2022; Dimitrov et al., 2022) have built on these works and manage to partially reconstruct data from FL updates in specific cases. Despite these recent works in gradient (or FL-update) attacks targeting the SA setting, the claim that, even though individual samples might be recovered, SA prevents linking those samples back to their respective clients has remained unchallenged so far.

In this work we demonstrate for the first time that the nature of FL updates does allow one to link samples, despite the use of SA, assuming the model has a fully connected first layer and that we are in the *cross-silo* FL setting (number of clients from 2 to 50). Targeting this newly found weakness, we devise an attack, named SRATTA (Sample Re-ATTribution Attack against Secure Aggregation in Federated learning), to group samples belonging to individual clients together, recovering per-client contributions up to a permutation. This is done in the *honest but curious* setting,

[*]Alphabetical order [1]Owkin Inc.. Correspondence to: Tanguy Marchand <tanguy.marchand@owkin.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

where the clients and server follow the protocol. This new breach of privacy could have important consequences in multiple real-world use-cases, such as healthcare applications, where the number of clients is typically limited. For instance, the prior knowledge that one specific sample belongs to one specific client, could be propagated to other recovered samples. In the worst-case scenario, this would lead to a one to one assignment of all samples of an individual client exposing the entirety of its dataset. This grouping threat on top of sample recovery should be addressed, as it might create other important privacy-leakages. Indeed, the knowledge that specific data are being collected by a specific client could compromise its area of research or target market. We also believe that this threat advocates for a paradigm shift in the design of privacy preserving bricks in (cross-silo) FL: clients must take a more active part in defending themselves, rather than only relying on a secure central server.

Contributions. After presenting the attack environment and assumptions we make on the model and data in Section 2, we describe a theoretically justified attack which allows to recover and group samples from the aggregated updates. More specifically, we make the following contributions.

- A new analytical data-reconstruction attack on FL updates, which can target any type of loss function, using a data prior that can easily be constructed for many data modalities (Section 3.2).
- A method to group recovered samples belonging to the same client together in spite of the use of SA, unveiling a new target for attacks on cross-silo FL trainings with SA (Section 3.2).
- Evidence that SRATTA poses a real threat on several benchmark datasets and machine learning tasks (Section 4).
- Defensive schemes against SRATTA, highlighting the role clients can play to guarantee their own privacy (Section 5).

We also provide a Python implementation of SRATTA, and of the proposed defensive schemes.

2. Attack environment and assumptions

This section presents the setting and main assumptions, and discusses them in light of the current literature. Section 2.1 formally introduces (i) the algorithm that SRATTA targets and (ii) the quantities and information the attacker needs to access in order to perform the attack. Section 2.2 states, motivates and discusses the additional assumptions that we make, in particular concerning the machine learning (ML) models which SRATTA targets. Finally, in Section 2.3, we further discuss the works related to this threat model and our assumptions, both in and out the FL framework.

2.1. Attack Environment

Cross-silo federated learning setting We consider a cross-silo FL setting (Kairouz et al., 2021), where a central server trains an ML model on data distributed across K clients identified by $k \in \{1, \dots, K\}$. Each client k has a dataset $\mathcal{D}_k$ of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where x is the data sample, y is the associated ground-truth label—where the term label denotes any (un)supervised learning task. We denote with $\mathcal{D}$ the complete "pooled" dataset $\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k$.

The machine learning model m is a function parametrized by $\theta \in \mathbb{R}^p$ such that $\widetilde{y} \stackrel{\text{def}}{=} m_\theta(x) \in \mathbb{R}^C$ is the prediction for sample $x \in \mathcal{X}$ by the model m_θ. We use a loss function $\ell(\widetilde{y}, y)$ and assume that $\ell : \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}$ is differentiable in its first variable. The training optimization procedure aims at minimizing the average loss of the model predictions over all data samples from all clients, that is

$$\mathcal{L}(\theta; \mathcal{D}) = \sum_{(x,y)\in\mathcal{D}} \ell(m_\theta(x), y) = \sum_{k=1}^{K} \mathcal{L}(\theta; \mathcal{D}_k), \qquad (1)$$

where $\mathcal{L}(\theta; D) \stackrel{\text{def}}{=} \sum_{(x,y)\in D} \ell(m_\theta(x), y)$ for any set D of sample/label pairs. We make the following assumption, which is standard in cross-silo FL (Kairouz et al., 2021).

Assumption 2.1 (cross-silo federated averaging)**.** The loss in Equation (1) is optimized using the original FedAvg algorithm (McMahan et al., 2017) with local minibatch SGD (Bottou, 2012), *with all clients participating at each round*, and using SA at the averaging step. This algorithm is described in Algorithms 1 and 2.

Algorithm 1 FedAvg

Require: Initialization θ_0
  for $t = 1$ **to** $t_{\max}$ **do**
    send model θ_{t-1} to each server
    for $k = 1$ **to** K **in parallel do**
      $\theta_{t,k} = \text{LocalUpdate}^{(k)}(\theta_{t-1})$
    end for
    $\theta_t = \frac{1}{K} \sum \theta_{t,k}$ // Done using Secure Aggregation
  end for
Ensure: $\theta_{t_{\max}}$

Algorithm 2 LocalUpdate // Executed on server k

Require: initial model θ_{t-1}, local dataset $\mathcal{D}_k$, batch size b, learning rate η
  $\theta_{t,k,i=0} \leftarrow \theta_{t-1}$
  for $i = 0$ **to** $n_{\text{updates}} - 1$ **do**
    $B_{t,k,i} \leftarrow$ batch of size b from $\mathcal{D}_k$
    $\theta_{t,k,i+1} \leftarrow \theta_{t,k,i} - \eta \nabla_\theta \mathcal{L}(\theta_{t,k,i}, B_{t,k,i})$
  end for
Ensure: $\theta_{t,k,n_{\text{updates}}}$

We refer the reader to the notation introduced in the pseudo-code above throughout our demonstration. Regarding the aggregation done with SA, in some cases, the server has access to the aggregated parameters of the model θ_t while in other implementations of SA (Cramer et al., 2015), the central server either does not exist or cannot know the value of θ_t. Following Pasquini et al. (2022), we do not make any assumption on the actual implementation used to perform SA as long as it works as intended. The shorthand SA thus designates an ideal invocation of such a protocol.

Finally, note that Assumption 2.1 can be relaxed without impacting the performances of SRATTA (see Appendix C).

Threat model

Assumption 2.2 (Threat model)**.** We assume a *honest but curious* threat model where each participant follows the protocol but tries to infer as much information as possible on the data of other participants *from the quantities they receive*. We further assume that no client collusion is possible (two clients cannot agree to perform an attack together), therefore placing ourselves in a particularly unfavorable setting for the attacker. The attacker described in this work is anyone who knows the target ML task, which only entails knowing the data type of each feature and whether additional preprocessing is used. In addition, the attacker has access to the aggregated model after SA at each round: θ_t for $t \in 0, \dots, t_{\max}$. If multiple trainings are performed (when testing different seeds or learning rates), the attacker has access to all the aggregated iterates from all training phases.

The attacker does **not** need to have access to the client-specific models. The attacker can be either the server if the cryptographic protocols used do not prevent it, or any of the clients.

2.2. Additional assumptions

Assumption on the model We assume that data samples are vectors of dimension d ($\mathcal{X} = \mathbb{R}^d$). The main limiting assumption of our work is that the first layer of the model is fully connected with H hidden neurons with bias.

Assumption 2.3 (Fully connected first layer with bias)**.** The model parameters can be decomposed as $\theta = (W, b, \phi)$ where $W \in \mathbb{R}^{H \times d}$, $b \in \mathbb{R}^H$ and $\phi \in \mathbb{R}^r$. The model is of the form

$$m_\theta(x) = f_\phi(\mathrm{ReLU}(Wx + b)), \;\; \theta \in \mathbb{R}^p, \; x \in \mathbb{R}^d, \quad (2)$$

where $f_\phi(z)$ is itself a (sub) differentiable model in $z \in \mathbb{R}^H$. $W \in \mathbb{R}^{H \times d}$ is referred to as the weights, and $b \in \mathbb{R}^H$ is referred to as the bias. The couple $\overline{W} = (W, b) \in \mathbb{R}^{H \times d} \times \mathbb{R}^H$ is referred to as the extended weights.

We also denote by $W^h \in \mathbb{R}^d$ the h-th line of W, with $b^h \in \mathbb{R}$ the bias of the h-th hidden neuron, and with $\overline{W}^h = (W^h, b^h) \in \mathbb{R}^{d+1}$ the extended weights of the h-th hidden neuron. Finally, for any iterate θ_s described in the FedAvg algorithm, we denote by $W_s, b_s, \overline{W}_s^h, W_s^h, b_s^h, \overline{W}_s$ the corresponding subset of parameters.

Assumption 2.3 is a strong assumption. It is satisfied by multi-layer perceptron (MLP) architectures, which have been scaled with success to a variety of problems such as for tabular data (Kadra et al., 2021) or recently for computer vision (Tolstikhin et al., 2021). These architectures are therefore the main targets of SRATTA.

Finally, we make the following assumption.

Assumption 2.4. Samples of $\mathcal{D}$ are unique.

In the cross-silo FL applications we consider (notably associated with FL for healthcare), we believe this assumption to be satisfied in a vast majority of cases. Note however that in cross-device FL settings, for example in NLP applications, the repeated presence of certain relatively short sentences could prevent Assumption 2.4 from holding.

In order to decide whether an element $x \in \mathbb{R}^d$ is a true data sample from one of the clients, we use a *data prior*. By *data prior*, we mean a subset $\mathcal{P} \subset \mathcal{X}$ which contains the true data samples from the dataset $\mathcal{D}$. This data prior will be used in the first step of SRATTA (see Section 3.2) to identify true data samples from a list of candidates defined in that step.[1] It will therefore have to be small enough to filter out all false candidates. Our experiments will confirm that this requirement is satisfied empirically. Indeed, while the existence of a such a data prior may seem a strong hypothesis, it can be easily built using knowledge of general data properties, which are available to the attacker in our threat model 2.2. This is the case in the following examples.

Example 1. If the possible values taken by certain features $F \subset \{1, ..., d\}$ of the data samples x take values in a known finite set $\mathcal{V}$, we can take the data prior $\mathcal{P} = \{x \in \mathbb{R}^d \; : \; (x_f)_{f \in F} \in \mathcal{V}\}$. This is the case with binary features where $\mathcal{V} = \{0, 1\}^F$, for most image datasets where $\mathcal{V} = \{k/255, k = 0, \dots, 255\}^{d}$[2], and for any multimodal dataset where **a single modality**, corresponding to the features F, belongs to a known discrete data prior (we illustrate this in Section 4).

Example 2. If the dataset is normalized, we can use the data prior $\mathcal{P} = \{||x|| = 1\}$.

[1] If we do not have such a data prior, we can do as done by Pan et al. (2022), and identify data samples as duplicates.

[2] If we further perform standardization of the image such as scaling, the data prior still works but its values will be shifted.

2.3. Related work

Our work uses recent results on analytical data reconstruction attacks, in order to take advantage of FL updates to disaggregate SA.

Data reconstruction attacks

Optimization-based attacks on raw gradients. Zhu et al. (2019) show that one can recover samples from a training dataset using the gradients generated during training in the case of classification tasks with cross-entropy loss and small batches. Their approach consists in generating "dummy" samples as close as possible to the real samples by matching the gradients obtained when training using the "dummy" samples with the ones observed during the real training. While this work is not specific to FL, such attacks are particularly important in FL as the gradients -or the model updates- are shared and sent over the network.

Multiple works improve the accuracy of such methods using different image priors (Zhao et al., 2020; Geiping et al., 2020; Yin et al., 2021) and scaling the attack to handle medium to large batches of gradients (Yin et al., 2021). Although the grouping of samples in SRATTA is optimization-based like the attacks above, our data recovery step is purely analytical. Moreover, while previous attacks are limited to the use of cross-entropy loss over images, SRATTA can target different dataset types and any differentiable loss.

Analytical attacks on raw gradients. Aono et al. (2017); Geiping et al. (2020); Pan et al. (2022) point out that, in the case of a fully-connected layer, if only one sample activates a hidden neuron, the gradients of the weights corresponding to that hidden layer are proportional to the input of the layer, and the proportionality coefficients correspond to the gradient of the bias irrespective of the loss used (cf Section 3.2). Zhu & Blaschko introduce an analytical reconstruction attack which recursively reconstructs activation maps layer per layer. We make use of such results in SRATTA. However (i) we scale such attacks from gradients to multiple FL updates, (ii) our method introduce the novel use of a data prior to allow an analytical reconstruction, and (iii) the core of SRATTA is grouping samples from the same clients together, which is a new idea.

Attacks on FL updates. Geiping et al. (2020) and Xu et al. (2022) show qualitatively that, if very small learning rates are used (or *single-batch approximation*), original optimization based attacks still work to some extent. A related approach introduced by Dimitrov et al. (2022) leverages higher-order automatic differentiation to fully simulate multiple gradient updates by optimizing over multiple dummy batches simultaneously. While the two previous attack rely on optimizing a gradient matching problem, a third line of work introduced by Kariyappa et al. (2022) optimizes an independent component analysis problem to recover individual gradient, before applying gradient inversion attacks to recover samples.

All attacks presented in this paragraph are optimization-based gradient attacks and thus suffer from the same drawbacks: they can only target image classification tasks (when using gradient matching) and more importantly do not target samples grouping at all. In addition, while all data reconstruction attacks including ours are sensitive to the number of updates, the learning-rate and the batch-size, we show in Appendix E.4 that SRATTA can accomodate multiple updates (up to 20 updates) and large batch-sizes (up to 32) while still maintaining reasonable accuracy ($> 10\%$), which is rare in this literature.

Leveraging multiple rounds and tranings. Both Xu et al. (2022) and Dimitrov et al. (2022) exploit the redundancy of samples between each round to increase the effectiveness of the attack. SRATTA also exploits this redundancy, in a novel way, when performing the grouping step (cf Section 3.2).

From MLPs to convolutional neural networks (CNN). Many of the works cited in this section, such as those by Pan et al. (2022); Zhu & Blaschko; Boenisch et al. (2021); Kariyappa et al. (2022), extend their gradient-based recovery strategy from MLPs, which satisfy Assumption 2.3, to CNNs which do not satisfy this assumption anymore, by inverting the convolutional part of the network. While we believe the techniques used in these works could also be applied in our setting, our current work relies heavily on Assumption 2.3, and is therefore not directly applicable to break SA for CNNs.

Disaggregating Secure Aggregation. Different lines of research tackle the problem of retrieving client-specific information from aggregated updates. However, they differ from our work: they either assume a malicious threat model, or do not group recovered samples together. Pasquini et al. (2022) consider a threat model where the central server controls the updates sent to each client, exposing the data despite the use of SA, which is not the case in standard cross-silo FL training. In this attack, the malicious aggregator sends different models to each client. All but one clients receive a model that produces null gradients while one of the servers, the target, receives an unaltered model. Therefore, after SA, the aggregator can obtain the gradient from a single client. Within our threat model defined in Section 2, such an attack is not possible as we assume that the central server is honest-but-curious and that the protocol prevents the aggregator from controlling the models sent to each client. Similarly, Zhao et al. (2023); Fowl et al. (2022) retrieve client data, but assuming a malicious threat model, in which the aggregator can choose the model it sends to each client. Lam et al. (2021) assume that the model updates sent by each client are constant, or almost constant

across the rounds, and that the participation differs between each round. They recover therefore both the participation matrix and the model updates of each client. Within our threat model and framework, defined in Section 2, such an approach does not hold as: (i) the matrix participation is constant, (ii) only a small fraction of the local dataset might be used at each step, therefore there is no reason for the model updates to be constant across the rounds for a given client. Finally, Pejó et al. (2022) considers a framework similar to ours, but recover only a high level description of the data quality of each client, and not the content of their dataset.

In view of these different works, we believe there has been no other related attempts to perform sample re-attribution despite the use of SA under similar assumptions.

3. Presentation of SRATTA

In this section, we present SRATTA, our method to recover and group client samples together. The full SRATTA algorithm can be found in Algorithm 3.

3.1. Preliminary: activation sets

In this part, we introduce the central tool of our attack: activation sets. Intuitively, activation sets consist in samples which contribute linearly to a given weight update. We target these activation sets and partially recover them in our attack. The rest of the subsection provides formal definitions of batch and round-level activation sets, and states the associated linear decompositions.

Batch-level activation sets Given a parameter θ, a batch B and a neuron h, the batch-level activation set of the neuron h for the batch B is defined as

$$\mathcal{A}^h(\theta, B) \overset{\text{def}}{=} \{x \in B \ : W^h \cdot x + b^h > 0 \text{ and } \frac{\partial \mathcal{L}(\theta; x, y)}{\partial z^h} \neq 0\}, \tag{3}$$

where z^h is the output of the first activation corresponding to neuron h, and $\frac{\partial \mathcal{L}(\theta;x,y)}{\partial z^h}$ is the differential of the loss w.r.t. z^h at parameter θ and data point (x, y) (cf Equation (12)). This set characterizes the samples which contribute *linearly* to the gradient of the loss $\mathcal{L}$, as shown in the following proposition.

Proposition 3.1. *Fix a neuron h, a model parameter θ, and a batch B. Denoting $(x_1, ..., x_N)$ the complete list of samples in $\mathcal{A}^h(\theta, B)$, there exists $(\lambda_i) \in \mathbb{R}\backslash\{0\}^N$ such that*

$$\nabla_{W^h} \mathcal{L}(\theta, B) = \sum_{i=1}^{N} \lambda_i x_i, \qquad \nabla_{b^h} \mathcal{L}(\theta, B) = \sum_{i=1}^{N} \lambda_i. \tag{4}$$

This result, proved in Appendix A, is an extension of Proposition D.1 by Geiping et al. (2020).

Round-level activation sets Since the attacker only has access to round-level updates $\Delta\overline{W}_t^h \overset{\text{def}}{=} \overline{W}_t^h - \overline{W}_{t-1}^h$ and not to individual batch gradients, we define the round-level activation set of neuron h during round t as the union of all batch-level activation sets encountered by all clients during local training.

$$\mathcal{A}_t^h = \bigcup_{k=1}^{K} \bigcup_{i=0}^{n_{\text{updates}}-1} \mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i}). \tag{5}$$

The batch-level linear decomposition property of Proposition 3.1 is preserved at round-level: the round-level update can be decomposed on the round-level activation set.

Proposition 3.2. *Fix a neuron h, and $t \in \{1, ..., t_{\max}\}$. Denoting $(x_1, ..., x_N)$ the complete list of samples in $\mathcal{A}_t^h$, there exists $(\lambda_i) \in \mathbb{R}\backslash\{0\}^N$ such that*

$$\Delta W_t^h = \sum_{i=1}^{N} \lambda_i x_i, \qquad \Delta b_t^h = \sum_{i=1}^{N} \lambda_i. \tag{6}$$

This proposition results from the nature of the FedAvg updates and is proved in Appendix A.2. It is the cornerstone of the first two steps of SRATTA.

3.2. Main steps of SRATTA

SRATTA consists of three main steps: (i) recovering samples using the data prior, (ii) recovering small activation sets, and (iii) grouping samples coming from the same client together. We emphasize that (iii) is completely novel, and highlights an important finding of the present paper: using SA does not prevent attackers from recovering groups of samples belonging to the same client.

Step 1. Sample recovery

The first step of SRATTA consists in recovering individual data samples. It is based on the observation that certain samples can be recovered as a ratio of round-level updates, which is a direct consequence of Proposition 3.2.

Proposition 3.3. *For any neuron h and round t, if the round-level activation set $\mathcal{A}_t^h$ contains a single sample x, then $x = \Delta W_t^h / \Delta b_t^h$. We say that such a sample x is an "isolated sample".*

To recover all isolated samples, we start by constructing the family of ratios $(r_t^h \overset{\text{def}}{=} \Delta W_t^h / \Delta b_t^h)_{t,h}$ from the round-level updates, which are accessible to the attacker under Assumption 2.2 (skipping (t, h) for which $\Delta b_t^h = 0$). We then filter

out elements by only keeping those which belong to the data prior $\mathcal{P}$, defined in Section 2.2. In SRATTA, the set of recovered data samples $\mathcal{R}$ is therefore defined as

$$\mathcal{R} \overset{\text{def}}{=} \left\{ r_t^h \in \mathcal{P} \ : \ \Delta b_t^h \neq 0 \right\}. \tag{7}$$

While we are sure that $\mathcal{R}$ contains all isolated samples by Proposition 3.3, $\mathcal{R}$ may still contain "false" data samples. However, this has never happened in our experiments, and we recover a reasonable proportion of the client datasets (see Section 4). We explain this by the combination of two factors. First, our data priors are discrete, hence "small" in $\mathbb{R}^d$. Second, if $\mathcal{A}_t^h$ is of size greater than two, the corresponding ratio r_t^h is a sum of *multiple data samples*, weighted by coefficients whose structure has no link with the data prior (see Proposition 3.2).

Remark 3.4 (Analytical recovery of the samples). A recovered sample as defined in Equation (7) is analytically recovered: samples are reconstructed up to machine precision. In our experiments, it will therefore be irrelevant to measure the PSNR, the LPIPS or any similarity metric between the reconstruction and the original sample.

Step 2. Reconstructing small activation sets

The second step of SRATTA consists in reconstructing certain round-level activation sets; this will be needed to group samples together in step 3.

To reconstruct an activation set $\mathcal{A}_t^h$, we reverse engineer Proposition 3.2, and look for a linear decomposition of the round-level updates $\Delta\overline{W}_t^h$ in the form of Equation (6), where the samples are taken in $\mathcal{R}$. More formally, for all (t, h), we look for $N \in \mathbb{N}$, $(\lambda_r)_{1 \leqslant r \leqslant N} \in \mathbb{R}^d \backslash \{0\}$ and $(x_r)_{1 \leqslant r \leqslant N} \in \mathcal{R}^N$ such that

$$\Delta W_t^h = \sum_{r=1}^{N} \lambda_r x_r \tag{8a}$$

$$\Delta b_t^h = \sum_{r=1}^{N} \lambda_r. \tag{8b}$$

In general, there is no unique solution to this problem: (i) it could have no solution if the round-level activation set $\mathcal{A}_t^h$ is not included in $\mathcal{R}$, or (ii) it could have multiple solutions if the elements of either $\mathcal{A}_t^h$ or $\mathcal{R}$ are not linearly independent. The first issue is unavoidable as it depends on the quality of the recovery step, but would simply result in not recovering the specific $\mathcal{A}_t^h$. The second issue is more problematic as it could lead to "wrong" recovered activation sets. In order to reduce the risk of linear dependencies, SRATTA only attempts to recover activation sets of size smaller than a fixed threshold $N_{\text{max}} \ll d$. In our applications, we set $N_{\text{max}} = 20$, the smallest dataset dimension being $d = 180$.

Together with the constraint that $N \leqslant N_{\text{max}}$, Equation (8) becomes a sparse reconstruction problem (Zhang et al. (2015), see Appendix B for further details). Many algorithms exist to tackle this problem, such as orthogonal matching pursuit (OMP) by Mallat & Zhang (1993), or the LASSO (Kim et al., 2007). While the assumptions to guarantee the convergence of these algorithms to a solution cannot be checked by the attacker, we found that using OMP works well in practice. At the end of this step, SRATTA recovers a set $\widetilde{\mathcal{A}}_t^h$ of samples in $\mathcal{R}$, for all pairs $(t, h) \in L_{\text{rec}}$ whose associated problem 8 is solved by the OMP. While the recovered $\widetilde{\mathcal{A}}_t^h$ are not formally guaranteed to be equal to $\mathcal{A}_t^h$, we will assume they are to keep notations simple. This step is the most computationally expensive of SRATTA.

Step 3a). Theoretical foundation of sample grouping

The third step of SRATTA consists in grouping samples which belong to the same client dataset together, and is the key contribution of the paper.

We start by providing the intuition and theoretical results which allow for this grouping of samples, before effectively using them in step 3b). For any t, h, we introduce the set

$$\widetilde{\mathcal{A}}_{t,0}^h \overset{\text{def}}{=} \{x \in \mathcal{A}_t^h \ : \ W_{t-1}^h x + b_{t-1}^h > 0\}, \tag{9}$$

which is computable by SRATTA if $(t, h) \in L_{\text{rec}}$. Intuitively, the set $\widetilde{\mathcal{A}}_{t,0}^h$ contains all samples which activate neuron h at the beginning of round t. These samples are *the only ones* which make the extended weights of the neuron h move away from their initial value. Thus, if a given client k has a sample which activates neuron h during round t, its *first* sample to activate neuron h necessarily belongs to $\widetilde{\mathcal{A}}_{t,0}^h$. This reasoning yields the following central result (see Appendix A.3 for a proof).

Theorem 3.5. *Under Assumption 2.4, if a sample from client k is in $\mathcal{A}_t^h$, then at least one sample from client k is in $\widetilde{\mathcal{A}}_{t,0}^h$.*

Theorem 3.5 can be exploited by attackers to group samples together; while SRATTA exploits it by using Corollary 3.6, there could be more elaborate ways of using this result.

Corollary 3.6. *Under Assumption 2.4:*

*(i) if there is only **one sample** in $\widetilde{\mathcal{A}}_{t,0}^h$, all samples in $\mathcal{A}_t^h$ belong to the same client.*

(ii) if all the samples in $\widetilde{\mathcal{A}}_{t,0}^h$ belong to the same client, all samples in $\mathcal{A}_t^h$ belong to that same client.

Step 3b). Effectively matching samples in SRATTA

In this last step, SRATTA leverages Corollary 3.6 to iteratively build a graph on the recovered samples $\mathcal{R}$, where edges are drawn between samples if they belong to the same client. As a result, all samples in a connected component of the graph come from the same client. Recall that at the end of step 2, we assume that SRATTA has reconstructed $\mathcal{A}_t^h$

and hence $\widetilde{\mathcal{A}}^h_{t,0}$ for all $(t,h) \in L_{\text{rec}}$. Initially, there are no edges in the graph.

SRATTA starts by using (i) of corollary 3.6: for all $(t,h) \in L_{\text{rec}}$ such that $\widetilde{\mathcal{A}}^h_{t,0}$ is of size 1, an edge is drawn between all samples in $\mathcal{A}^h_t$, as they belong to the same client.

SRATTA then iteratively adds edges by using (ii) of corollary 3.6. For all $(t,h) \in L_{\text{rec}}$, we check if $\widetilde{\mathcal{A}}^h_{t,0}$ is included in a connected component of the graph. If this is the case, all samples in $\widetilde{\mathcal{A}}^h_{t,0}$ belong to the same client: an edge is drawn between all samples in $\mathcal{A}^h_t$.

This procedure is stopped when no new edge is added. SRATTA finally returns the connected components of the graph, which is a partition $(P_1, ..., P_p)$ of the recovered samples: $P_1 \cup ... \cup P_p = \mathcal{R}$.

4. Applying SRATTA

In this section, we evaluate the performance of SRATTA on different datasets. In Section 4.1, we show that SRATTA allows (i) to recover an important proportion of the original data samples (step 1 of our method) and (ii) to correctly group those samples together. In Section 4.2, we compare the grouping step of SRATTA to a simple baseline. This comparison allows us to focus on an interesting feature of SRATTA: the fact that the grouping of samples does not depend on any heterogeneity in the sample distribution across clients: SRATTA works equally well in *i.i.d* or non *i.i.d.* settings. The code for SRATTAis available at https://github.com/owkin/SRATTA.

Dataset used We perform SRATTA on four different datasets. Two of them are image datasets: CIFAR10 (Krizhevsky et al., 2009) and FashionMNIST (Xiao et al., 2017). One is a binary dataset, the Primate Splice-Junction Gene Sequences (hereafter DNA dataset) dataset available in the OpenML suite (Vanschoren et al., 2014). The final dataset is a multi-modal and multi-centric version of the TCGA-BRCA (Tomczak et al., 2015; Ogier du Terrail et al., 2022) dataset, containing binary, discrete and continuous entries. Further details on the dataset used are listed in Appendix E.2. For FashionMNIST and CIFAR10 the possible pixel values of data samples are in $\mathcal{P} = \{k/255, k = 0, \dots, 255\}^d$, which we use as a data prior for these datasets. The DNA data are binary, therefore for this dataset $\mathcal{P} = \{0,1\}^d$. Finally for TCGA-BRCA, we use a data prior on a subset of the features which are binary: $\mathcal{P} = \{0,1\}^F \times \mathbb{R}^{d-F}$.

Metrics As noted in Remark 3.4, as the sample recovery process is exact up to machine precision, it is irrelevant to compare the recovered data samples with their original counterpart. Recall the definition of the groups P_i inferred by the attacker in Step 3 (bis). We will use the following metrics.

$\rho_{\text{recovered}} \overset{\text{def}}{=} \#\mathcal{R}/\#\mathcal{D}$ is the ratio of recovered samples and measures the quality of the recovery process.

$\rho_{\text{matched}} \overset{\text{def}}{=} \mathcal{R}_{\text{matched}}/\#\mathcal{D}$, where $\mathcal{R}_{\text{matched}}$ is the number of samples which have been recovered and grouped with at least another sample.

$\rho_{\text{component}}$ is the ratio between the average size of the K largest components in the partition $(P_1, ..., P_p)$ and the average size of a client's dataset.

The metrics above measure the quality of the recovery step but do not give indications on the quality of the grouping *of the recovered samples*. This is why we also use the V-measure of the partition $(P_1, ..., P_p)$ of $\mathcal{R}$ w.r.t. the true partition of $\mathcal{R}$ between the K clients (Rosenberg & Hirschberg, 2007): $V_{\text{recovered}}$. Informally, the V-measure, $V_{\text{recovered}}$, is a mix of two quantities, *homogeneity*: if grouped samples are from the same client, and *completeness*: if samples from the same client are in a single cluster.

Remark 4.1. Note that more standard metrics such as recall or precision are insufficient to provide a satisfying ranking of different groupings as they cannot measure completeness of the partitions and would thus give a similar score to groupings of drastically different qualities.

Finally we introduce $V_{\text{normalized}} \overset{\text{def}}{=} \rho_{\text{recovered}} \times V_{\text{recovered}}$ the Vmeasure normalized by the recovery ration in order to measure the quality of the attack as a whole (recovery **and** grouping). All metrics have values in $[0,1]$. This way, reported numbers are not inflated in the case where one step of the attack is failing indeed $V_{\text{normalized}}$ is equal to one if and only if the attack works perfectly: $\rho_{\text{recovered}} = 1$, $\rho_{\text{matched}} = 1$, $\rho_{\text{component}} = 1$ and $V_{\text{normalized}} = 1$.

For additional details on these metrics, we refer to Appendix E.1

Running the attack We assume that the attacker is in a typical data science workflow where (i) multiple learning rates are tested, in order to select the best one, and (ii) different trainings will be run for each learning rate, in order to have a measure of the uncertainty. Thus, it is plausible that the attacker has access to training sequences for different seeds and different learning rates.

4.1. Evaluating the attack on four datasets

For each dataset, we choose realistic hyper-parameters (HP) (learning rate, number of hidden neurons), and we perform FL trainings, with 5 clients each containing $\#\mathcal{D}_k = 100$ data samples and with $n_{\text{updates}} = 5$, $t_{\max} = 20$, as described in Section 2.1 and Algorithm 1. We then perform SRATTA using only the model parameters at each round θ_t and report the metrics $\rho_{\text{recovered}}$, ρ_{matched}, $\rho_{\text{component}}$ and

$V_{\text{normalized}}$ in Figure 1.

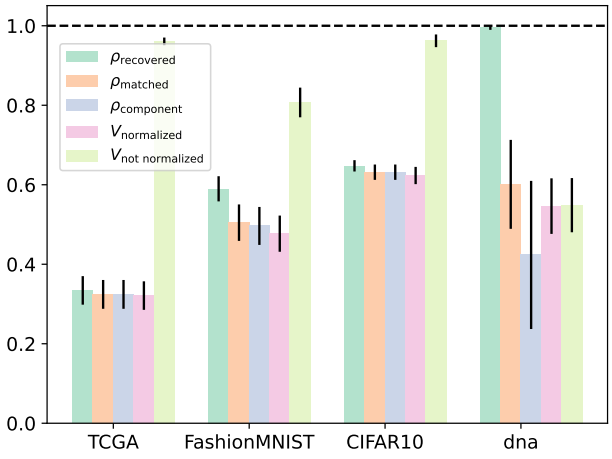


Figure 1. Result of SRATTA for 4 different datasets. In each case, we consider $K = 5$ clients with datasets of size $\#\mathcal{D}_k = 100$, $n_{\text{updates}} = 5$, $t_{\max} = 20$ with 20 training attacks in each case. All the hyperparameters used in this figures are listed in Table 4.

Impact of the hyper-parameters In Appendix G, we explore in detail the impact of the different HP on the performance of SRATTA on FashionMNIST. It confirms the intuition that as the number of hidden neurons, trainings or $t_{\max}$ grows, so does the performance. It also demonstrates that increasing the batch size (1-32) or the number of updates (1-20) significantly degrades the quality of SRATTA. However, simply increasing these parameters would not constitute a proper defense, as the proportion of recovered samples is still significant (> 0.1). Finally, we show that there is an optimal set of learning rates for which the grouping part of the attack works particularly well. This set of learning rates does not necessarily contain the optimal learning rate in terms of performance, but is close enough so that it will be tested during an HP search.

4.2. Dependence on inter-client datasets' homogeneity

In this section, we explore the fact that *a priori*, our grouping method is not sensitive to inter-client datasets' homogeneity, as it relies on a combinatorial argument. Indeed its is harder to group samples when clients' datasets are similar (homogeneous). In order to test this hypothesis we use Dirichlet sampling (Hsu et al., 2019) on samples' labels to build our clients with varying heterogeneity by varying the α parameter of the Dirichlet distribution (cf Appendix E.3). We compare SRATTA with a naïve attack which consists in performing a K-means algorithm, K being the number of clients, on *the recovered data samples $\mathcal{R}$ (from Step 1)*.

We compare both methods using the normalized V measure as defined above, which is particularly interesting as it takes into account both the homogeneity, which favours SRATTA, as well as completeness, which favour K-means. We report the results in Fig.2. In this figure, each point of the curve corresponds to a different dataset whose labels are more homogeneously distributed amongst clients as α increases. α is not a hyper-parameter chosen by the attacker or by the clients, but a parametrization of intrinsic data heterogeneity.

The K-means attack works well in the heterogeneous case, and not at all in the homogeneous case, while SRATTA works reasonably well in all cases.

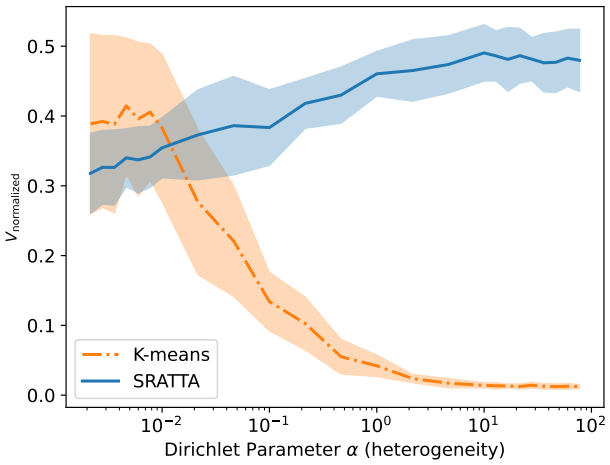


Figure 2. Success rate of SRATTA and K-means as a function of inter-client homogeneity measured by the Dirichlet parameter α. Cf Appendix E.3 and Table 4 for experimental details

5. Fighting secure disaggregation

To defend against this new type of attack, we believe that clients should take matters into their own hands, actively controlling the updates shared to the server.

We propose the following defense mechanism. At each round t, each client k dynamically checks its own activation set $\mathcal{A}^h_{t,k}$ of each neuron h (for a formal definition, see Equation (26) in Appendix F.1). If this activation set is too large to be attacked (larger than a threshold q), the corresponding update is shared. Otherwise, the neuron h is frozen:

$$\text{If} \quad \#\mathcal{A}^h_{t,k} \leqslant q, \quad \overline{W}^h_{t,k,n_{\text{updates}}} \text{ is reset to } \overline{W}^h_{t-1}. \quad (10)$$

This removes easy-to-recover activation sets with less than q samples and nullifies the effect of SRATTA. This censoring might reduce the final performance of the model. However in table 1, we show that we are able to effectively defend ourselves against SRATTA while preserving the accuracy of the model. In Appendix F.1, Figures 6 and 7 we provide more evidence that this defense does not affect model accuracy independently of the learning rate. This result is

coherent with related works on gradient pruning (Sattler et al., 2019) or dropout (Srivastava et al., 2014) where models are efficiently trained with a higher proportion of frozen neurons than ours.

We observe situations where neurons were activated by several samples but only one of them with significant amplitude, allowing reconstruction. Moving from $q = 1$ to $q = 4$ improves the resilience of the defense to this case.

Finally, note that the grouping part of SRATTA is relevant only if $n_{\text{updates}} > 1$. Enforcing $n_{\text{updates}} = 1$ effectively breaks sample matching, but it is out of the scope of this paper, for reasons detailed in Appendix F.4.

Dataset	$\rho_{\text{recovered}} \downarrow$	$V_{\text{normalized}} \downarrow$	P_{censored}	Model Acc. $\uparrow$
CIFAR10 ($q = 0$)	0.585 ± 0.015	0.490 ± 0.019	0.000 ± 0.000	0.261 ± 0.039
CIFAR10 ($q = 1$)	0.002 ± 0.002	0.002 ± 0.002	0.098 ± 0.011	0.272 ± 0.033
CIFAR10 ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.180 ± 0.014	$\mathbf{0.274 \pm 0.040}$
DNA ($q = 0$)	0.516 ± 0.080	0.233 ± 0.032	0.000 ± 0.000	0.929 ± 0.017
DNA ($q = 1$)	0.035 ± 0.015	0.031 ± 0.013	0.005 ± 0.002	0.929 ± 0.019
DNA ($q = 4$)	$\mathbf{0.000 \pm 0.001}$	$\mathbf{0.000 \pm 0.001}$	0.038 ± 0.006	$\mathbf{0.932 \pm 0.020}$
FashionMNIST ($q = 0$)	0.476 ± 0.027	0.284 ± 0.015	0.000 ± 0.000	$\mathbf{0.732 \pm 0.053}$
FashionMNIST ($q = 1$)	0.009 ± 0.004	0.006 ± 0.004	0.042 ± 0.006	0.729 ± 0.043
FashionMNIST ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.149 ± 0.022	0.731 ± 0.046
TCGA ($q = 0$)	0.279 ± 0.016	0.193 ± 0.010	0.000 ± 0.000	$\mathbf{0.660 \pm 0.065}$
TCGA ($q = 1$)	0.051 ± 0.016	0.036 ± 0.012	0.135 ± 0.016	0.658 ± 0.071
TCGA ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.257 ± 0.030	0.658 ± 0.071

Table 1. Influence of the defense on the efficiency of SRATTA and on training performance. In this table we take the point of view of the **defender/client**. The defense strategy must therefore i) not impact the model accuracy compared to the baseline without defense and ii) nullify the attack. P_{censored} is the average proportion of neurons censored per update on the total number of neurons. We simulate an attack performed on clients defending themselves while conducting a grid search where they test 20 different learning rates. This experiment is repeated 10 times. The model accuracy reported is the best one across learning rates, averaged over repetitions. This setting highlights the defense success to prevent the attack, with almost no loss of accuracy.

5.1. Strengths and limitations of the defense

As in most adversarial settings, the defense mechanism we present in this section is not proved to be absolute. Apart from the empirical evidence presented in **??**, two remarks strengthen our belief that knowing the defense mechanism does not result in a straightforward workaround for the attacker.

- If the attacker knows that certain neurons have been censored, retrieving them is not obvious, as this censoring is done at client level, and is not directly reflected in the aggregation.
- Even assuming that the attacker is able to trace which neuron has been censored by which client, they can only infer that the data point which trigerred the defense lies in a certain half space (and this is assuming the knowledge of the weights defining the half space, which is not obvious considering they are modified by local updates).

However, this does not provide a formal proof of the inviolability of the defense. Another point of concern could be that the defense could impact model accuracy, in the case where underrepresented samples are censored too often, and which is not reflected in the datasets we studied. To mitigate this concern, we note that the defense only censors the update for specific neurons for specific clients. Thus a neuron censored by Client A can be updated by Client B in this very round, and thus not be censored at another round for the same sample. Moreover, for a sample to have virtually no impact on any neuron during the training, all neurons activated by the underrepresented sample need to be censored.

6. Discussion

We believe SRATTA to be an important proof of concept opening new interesting research avenues in both the design of attacks, and of privacy layers in cross-silo FL settings. In particular, it stresses the fact that clients have to play a bigger role in actively defending themselves, and that relying on secure centralized computations alone is not enough. In our work, the restriction to models with a fully connected first layer, such as MLPs, limits the impact of SRATTA.

Extension to CNNs. Assessing if this attack can be extended to a broader class of models, such as CNNs, will be important to make sure that clients implement the appropriate defense mechanisms. In Pan et al. (2022); Zhu & Blaschko; Boenisch et al. (2021); Kariyappa et al. (2022), the attack on MLPs is extended to CNNs by attacking the first fully connected layer. The key component needed for SRATTA to work on this setting, is a prior on the deep features allowing to filter out reconstruction candidates relying solely on their feature representations as is done now in SRATTA using raw data. Previous work Pan et al. (2022); Zhu & Blaschko; Dong et al. (2021) have shown that inverting the convolutional part of the network is possible, which could allow us to use priors on raw data space, and select reconstructed deep features based on their inversion to the raw data space. This is but one way to approach the problem, and we are convinced subsequent works will be able to bridge this gap.

Acknowledgements

The authors thank the anonymous reviewers, ethics reviewer, and meta-reviewer for their feedback and ideas, which significantly improved the paper. The authors also thank Geneviève Robin , Alex Nowak and John Klein for their help in proofreading the paper as well as Mathieu Andreux for his involvment in initial discussions. The authors are supported by Owkin, Inc.

References

Andreux, M., Manoel, A., Menuet, R., Saillard, C., and Simpson, C. Federated survival analysis with discrete-time cox models. *arXiv preprint arXiv:2006.08997*, 2020.

Aono, Y., Hayashi, T., Wang, L., Moriai, S., et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

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 preprint arXiv:2112.02918*, 2021.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Courtiol, P., Tramel, E. W., Sanselme, M., and Wainrib, G. Classification and disease localization in histopathology using only global labels: A weakly-supervised approach. *arXiv preprint arXiv:1802.02212*, 2018.

Cox, D. R. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

Cramer, R., Damgård, I. B., et al. *Secure multiparty computation*. Cambridge University Press, 2015.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dimitrov, D. I., Balunovic, M., Konstantinov, N., and Vechev, M. Data leakage in federated averaging. *Transactions on Machine Learning Research*, 2022.

Dong, X., Yin, H., Álvarez, J. M., Kautz, J., and Molchanov, P. Deep neural networks are surprisingly reversible: A baseline for zero-shot inversion. *ArXiv*, abs/2107.06304, 2021.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

El Ouadrhiri, A. and Abdelhadi, A. Differential privacy for deep and federated learning: A survey. *IEEE Access*, 10: 22359–22380, 2022.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=fwzUgo0FM9v.

Frey, B. J. and Dueck, D. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Ganju, K., Wang, Q., Yang, W., Gunter, C. A., and Borisov, N. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pp. 619–633, 2018.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

Ghaznavi, F., Evans, A., Madabhushi, A., and Feldman, M. Digital imaging in pathology: whole-slide imaging and beyond. *Annu Rev Pathol*, 8(1):331–359, 2013.

Hatamizadeh, A., Yin, H., Molchanov, P., Myronenko, A., Li, W., Dogra, P., Feng, A., Flores, M. G., Kautz, J., Xu, D., and Roth, H. R. Do Gradient Inversion Attacks Make Federated Learning Unsafe? (arXiv:2202.06924). URL http://arxiv.org/abs/2202.06924.

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.

Heyndrickx, W., Mervin, L., Morawietz, T., Sturm, N., Friedrich, L., Zalewski, A., Pentina, A., Humbeck, L., Oldenhof, M., Niwayama, R., et al. Melloddy: cross pharma federated learning at unprecedented scale unlocks benefits in qsar without compromising proprietary information. 2022.

Hsu, T.-M. H., Qi, H., and Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241, 2021.

Jenkins, S. P. Survival analysis. *Unpublished manuscript, Institute for Social and Economic Research, University of Essex, Colchester, UK*, 42:54–56, 2005.

Kadra, A., Lindauer, M., Hutter, F., and Grabocka, J. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

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. *CoRR*, abs/2209.05578, 2022. doi: 10.48550/arXiv.2209.05578. URL https://doi.org/10.48550/arXiv.2209.05578.

Kim, S., Koh, K., Lustig, M., Boyd, S., and Gorinevsky, D. An interior-point method for large-scale l1-Regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

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

Lam, M., Wei, G.-Y., Brooks, D., Reddi, V. J., and Mitzenmacher, M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *International Conference on Machine Learning*, pp. 5959–5968. PMLR, 2021.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Long, G., Tan, Y., Jiang, J., and Zhang, C. Federated learning for open banking. In *Federated learning*, pp. 240–254. Springer, 2020.

Mallat, S. G. and Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Ogier du Terrail, J., Ayed, S.-S., Cyffers, E., Grimberg, F., He, C., Loeb, R., Mangold, P., Marchand, T., Marfoq, O., Mushtaq, E., et al. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. *arXiv preprint arXiv:2210.04620*, 2022.

Ogier du Terrail, J., Leopold, A., Joly, C., Béguier, C., Andreux, M., Maussion, C., Schmauch, B., Tramel, E. W., Bendjebbar, E., Zaslavskiy, M., Wainrib, G., Milder, M., Gervasoni, J., Guerin, J., Durand, T., Livartowski, A., Moutet, K., Gautier, C., Djafar, I., Moisson, A.-L., Marini, C., Galtier, M., Balazard, F., Dubois, R., Moreira, J., Simon, A., Drubay, D., Lacroix-Triki, M., Franchet, C., Bataillon, G., and Heudel, P.-E. Federated learning for predicting histological response to neoadjuvant chemotherapy in triple-negative breast cancer. *Nature Medicine*, 2023. doi: 10.1038/s41591-022-02155-w. URL https://doi.org/10.1038/s41591-022-02155-w.

Pan, X., Zhang, M., Yan, Y., Zhu, J., and Yang, Z. Exploring the security boundary of data reconstruction via neuron exclusivity analysis. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3989–4006, 2022.

Pasquini, D., Francati, D., and Ateniese, G. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2429–2443, 2022.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pejó, B., Tóth, A., and Biczók, G. Quality inference in federated learning with secure aggregation, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Rosenberg, A. and Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1043.

Rubinstein, R., Zibulevsky, M., and Elad, M. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. 2008.

Saillard, C., Dehaene, O., Marchand, T., Moindrot, O., Kamoun, A., Schmauch, B., and Jegou, S. Self supervised learning improves dmmr/msi detection from histology slides across multiple cancers. *arXiv preprint arXiv:2109.05819*, 2021.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Schmauch, B., Romagnoni, A., Pronier, E., Saillard, C., Maillé, P., Calderaro, J., Kamoun, A., Sefta, M., Toldo, S., Zaslavskiy, M., et al. A deep learning model to predict rna-seq expression of tumours from whole slide images. *Nature communications*, 11(1):1–15, 2020.

Shokri, R. and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

Tomczak, K., Czerwińska, P., and Wiznerowicz, M. Reviewthe cancer genome atlas (tcga): an immeasurable source of knowledge. *Contemporary Oncology/Współczesna Onkologia*, pp. 68–77, 2015. ISSN 1428-2526. doi: 10.5114/wo.2014.47136. URL http://dx.doi.org/10.5114/wo.2014.47136.

Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., and Stuart, J. M. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120, 2013.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Xu, J., Hong, C., Huang, J., Chen, L. Y., and Decouchant, J. Agic: Approximate gradient inversion attack on federated learning. *arXiv preprint arXiv:2204.13784*, 2022.

Yin, H., Mallya, A., Vahdat, A., Alvarez, J. M., Kautz, J., and Molchanov, P. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

Zhang, T., Ramakrishnan, R., and Livny, M. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

Zhang, Z., Xu, Y., Yang, J., Li, X., and Zhang, D. A survey of sparse representation: Algorithms and applications. *IEEE Access*, 3:490–530, 2015. doi: 10.1109/ACCESS.2015.2430359.

Zhao, B., Mopuri, K. R., and Bilen, H. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 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, 2023.

Zheng, Z., Zhou, Y., Sun, Y., Wang, Z., Liu, B., and Li, K. Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges. *Connection Science*, 34(1):1–28, 2022.

Zhu, J. and Blaschko, M. R-GAP: Recursive Gradient Attack on Privacy. URL http://arxiv.org/abs/2010.07733.

Zhu, L., Liu, Z., and Han, S. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

A. Proofs of the theorems

A.1. Proof of Proposition 3.1

We provide here a proof of Proposition 3.1, which was already stated, with other notations in various works such as Aono et al. (2017); Geiping et al. (2020); Pan et al. (2022). We start by computing the gradient of the loss at a single data point, before proving the proposition on a batch, and finally extending it to a loss which is not decomposable on the points of the batch (this will be useful for the Cox loss in the TCGA dataset).

In order to make differentials more explicit, we write $m(x;\theta) \overset{\text{def}}{=} m_\theta(x)$ for $x \in \mathbb{R}^d, z \in \mathbb{R}^H, \phi \in \mathbb{R}^q$ and $\theta \in \mathbb{R}^p$.

1. Gradient of the loss at a single point. Let $(x, y) \in \mathbb{R}^d \times \mathcal{Y}$. We recall that the neural network considered starts with a fully connected layer, and hence

$$m(x;\theta) = f_\phi(z(x)),\ z(x) \overset{\text{def}}{=} \mathrm{ReLU}(Wx + b)$$

For any $1 \leqslant h \leqslant H$, denote with z^h the activation of the h-th hidden neuron: $z^h(x) \overset{\text{def}}{=} \mathrm{ReLU}(W^h x + b^h)$. Note that $z = (z^h)_{1 \leqslant h \leqslant H}$. Using the chain rule, it holds

$$\begin{aligned} \nabla_{W^h} \ell(m(x;\theta), y) &= \frac{\partial \ell(f_\phi(z(x)), y)}{\partial z^h} \frac{\partial z^h(x)}{\partial W^h}, \\ \nabla_{b^h} \ell(m(x;\theta), y) &= \frac{\partial \ell(f_\phi(z(x)), y)}{\partial z^h} \frac{\partial z^h(x)}{\partial b^h}. \end{aligned} \tag{11}$$

Note that in the main text (cf Equation (3)), we define

$$\frac{\partial \mathcal{L}(\theta; x, y)}{\partial z^h} \overset{\text{def}}{=} \frac{\partial \ell(f_\phi(z(x)), y)}{\partial z^h} \tag{12}$$

Therefore,

$$\begin{aligned} \nabla_{W^h} \ell(m(x;\theta), y) &= \frac{\partial \mathcal{L}(\theta; x, y)}{\partial z^h} \frac{\partial z^h(x)}{\partial W^h}, \\ \nabla_{b^h} \ell(m(x;\theta), y) &= \frac{\partial \mathcal{L}(\theta; x, y)}{\partial z^h} \frac{\partial z^h(x)}{\partial b^h}. \end{aligned} \tag{13}$$

Now if $W^h x_i + b^h \leqslant 0$, then $\frac{\partial z^h(x)}{\partial W^h} = 0$ and $\frac{\partial z^h(x)}{\partial b^h} = 0$, taking the convention that the gradient of the ReLU is 0 at 0.

On the other hand, if $W^h x + b^h > 0$, then the gradient of the ReLU is 1 and $\frac{\partial z^h(x)}{\partial W^h} = x$, $\frac{\partial z^h(x)}{\partial b^h} = 1$. In both cases, we have

$$\begin{aligned} \nabla_{W^h} \ell(m(x;\theta), y) &= \lambda(\theta; x, y)\ x, \\ \nabla_{b^h} \ell(m(x;\theta), y) &= \lambda(\theta; x, y). \end{aligned} \tag{14}$$

with $\lambda(\theta; x, y) = \frac{\partial \mathcal{L}(\theta; x, y)}{\partial z^h} \mathbf{1}_{W^h x + b^h > 0}$.

2. Gradient of the loss on the whole batch. Now, let $D = (x_i, y_i)_{1 \leqslant i \leqslant b} \in \mathbb{R}^b$ be a batch of size b.

The loss computed over a batch D is the following sum:

$$\mathcal{L}(\theta; D) = \sum_{i=1}^{b} \ell(m(x_i;\theta), y_i) \tag{15}$$

and hence, by applying the gradient and using Equation (14)

$$\begin{aligned}\nabla_{W^h}\ell(m(x;\theta),y) &= \sum_{i=1}^{b}\lambda(\theta;x_i,y_i)\,x_i,\\ \nabla_{b^h}\ell(m(x;\theta),y) &= \sum_{i=1}^{b}\lambda(\theta;x_i,y_i).\end{aligned} \tag{16}$$

By definition of the activation set $\mathcal{A}^h(\theta,D)$ in Equation (3), and that of λ above we have

$$\lambda(\theta;x_i,y_i)\neq 0 \text{ i.i.f. } x_i\in\mathcal{A}^h(\theta,D),$$

which concludes the proof.

3. Gradient of a loss which is not separable on a batch Assume now that the loss is not decomposable along points of the batch, but that we have instead

$$\mathcal{L}(\theta;D)\overset{\text{def}}{=}\ell(m_\theta(x_D),D),\ \text{where } m_\theta(x_D)\overset{\text{def}}{=}(m_\theta(x_i))_{1\leqslant i\leqslant b}\in\mathbb{R}^{C\times b}, \tag{17}$$

for some differentiable loss $\ell(\cdot,D):\mathbb{R}^{C\times b}\to\mathbb{R}$ on the batch. Then in the same way, we define the activation set as

$$\begin{aligned}\mathcal{A}^h(\theta,D)\overset{\text{def}}{=}\{x_i\in D\ &: W^h\cdot x_i+b^h>0\\ &\text{and }\frac{\partial\mathcal{L}(\theta;D)}{\partial z_i^h}\neq 0\},\end{aligned} \tag{18}$$

where $\frac{\partial\mathcal{L}(\theta;D)}{\partial z_i^h}$ is the differential of the loss w.r.t. the output of the first neuron at parameter θ on sample x, which is formally the partial derivative of $Z=(z_1,...,z_b)\in\mathbb{R}^{C\times b}\mapsto\ell((f_\phi(z_i))_{1\leqslant i\leqslant b},D)$ with respect to $[Z]_{hi}=z_i^h$ at point $Z=(\mathrm{ReLU}(Wx_i+b))_{1\leqslant i\leqslant b}$.

We can then compute the partial derivatives, and obtain exactly the same result as the one above.

A.2. Proof of Proposition 3.2

As the clients are performing standard SGD, the updates of the clients are a sum of the gradients computed at each update. Besides, as the FL protocol is FedAvg, the aggregated model θ_t is a simple average of all the client updates. Therefore:

$$\Delta\theta_t=-\tfrac{\eta}{K}\sum_{k=1}^{K}\sum_{i=0}^{n_{\text{updates}}-1}\nabla\mathcal{L}(\theta_{t,k,i},B_{t,k,i})$$

And thus for the first layer's weights and biases we have:

$$\Delta W_t^h=-\tfrac{\eta}{K}\sum_{k=1}^{K}\sum_{i=0}^{n_{\text{updates}}-1}\sum_{j=0}^{b}\lambda(\theta_{t,k,i};x_j^{B_k^i},y_j^{B_k^i})x_j$$

$$\Delta b_t^h=-\tfrac{\eta}{K}\sum_{k=1}^{K}\sum_{i=0}^{n_{\text{updates}}-1}\sum_{j=0}^{b}\lambda(\theta_{t,k,i};x_j^{B_k^i},y_j^{B_k^i})$$

Where $x_j^{B_k^i}$ indicates the j-th sample of the batch of client k at update i.

Therefore, Proposition 3.2 is a straightforward consequence of Proposition 3.1.

A.3. Proofs of Theorem 3.5

Let $x \in \mathcal{A}_t^h$. Assume that x is a sample from client k; since we suppose that Assumption 2.4 is satisfied, then x must be in one of the batches of client k, *i.e.*, must belong to one of the batches in the sequence $(B_{t,k,i})_{0 \leqslant i \leqslant n_{\text{updates}}-1}$. Note that if Assumption 2.4 were not satisfied, it could belong to another client's batches.

Let i_k^h be the minimal index of a batch in the sequence $(B_{t,k,i})_{0 \leqslant i \leqslant n_{\text{updates}}-1}$ containing a sample which activates neuron h, (it is well defined because x is such an element), that is

$$i_k = \min\{0 \leqslant i \leqslant n_{\text{updates}} \ : \ \mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i}) \neq \varnothing\}.$$

Since for all $i < i_k^h$, we have $\mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i}) = \varnothing$, by Proposition 3.1, we have $\nabla_{\overline{W}^h}\mathcal{L}(\theta_{t,k,i}, B_{t,k,i}) = 0, \ 0 \leqslant i < i_k^h$. Using the fact that SGD updates are performed, $\overline{W}^h_{t,k,i+1} = \overline{W}^h_{t,k,i} - \eta\nabla_{\overline{W}^h}\mathcal{L}(\theta, B_{t,k,i}) = \overline{W}^h_{t,k,i}$ for all $0 \leqslant i < i_k^h$, meaning that

$$\overline{W}^h_{t-1} = \overline{W}^h_{t,k,0} = ... = \overline{W}^h_{t,k,i_k^h}.$$

Let x' be an element in $\mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i_k^h})$ (it is possible that x' is in fact x). It is in $\mathcal{A}_t^h$ by definition, and since $\overline{W}^h_{t,k,i_k^h} = \overline{W}^h_{t-1}$, it is also in $\widetilde{\mathcal{A}}^h_{t,0}$. Thus, we have proven that there is an element x' from client k in $\widetilde{\mathcal{A}}^h_{t,0}$.

A.4. Proof of Corollary 3.6

Assume all elements of $\widetilde{\mathcal{A}}^h_{t,0}$ come from the same client k. Let's consider $x \in \mathcal{A}_t^h$. According to Theorem 3.5, there exists $x' \in \widetilde{\mathcal{A}}^h_{t,0}$ such that x and x' are from the same client. As x' comes from client k, x comes from client k. Therefore, all elements $x \in \mathcal{A}_t^h$ come from client k. In particular, if $\widetilde{\mathcal{A}}^h_{t,0}$ contains only one element, by Assumption 2.4, this element belongs to one and only one client k. The result then follows.

B. Recovering the activation sets using Orthogonal matching pursuit

The goal of this section is to explain how we do the second step of SRATTA described in **Step 2** of Section 3.2.

B.1. Problem reformulation

Recall from Equation (8) that for each couple (t, h) of neurons and rounds, we try to find $N \in \mathbb{N}$, $(\lambda_r)_{1 \leqslant r \leqslant N}$ and $(x_r)_{1 \leqslant r \leqslant N} \in \mathcal{R}^N$ such that

$$\Delta W_t^h = \sum_{r=1}^{N} \lambda_r x_r \tag{19a}$$

$$\Delta b_t^h = \sum_{r=1}^{N} \lambda_r \tag{19b}$$

$$\forall r, \lambda_r \neq 0 \tag{19c}$$

Let us reformulate Equation (19). Denote with $N_{\mathcal{R}}$ the number of recovered samples in $\mathcal{R}$ and let $x_1, .., x_{N_{\mathcal{R}}} \in \mathcal{R}$ be the list of recovered samples. Let $X \overset{\text{def}}{=} (x_1, ..., x_{N_{\mathcal{R}}})^\top \in \mathbb{R}^{N_{\mathcal{R}} \times d}$ be the matrix of recovered samples, and $\overline{X} \overset{\text{def}}{=} \begin{pmatrix} X & \mathbf{1} \end{pmatrix} \in \mathbb{R}^{N_{\mathcal{R}} \times (d+1)}$ be the extended recovered samples matrix. Equation (19) can simply be written

$$\text{Find } \lambda \in \mathbb{R}^{N_{\mathcal{R}}} \text{ s.t. } \overline{X}^\top \lambda = \overline{W}_t^h, \tag{20}$$

and a recovered activation set can be defined from λ as the set of recovered samples whose corresponding coefficient λ_r is non zero, that is $\widetilde{\mathcal{A}}_t^h(\lambda) \overset{\text{def}}{=} \{x_r \in \mathcal{R} \ : \ \lambda_r \neq 0\}$.

B.2. Restriction to small activation sets

Solving Equation (20) and using a resulting λ to build the recovered activation set $\widetilde{\mathcal{A}}_t^h(\lambda)$ is not satisfactory. Indeed, as soon as $N_{\mathcal{R}} > d+1$ there are infinitely many solutions to Equation (20), corresponding to potentially infinitely many recovered activation set $\widetilde{\mathcal{A}}_t^h(\lambda)$. In that case, there is no way of knowing which one of these approximations is actually the true activation set.

As explained in the main text (see Section 3.2), to avoid this problem, we further constrain Equation (20) to the λ corresponding to activation sets of size at most N_{max}, where N_{max} is a hyper-parameter selected by the attacker in order to recover meaningful activation sets, and where $N_{\text{max}} \ll d$ in order to avoid the problem of having infinitely many solutions. This can be formalized as the following problem

$$\text{Find } \lambda \in \mathbb{R}^{N_{\mathcal{R}}} \text{ such that } \|\lambda\|_0 \leqslant N_{\text{max}} \text{ and } \overline{X}^\top \lambda = \overline{W}_t^h, \tag{21}$$

where $\|\lambda\|_0$ denotes the number of non zero coefficients of λ. The intuition behind this restriction is that (i) we wish to recover small activation sets and (ii) it is much less likely that Equation (21) has a solution with a small N_{max}, so that if we find one, the corresponding $\widetilde{\mathcal{A}}_t^h(\lambda)$ is likely to be equal to the true activation set $\mathcal{A}_t^h$. In practice, to solve Equation (21), we solve the relaxed version

$$\text{Find } \lambda \in \mathbb{R}^{N_{\mathcal{R}}} \text{ such that } \lambda \in \arg\min_{\|\lambda\|_0 \leqslant N_{\text{max}}} \left\| \overline{X}^\top \lambda - \overline{W}_t^h \right\|^2. \tag{22}$$

We then check that the recovered solution λ satisfies $\overline{X}^\top \lambda = \overline{W}_t^h$ with sufficient precision (if it is not the case, this means that Equation (21) has no solution). In the main paper, we denote with L_{rec} the set of couples (t, h) for which Equation (21) has a solution.

This last formulation is quite standard, and can be solved using different methods, such as orthogonal matching pursuit (OMP), (Rubinstein et al., 2008; Mallat & Zhang, 1993) or the LASSO method (Kim et al., 2007). In practice, we solve this problem using OMP, and N_{max} is selected in order to lead to clusters of reasonable size. OMP is based on a greedy algorithm and builds the activation set iteratively. At each step, it adds to the activation set the sample which is most highly correlated with the residual. It is called orthogonal matching pursuit because at each iteration, the residual is recomputed using an orthogonal projection on the current state of the activation set. For more details, see Mallat & Zhang (1993) as well as the scikit-learn documentation Pedregosa et al. (2011).

C. Relaxation of hypotheses

We note that although we assume full-participation in the rest of the article it is not strictly necessary for SRATTA to work. In addition, the local and global learning rate, batch size, number of updates do not have to be fixed. The only important thing is to perform multiple gradient updates, and that the learning rate is the same for the weights and the bias at each local minibatch's gradient step. The sum across clients could be weighted (in fact this can be included in the label variable, by adding an extra label variable k for the client).

D. Pseudocode of SRATTA

The pseudocode of SRATTA can be found below in Algorithm 3.

Algorithm 3 SRATTA

Require: number of trainings $num_{trainings}$, number of rounds per training t_{max}, number of hidden neurons in the first layer of the model H, model updates for all trainings, rounds, neurons $(\Delta W_t^h, \Delta b_t^h)$, a data prior on the data $\mathcal{P}$, maximum number of samples allowed in the Orthogonal Matching Pursuit $N_{\max}$

———————- Step 1: Sample recovery ———————-

$\mathcal{R} \leftarrow \varnothing$
for $training = 1$ **to** $num_{trainings}$ **do**
  for $t = 1$ **to** t_{max} **do**
    for $h = 1$ **to** H **do**
      $candidate = \Delta W_t^h / \Delta b_t^h$
      if $distance(candidate, \mathcal{P}) \approx 0$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{candidate\}$
      end if
    end for
  end for
end for

———————- Step 2: Reconstruction of activation sets ———————-

$L_{rec} \leftarrow \varnothing$
for $training = 1$ **to** $num_{trainings}$ **do**
  for $t = 1$ **to** t_{max} **do**
    for $h = 1$ **to** H **do**
      $\{(\lambda_r, x_r) : 1 \leqslant r \leqslant N\} \leftarrow$ Orthogonal Matching Pursuit$(\mathcal{R}, \Delta W_t^h, N_{max})$
      $\mathcal{A}_t^h = \{x_r : 1 \leqslant r \leqslant N\}$
      if $\mathcal{A}_t^h \neq \varnothing$ and $\Delta W_t^h \approx \sum_{r=1}^N \lambda_r x_r$ and $\Delta b_t^h \approx \sum_{r=1}^N \lambda_r$ **then**
        $L_{rec} \leftarrow L_{rec} \cup (h, t)$
      end if
    end for
  end for
end for

———————- Step 3a): Definitions ———————-

for $training = 1$ **to** $num_{trainings}$ **do**
  for $(h, t) \in L_{rec}$ **do**
    $\widetilde{\mathcal{A}}_{t,0}^h \leftarrow \{x \in \mathcal{A}_t^h \ : \ W_{t-1}^h x + b_{t-1}^h > 0\}$
  end for
end for
$relationshiplist \leftarrow []$
for $training = 1$ **to** $num_{trainings}$ **do**
  for $(h, t) \in L_{rec}$ **do**
    $relationshiplist.append([\widetilde{\mathcal{A}}_{t,0}^h, \mathcal{A}_t^h \backslash \widetilde{\mathcal{A}}_{t,0}^h])$
  end for
end for

———————- Step 3b): Creating groups ———————-

$\mathcal{G} \leftarrow$ edgeless graph where each node $\in \mathcal{R}$
while the number of edges in $\mathcal{G}$ is growing **do**
  for $relationship \in relationshiplist$ **do**
    if all samples from $relationship[0]$ are connected **then**
      Draw an edge in $\mathcal{G}$ between all samples in $relationship[0]$ and all samples in $relationship[1]$
    end if
  end for
end while
$clusteredclients \leftarrow$ connected components of $\mathcal{G}$

Ensure: $clusteredclients$

E. Details on numerical experiment

E.1. Details on numerical experiment

Additional details on the metrics used. We can see the output of SRATTA as a partition of the set of recovered samples $P_1 \cup ... \cup P_p = \mathcal{R}$, where the sets $P_1, ..., P_p$ which are disjoint. The sets P_i are inferred by the attacker in step 3b), and represent the largest possible groups of samples which they can build. We will assess the quality of the sample recovery process using the following different metrics.

$\rho_{\text{recovered}} \overset{\text{def}}{=} \#\mathcal{R}/\#\mathcal{D}$ is the ratio of recovered samples and measures the quality of the recovery process.

ρ_{matched} is the ratio of samples in the dataset which have been recovered and grouped with at least another sample (that is samples that belong to P_i with $\#P_i > 1$).

$\rho_{\text{component}}$ is the ratio between the average size of the K biggest elements in the partition $(P_1, ..., P_p)$ and the average size of a client dataset $\#\mathcal{D}_k$ (In practice, all client datasets will be of equal size). The intuition is that the K biggest elements of the partition would ideally each correspond to a different client dataset.

$V_{\text{normalized}} \overset{\text{def}}{=} \rho_{\text{recovered}} * V_{\text{recovered}}$ is the normalized V-measure of the grouped recovered samples, where $V_{\text{recovered}}$ is the V-measure (Rosenberg & Hirschberg, 2007) of the partition $(P_1, ..., P_p)$ of $\mathcal{R}$ with respect to the true partition of $\mathcal{R}$ into K sets across the K-clients. Exactly in the same way that the $F1$ score mixes precision and recall, here, the V-measure mixes two measures of the quality of the partition $P_1, ..., P_p$: homogeneity $h \in [0, 1]$ and completeness $c \in [0, 1]$. Homogeneity measures the diversity of each class: $h = 1$ if each P_i contains elements coming from only one client (this will be the case if we have no false positive). Completeness measures how each true client is distributed amongst different groups P_i ; $c = 1$ if for all k, all elements belonging to client k are assigned to the same element of the partition P_{i_k}. $V_{\text{recovered}}$ is the harmonic mean of these two quantities, so that $V_{\text{recovered}} = 0$ if either $c, h = 0$ and $V_{\text{recovered}} = 1$ if both $c, h = 1$. This measure is particularly adapted to our setting as we do not have a natural metric to compare different clusterings with a different number of clusters. We multiply by $\rho_{\text{recovered}}$ so that the attacker clusters all data points and to the right client if and only if $V_{\text{normalized}} = 1$. $\rho_{\text{recovered}}$ can be seen as the recall but taken over all samples not only recovered samples.

E.2. Datasets used

E.2.1. CIFAR10

CIFAR10 (Krizhevsky et al., 2009) is one of the most well-known image classification datasets. There are $60,000$ object-centric images of relatively low resolution 32x32x3 that are labeled with 10 different mutually exclusive classes (horse, frog, truck, airplane, automobile, bird, cat, deer, dog, ship).

E.2.2. FASHIONMNIST

FashionMNIST (Xiao et al., 2017) was released as a more challenging alternative to MNIST (LeCun et al., 1998) with grayscale images 28x28 of fashion items split across 10 classes (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot). There are $70,000$ images in total.

E.2.3. DNA

Primate Splice-Junction Gene Sequences (or DNA dataset) dataset is a part of the OpenML suite (Vanschoren et al., 2014). It consists of $3,186$ samples which are splice junctions. Each sample is a vector of 180 indicator binary variables and the ML task is a 3-way classification task (ei, ie, neither). More details can be found in (Vanschoren et al., 2014).

For all the datasets above, we use MLP networks directly on raw pixels that are just rescaled to be between 0 and 1 (with 255 distinct values).

E.2.4. TCGA-BRCA

The Cancer Genome Atlas (TCGA) (Weinstein et al., 2013; Tomczak et al., 2015) is an open initiative to gather and make available oncology patients' data for multiple modalities such as tabular, radiology, histology and genomics and different cancers. The TCGA is organized by study where each study focuses on a single cancer. BRCA is the BReast CAncer study. The samples in TCGA were collected across multiple institutions that are tagged by barcodes and identifiable, this is also

the case for BRCA. FLamby (Ogier du Terrail et al., 2022) uses such naturally split data inside TCGA-BRCA to provide a realistic distributed dataset where the ML task is survival prediction (Jenkins, 2005). For a more complete treatment of the ML task defined by this dataset see Appendix F in (Ogier du Terrail et al., 2022). We use this tabular dataset with 1088 patients and retrieve for each patient a feature obtained from its histology slides (see *Primer on histology* below if needed). We detail below in *Histology Preprocessing* the exact process used to extract such features and join both datasets.

TCGA license

The data terms can be found https://gdc.cancer.gov/access-data/data-access-processes-and-tools. In particular, as per the GDC data access policy, users should not attempt to identify individual human research participants from whom the data were obtained.

Primer on histology

Histology slides (Ghaznavi et al., 2013) are large-size images obtained by digitizing stained slices of biopsies of the patient's tumor sites. Such images, called Whole Slide Images (WSI), generally comprises millions of pixels representing the slice at different resolution and comprise a large uninformative background. This background has to be removed to maximize the Peak-Signal-to-Noise-Ratio of the information inside the WSI. As this data usually does not fit in GPU RAM, its dimensions are reduced using tiling and feature extraction (Courtiol et al., 2018).

Histology Preprocessing

Like in (Andreux et al., 2020), we use natural splits and retrieve the original histology sides corresponding to each patient using the patient ID. As some patients have multiple histology slides we keep only the first one after alphabetical sorting. We drop all patients for which histology slides could not be found (38 in total making a final total of 1050 patients). We then tile the matter on each slide at x20 magnification using a U-net (Ronneberger et al., 2015) trained on an in-house dataset and compute ResNet-50 features (He et al., 2016) pretrained on IMAGENET (Deng et al., 2009) for each extracted tile. Therefore each patient is associated with a row of clinical data and a variable number of ResNet-50 features proportional to the amount of matter found on the corresponding slide. We then train a linear autoencoder with bottleneck size 256 on the ResNet-50 features from all slides using an L2 cost as is common practice (Schmauch et al., 2020). Once this autoencoder is trained, we encode all features in each slide. Finally, we average all autoencoded features on each slide to get a fixed-sized feature, following MeanPool architectures (Saillard et al., 2021). Therefore for each patient we build a feature vector of size $39 + 256 = 295$ with 39 clinical features and 256 "histology" features.

Clinical Data Preprocessing We use FLamby(Ogier du Terrail et al., 2022)'s preprocessed version of TCGA-BRCA where original features are filtered and binarized except for the age feature (of integer type), which is left unaltered. This allows us to easily define the corresponding data prior to filter out candidates. See (Ogier du Terrail et al., 2022)'s Appendix F1 and (Andreux et al., 2020) for more details about the exact preprocessing steps.

Survival Loss function

One of the foundation models in survival analysis is the linear Cox proportional hazard (Cox, 1972). This model assumes:

$$h(t, x) = h_0(t) \exp(\beta^T x) \tag{23}$$

where h_0 is the baseline hazard function (common to all patients and dependent on time only) and β is the vector of parameters of our linear model. β is estimated by minimization of the negative Cox partial log-likelihood, which compares relative risk ratios:

$$L(\beta) = -\sum_{i:\delta_i=1} \Big[\beta^T x_i - \log\big(\sum_{j:t_j>t_i} \exp(\beta^T x_j)\big)\Big] \tag{24}$$

where i and j index patients and $\delta_i = 1$ indicates an event. This loss is extended in our case to handling $m_\theta(x_i) = \mathrm{ReLU}(Wx_i + b)$ instead of raw x_i where we backpropagate through W and b as well. This is a natural extension explored in (Andreux et al., 2020):

$$L(\beta) = -\sum_{i:\delta_i=1} \Big[\beta^T m_\theta(x_i) - \log\big(\sum_{j:t_j>t_i} \exp(\beta^T m_\theta(x_j))\big)\Big] \tag{25}$$

We minimize the negative Cox partial log-likelihood by mini-batch gradient descent w.r.t. β. We note that, as this loss is non-separable, its gradient on mini-batches might be degenerate if the sampled batch contains only non-admissible pairs that cannot be ranked together (Jenkins, 2005).

All experiments in the article on the presented datasets are repeated 10 times in order to produce confidence intervals.

E.3. Dirichlet split of the dataset across the clients.

In Section 4.2 and Figure 2, we use the same method as (Hsu et al., 2019) to split the FashionMNIST dataset into different clients, while parameterizing the heterogeneity of the data distribution by a parameter $0 < \alpha < \infty$. A small value of α corresponds to a heterogeneous data distribution, while a high value of α corresponds to a homogeneous data distribution.

More precisely, for a given value of α and a given client k, we draw a random vector $\vec{q}$ following a Dirichlet distribution of parameter α of size $L = 10$, where L is the number of labels present in FashionMNIST. Such a vector satisfies $\|\vec{q}\|_1 = 1$ and $q_l \geqslant 0$ for $l = 1, \dots, L$. The density distribution of $\vec{q}$ is proportional to $\prod_l q_l^\alpha$. For the client k, we then generate a dataset of size $\#\mathcal{D}_k$, such that the number of samples with label l is roughly $\lfloor q_l \#\mathcal{D}_k \rfloor$.

When $\alpha \to \infty$, we have $q_l \to 1/L$ for all l, therefore each client k as $\#\mathcal{D}_k/L$ samples of each label. When $\alpha \to 0$, we have $\vec{q} \to (0., \dots, 0, 1, 0, \dots)$, where the only non-zero component of $\vec{q}$ is randomly one of the L dimensions. In that case, each client has samples coming from only one label.

We provide in Figure 3 an extension of Figure 2 where we added the performance of two extra clustering algorithms: the BIRCH (Zhang et al., 1996) and the Affinity propagation (Frey & Dueck, 2007). We could not empirically make the attack works with other standard clustering algorithms.

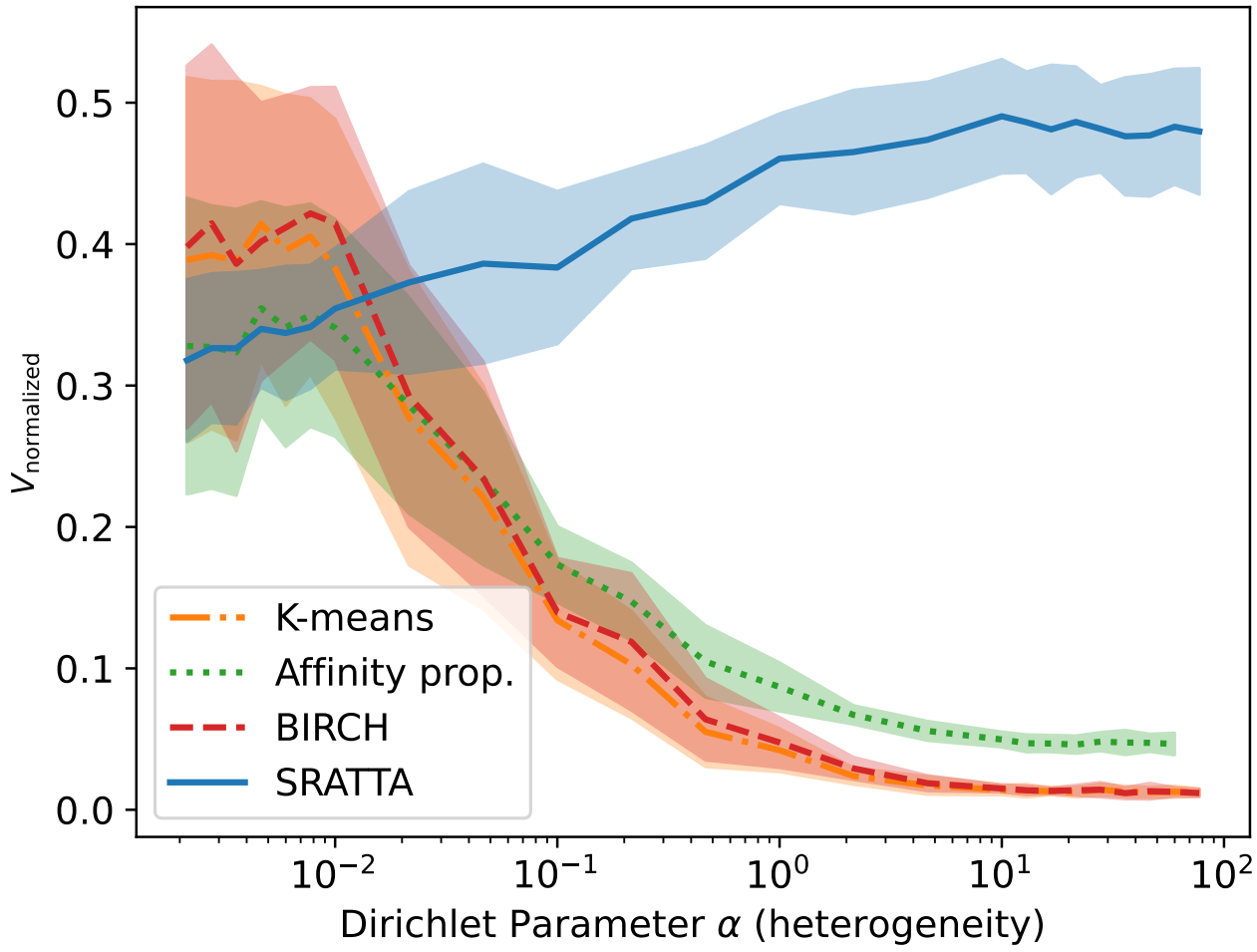


Figure 3. Success rate of SRATTA and K-means, affinity propagation (Frey & Dueck, 2007) and BIRCH (Zhang et al., 1996) as a function of inter-client homogeneity measured by the Dirichlet parameter α. Cf Appendix E.3 and Table 4 for experimental details

E.4. Effect of different hyperparameters on the strength of SRATTA

In this section, we give more details as to the effect of different hyper parameters on the strength of SRATTA. Figures 4 and 5 shows the impact of the batch size, the number of hidden neurons, the learning rate, the number of rounds, the number of trainings and the number of updates per round in the efficiency of SRATTA.

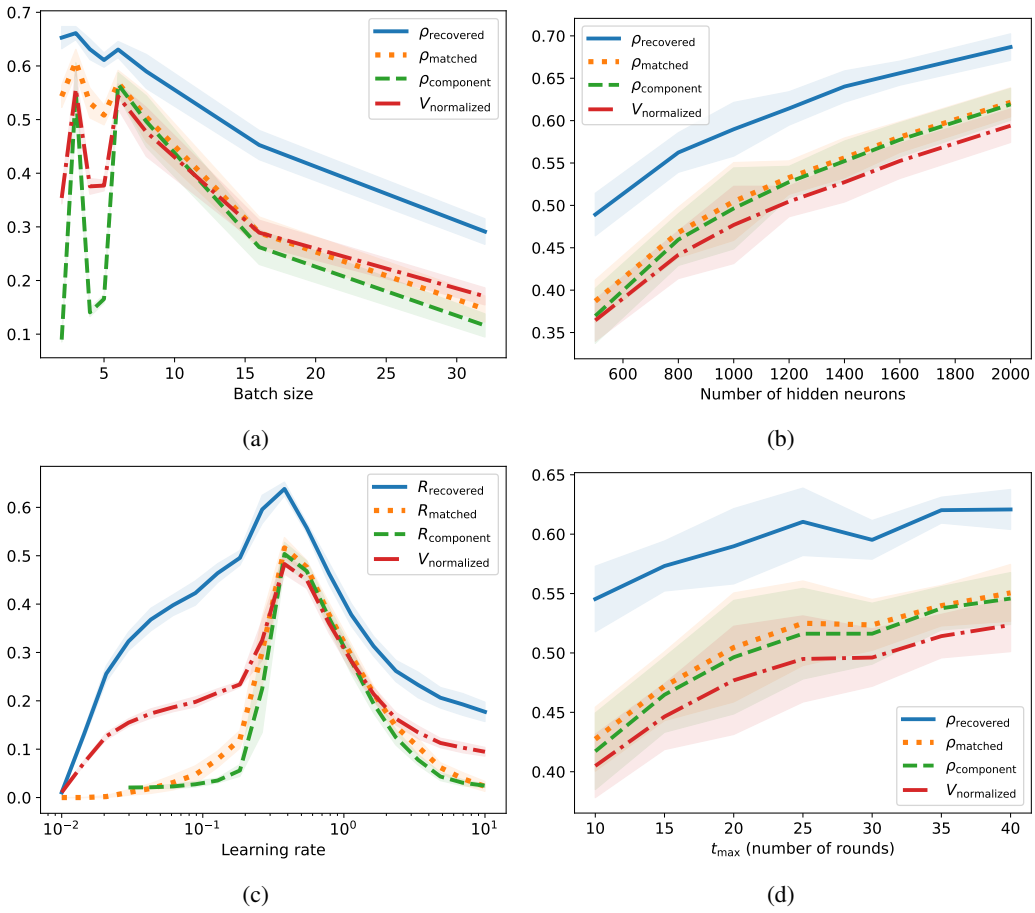


Figure 4. Effects of batch size, number of hidden neurons, learning rate and number of rounds on the performance of SRATTA on FashionMNIST. Hyper-parameters used are details in Table 4.

F. Defending against secure disaggregation

This section is devoted to the defense presented in Section 5, that we will name q-defense. We first state definitions used in the main text then we provide details and extra-results on the q-defense in Appendix F.1. Then, in Appendix F.3, we develop and implement another kind of defense, denoted β-defense, which is similar to the one presented in Appendix F.1, but relies on another way of selecting the neurons to censor.

F.1. Additional definitions for the q-defense

In Section 5, given a client k, we define the individual client activation set of neuron h at round t as

$$\mathcal{A}^h_{t,k} = \bigcup_{i=0}^{n_{\text{updates}}-1} \mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i}). \tag{26}$$

One can write the update of a specific client k as

$$\Delta \overline{W}^h_{t,k} \overset{\text{def}}{=} \overline{W}^h_{t,k,n_{\text{updates}}} - \overline{W}^h_{t-1}. \tag{27}$$

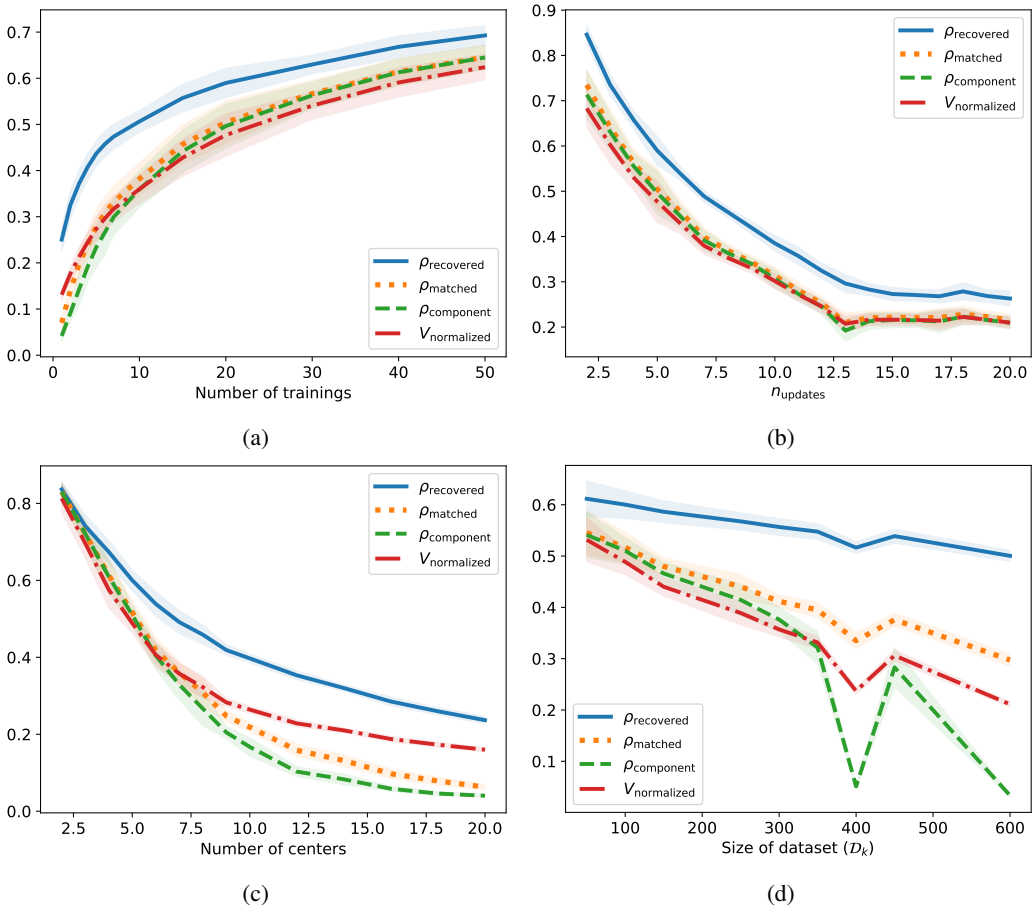


Figure 5. Effects of the number of trainings, number of local updates, number of centers and dataset size on the performance of SRATTA on FashionMNIST. Hyper-parameters used are details in Table 4.

Exactly as is the case for the round-level activation set, the update of a specific client is a linear combination of the elements in its activation set.

Fix a client k, a neuron h, and $t \in \{1, ..., t_{\max}\}$. Let $x_1, ..., x_N$ denote the activation points in $\mathcal{A}^h_{t,k}$. There exists non zero coefficients $(\lambda_r) \in \mathbb{R}\backslash\{0\}^N$ such that

$$\Delta W^h_{t,k} = \sum_{r=1}^{N} \lambda_r x_r \tag{28}$$

$$\Delta b^h_t = \sum_{r=1}^{N} \lambda_r \tag{29}$$

The q-defense, introduced in Section 5, consists in keeping track of the individual client activation set of all neurons, and censoring the update for neurons whose individual client activation set is non-empty and of size inferior to q.

Considering a client k, let fix $q = 1$ and $\tilde{h}$ be such a neuron, i.e:

$$\#\mathcal{A}^{\tilde{h}}_{t,k} = 1. \tag{30}$$

For this neuron, there exists a data sample such that $\{x\} = \mathcal{A}^{\tilde{h}}_{t,k}$, which can be recovered by the first step of the attack, as

from Equation (8):

$$\Delta W^{\tilde{h}}_{t,k} = \lambda x \tag{31a}$$

$$\Delta b^{\tilde{h}}_{t,k} = \lambda \tag{31b}$$

$$\lambda \neq 0 \tag{31c}$$

To avoid this situation, the defense relies on *censoring* this neuron, which means to set to zero the update sent to the server for $\tilde{h}$. We define an integer threshold q, and censor neurons whose individual client activation sets are smaller than q. The resulting algorithm is depicted in Algorithm 4. This defense therefore requires an active role of the client.

Algorithm 4 LocalUpdateDefended // Executed on server k

Require: initial model θ_{t-1}, local dataset $\mathcal{D}_k$, batch size b, learning rate η
  $\theta_{t,k,i=0} \leftarrow \theta_{t-1}$
  for $i = 0$ **to** $n_{\text{updates}} - 1$ **do**
    $B_{t,k,i} \leftarrow$ batch of size b from $\mathcal{D}_k$
    $\theta_{t,k,i+1} \leftarrow \theta_{t,k,i} - \eta \nabla_\theta \mathcal{L}(\theta_{t,k,i}, B_{t,k,i})$
  end for
  for $h = 0$ **to** d **do**
    if $\#\mathcal{A}^h_{t,k} \leqslant q$ **then**
      $\overline{W}^h_{t,k,n_{\text{updates}}} \leftarrow \overline{W}^h_{t-1}$
    end if
  end for
Ensure: $\theta_{t,k,n_{\text{updates}}}$

?? and Table 2 shows the efficiency of the q-defense against the attack. Note that if some samples are still recovered with $q = 1$, using $q = 4$ leads to a perfect defense (no sample recovered) in our experiments.

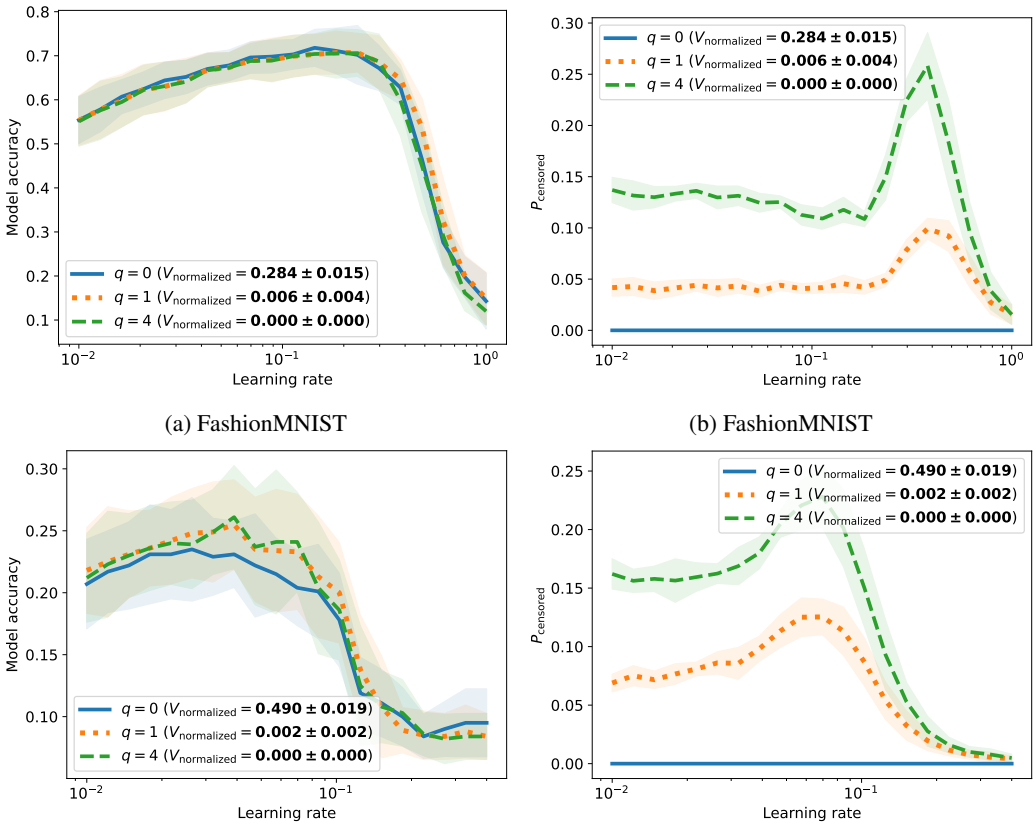
F.2. Additional results for q-defense

Table 2 shows the complete results of the q-defense on the different datasets, and Figure 7d shows the proportion of censored neurons by q-defense for different learning rates.

Dataset	$\rho_{\text{recovered}} \downarrow$	$\rho_{\text{matched}} \downarrow$	$\rho_{\text{component}} \downarrow$	$V_{\text{normalized}} \downarrow$	P_{censored}	Model Acc. $\uparrow$
CIFAR10 ($q = 0$)	0.585 ± 0.015	0.514 ± 0.021	0.511 ± 1.951	0.490 ± 0.019	0.000 ± 0.000	0.261 ± 0.039
CIFAR10 ($q = 1$)	0.002 ± 0.002	0.000 ± 0.001	0.000 ± 0.000	0.002 ± 0.002	0.098 ± 0.011	0.272 ± 0.033
CIFAR10 ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.180 ± 0.014	$\mathbf{0.274 \pm 0.040}$
DNA ($q = 0$)	0.516 ± 0.080	0.031 ± 0.018	0.022 ± 0.296	0.233 ± 0.032	0.000 ± 0.000	0.929 ± 0.017
DNA ($q = 1$)	0.035 ± 0.015	0.000 ± 0.000	0.000 ± 0.000	0.031 ± 0.013	0.005 ± 0.002	0.929 ± 0.019
DNA ($q = 4$)	$\mathbf{0.000 \pm 0.001}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.001}$	0.038 ± 0.006	$\mathbf{0.932 \pm 0.020}$
FashionMNIST ($q = 0$)	0.476 ± 0.027	0.285 ± 0.022	0.230 ± 0.015	0.284 ± 0.015	0.000 ± 0.000	$\mathbf{0.732 \pm 0.053}$
FashionMNIST ($q = 1$)	0.009 ± 0.004	0.001 ± 0.003	0.000 ± 0.000	0.006 ± 0.004	0.042 ± 0.006	0.729 ± 0.043
FashionMNIST ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.149 ± 0.022	0.731 ± 0.046
TCGA ($q = 0$)	0.279 ± 0.016	0.199 ± 0.014	0.175 ± 1.349	0.193 ± 0.010	0.000 ± 0.000	$\mathbf{0.660 \pm 0.065}$
TCGA ($q = 1$)	0.051 ± 0.016	0.024 ± 0.017	3.040 ± 1.158	0.036 ± 0.012	0.135 ± 0.016	0.658 ± 0.071
TCGA ($q = 4$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.257 ± 0.030	0.658 ± 0.071

Table 2. Influence of the q-defense (presented in Section 5 and detailed in Appendix F.1) on both the attack efficiency and the training performance. The attack is performed on a simulation of a grid search, with 20 training with different learning rates. The model accuracy reported is the best one for all the learning rates tested. This setting highlights the defense success to prevent the attack for a range of learning rates, with no-significant loss of model accuracy.

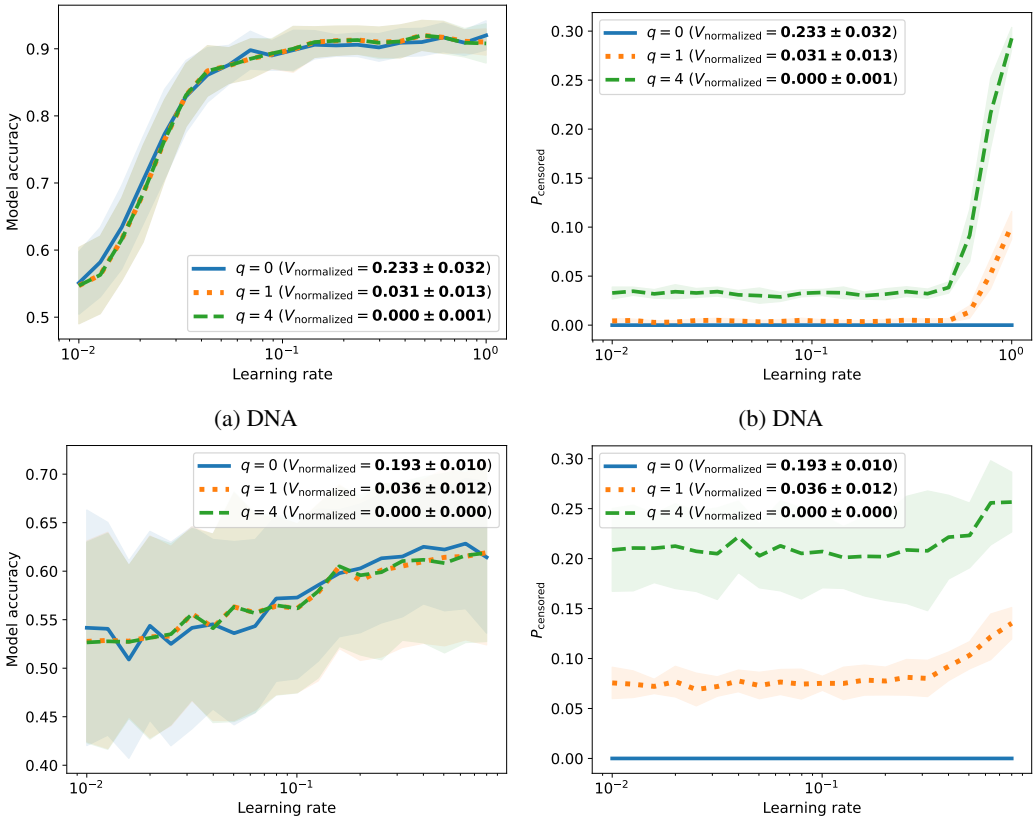
Both show the efficiency of the q-defense against SRATTA. Note that if some samples are still recovered with $q = 1$, using $q = 4$ leads to perfect defense (no sample recovered) in our experiments. In practice, samples can be recovered if $\mathcal{A}^{\tilde{h}}_{t,k}$ is not a singleton, in the case where one of the λ_r in Equation (28) is an order of magnitude larger than the others in absolute value.



(a) FashionMNIST (b) FashionMNIST

(c) CIFAR10 (d) CIFAR10

Figure 6. Effects of the q-defense (presented in Section 5 and detailed in Appendix F.1) on the model accuracy and the number of neurons frozen. P_{censored} corresponds to the proportion of neurons censored compared to the total number of neurons.



(a) DNA (b) DNA

(c) TCGA (d) TCGA

Figure 7. Effects of the q-defense (presented in Section 5 and detailed in Appendix F.1) on the model accuracy and the number of neurons frozen. P_{censored} corresponds to the proportion of neurons censored compared to the total number of neurons.

F.3. Another possible defense: the β-defense

We introduce in this section another defense, named "β-defense". Increasing the value of q is an efficient way to avoid sample recovery. However, it might censor more neurons than needed. Indeed, as mentioned above, if $\#\mathcal{A}^{\tilde{h}}_{t,k} > 1$ for a given neuron $\tilde{h}$, a sample x_r can be recovered if its associated λ_r in Equation (6) is larger than the other λ_j by an order of magnitude. Note that for each neuron h, for each sample x_r involved in the local training, the corresponding λ^h_r can be computed by the client. Following the computation of Appendix A.2, one can derive

$$\Delta W^h_{t,k} = -\eta \sum_{i=0}^{n_{\text{updates}}-1} \sum_{j=0}^{b} \lambda(\theta_{t,k,i}; x_j^{B^i_k}, y_j^{B^i_k}) x_j$$

$$\Delta b^h_{t,k} = -\eta \sum_{i=0}^{n_{\text{updates}}-1} \sum_{j=0}^{b} \lambda(\theta_{t,k,i}; x_j^{B^i_k}, y_j^{B^i_k})$$

with $\lambda(\theta; x, y) = \frac{\partial \mathcal{L}(\theta;x,y)}{\partial z^h} \mathbf{1}_{W^h x + b^h > 0}$. These last quantities are tractable by the client during the local update. We introduce a threshold β and we censor all neurons for which there exists $\tilde{x} \in \bigcup_{i=0}^{n_{\text{updates}}-1} \mathcal{A}^h(\theta_{t,k,i}, B_{t,k,i})$, and its associated $\tilde{\lambda}$ such that:

$$\frac{|\tilde{\lambda}|}{\sum_{i=0}^{n_{\text{updates}}-1} \sum_{j=0}^{b} |\lambda(\theta_{t,k,i}; x_j^{B^i_k}, y_j^{B^i_k})|} \geqslant \beta \tag{32}$$

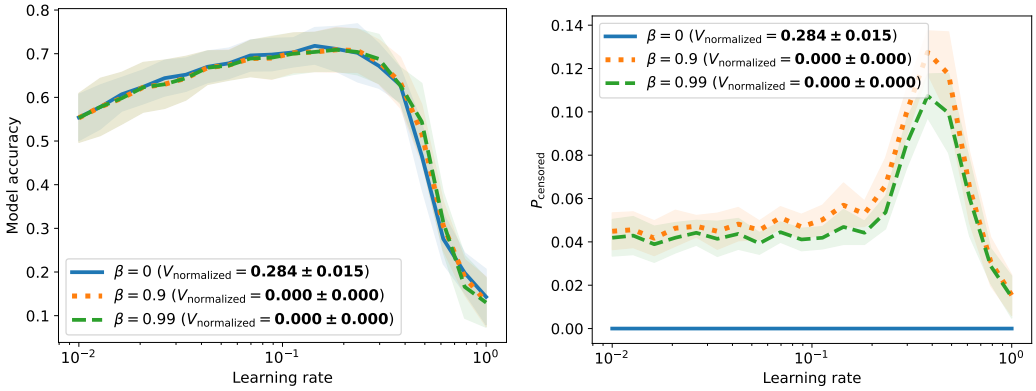
In plain english, if the linear weight of a sample contributes to the update more than a fraction β, we censor the neuron. We name this other defense β-defense, different to the q-defense in the way it selects the neurons to censor. The results exhibited in Figures 6 and 7 and table 3 highlights that we are able while censoring less neurons using the β-defense than when using the q-defense with $q = 4$, we successfully defend ourselves against the attack. Note that in these results the q-defense is never used, as the β-defense comes as a replacement of the q-defense.

Dataset	$\rho_{\text{recovered}} \downarrow$	$\rho_{\text{matched}} \downarrow$	$\rho_{\text{component}} \downarrow$	$V_{\text{normalized}} \downarrow$	P_{censored}	Model Acc. $\uparrow$
CIFAR10 ($\beta = 0.0$)	0.585 ± 0.015	0.514 ± 0.021	0.511 ± 0.020	0.490 ± 0.019	0.000 ± 0.000	0.261 ± 0.039
CIFAR10 ($\beta = 0.9$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.094 ± 0.013	0.270 ± 0.034
CIFAR10 ($\beta = 0.99$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.087 ± 0.014	$\mathbf{0.271 \pm 0.033}$
DNA ($\beta = 0.0$)	0.516 ± 0.080	0.031 ± 0.018	0.023 ± 0.003	0.233 ± 0.032	0.000 ± 0.000	0.929 ± 0.017
DNA ($\beta = 0.9$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.020 ± 0.007	0.931 ± 0.019
DNA ($\beta = 0.99$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.008 ± 0.003	$\mathbf{0.932 \pm 0.021}$
FashionMNIST ($\beta = 0.0$)	0.476 ± 0.027	0.285 ± 0.022	0.230 ± 0.015	0.284 ± 0.015	0.000 ± 0.000	$\mathbf{0.732 \pm 0.053}$
FashionMNIST ($\beta = 0.9$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.053 ± 0.006	0.725 ± 0.042
FashionMNIST ($\beta = 0.99$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.044 ± 0.005	0.727 ± 0.043
TCGA ($\beta = 0.0$)	0.279 ± 0.016	0.199 ± 0.014	0.176 ± 0.013	0.193 ± 0.010	0.000 ± 0.000	$\mathbf{0.660 \pm 0.065}$
TCGA ($\beta = 0.9$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.169 ± 0.032	0.655 ± 0.070
TCGA ($\beta = 0.99$)	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	$\mathbf{0.000 \pm 0.000}$	0.147 ± 0.025	0.658 ± 0.069

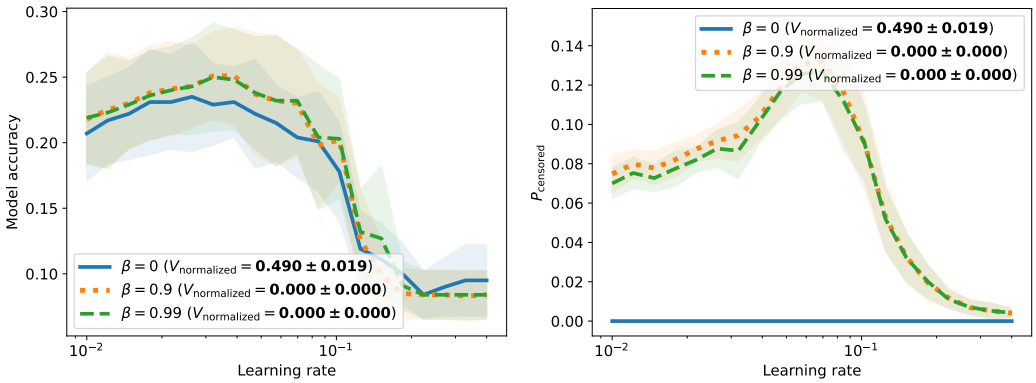
Table 3. Results on the relative defense presenting in Appendix F.3. Influence of the relative defense on both the attack efficiency and the training performance. The attack is performed on a simulation of a grid search, with 20 training with different learning rates. The model accuracy reported is the best one for all the learning rates tested. This setting highlights the defense success to prevent the attack for a range of learning rates, with no-significant loss of model accuracy. This table highlights that the relative defense ($\beta = 0.99$) is censoring less neurons that the absolute defense with $q = 4$ for the same efficiency at defending from the attack ($\rho_{\text{recovered}} = 0$).

F.4. Remark on preserving symmetry with $n_{\text{updates}} = 1$.

The second part of the attack is relevant only if $n_{\text{updates}} > 1$, which is the setting this paper focus on. It could be seen as a possible way of defending against SRATTA. However enforcing $n_{\text{updates}} = 1$ by either (i) considering one batch in each round or (ii) accumulating gradient at the initial point θ_{t-1}, boils down to distributed SGD. This algorithm is not suitable in all FL settings as it comes with high communication costs (McMahan et al., 2017), and is more sensitive to gradient attacks, as studied by other work (Geiping et al., 2020; Huang et al., 2021; Hatamizadeh et al.; Xu et al., 2022). As a result we do not explore this further.

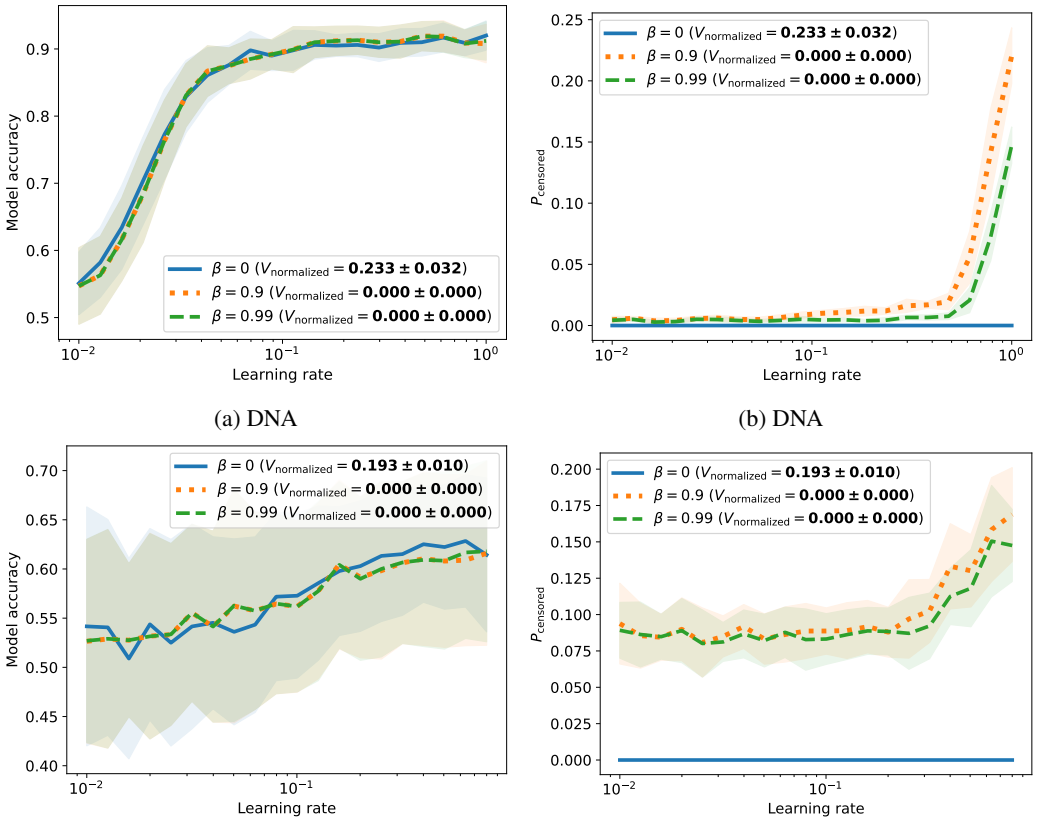


(a) FashionMNIST (b) FashionMNIST



(c) CIFAR10 (d) CIFAR10

Figure 8. Effects of the β-defense on FashionMNIST and CIFAR (presented in Appendix F.3) on the model accuracy and the number of neurons frozen. P_{censored} corresponds to the proportion of neurons censored compared to the total number of neurons.$\beta = 0$ means that no defense is applied, neither β-defense nor q-defense.



(a) DNA (b) DNA

(c) TCGA (d) TCGA

Figure 9. Effects of the β-defense on DNA and TCGA (presented in Appendix F.3) on the model accuracy and the number of neurons frozen. P_{censored} corresponds to the proportion of neurons censored compared to the total number of neurons.$\beta = 0$ means that no defense is applied, neither β-defense nor q-defense.

G. Hyper-parameters used in numerical experiments

Table 4 lists the hyper-parameters used in the different numerical experiments shown in this paper.

Figure	Dataset	# centers	$\#\mathcal{D}_k$	# batch	# hid. neur.	n_{updates}	$t_{\max}$	# trainings	lr
Figure 1	CIFAR10	5	100	8	1000	5	20	20	0.1
Figure 1	FMNIST	5	100	8	1000	5	20	20	0.5
Figure 1	DNA	5	100	8	1000	5	20	20	1.0
Figure 1	TCGA	5	100	8	1000	5	20	20	0.8
Figures 2 and 3	FMNIST	5	100	8	1000	5	20	20	0.5
Figure 4a	FMNIST	5	100	-	1000	5	20	20	0.5
Figure 4b	FMNIST	5	100	8	-	5	20	20	0.5
Figure 4c	FMNIST	5	100	8	1000	5	20	20	-
Figure 4d	FMNIST	5	100	8	1000	5	-	20	0.5
Figure 5a	FMNIST	5	100	8	1000	5	20	-	0.5
Figure 5b	FMNIST	5	100	8	1000	-	20	20	0.5
Figure 5d	FMNIST	5	-	8	1000	5	$0.2\#\mathcal{D}_k$	20	0.5
Figure 5c	FMNIST	-	100	8	1000	5	20	20	0.5
Figures 6a and 6b	FMNIST	5	100	8	1000	5	20	20	-
Figures 8a and 8b	FMNIST	5	100	8	1000	5	20	20	-
Figures 6c and 6d	CIFAR10	5	100	8	1000	5	20	20	-
Figures 8c and 8d	CIFAR10	5	100	8	1000	5	20	20	-
Figures 7a and 7b	DNA	5	100	8	1000	5	20	20	-
Figures 9a and 9b	DNA	5	100	8	1000	5	20	20	-
Figures 7c and 7d	TCGA	5	100	8	1000	5	20	20	-
Figures 9c and 9d	TCGA	5	100	8	1000	5	20	20	-

Table 4. Hyper-parameters used in the numerical experiments.