FUBA: Federated uncovering of backdoor attacks for heterogeneous data

Fabiola Espinoza Castellon, Deepika Singh, Aurélien Mayoue, Cedric Gouy-Pailler

FUBA: Federated Uncovering of Backdoor Attacks for Heterogeneous Data

Fabiola Espinoza Castellon*, Deepika Singh*, Aurélien Mayoue*, Cédric Gouy-Pailler*
fabiola.espinozacastellon@cea.fr
* *Institut LIST, CEA, Universite Paris-Saclay*
F-91120, Palaiseau, France

***Abstract*—This paper proposes a post-training defense against pattern-triggered backdoor attacks in federated learning contexts. This approach relies first on the server estimating the attack pattern. The server then provides the estimated pattern to the end-users, who use it directly on their local data to mitigate backdoor attacks during inference time. This scheme offers an improvement over the existing approaches by demonstrating robustness to data heterogeneity among users without needing a shared dataset or additional information from users and regardless of the number of malicious clients. Based on extensive comparison with existing state-of-the-art methods on well-known computer vision datasets, the proposed method is shown to succeed in mitigating backdoor attacks while preserving high accuracy on clean inputs.**

***Index Terms*—federated learning, backdoor attacks, defense, heterogeneous data**

I. Introduction

Nowadays, machine learning is used in countless applications, owing to the fact that different entities, called clients or participants, generate large amounts of data. This data can be generated by clients subscribed to an IoT service, medical establishments possessing patients' data, autonomous cars, etc. Traditionally, data provided by participants is centralized in a data center to train the machine learning model. However, the data produced could be confidential in nature or sizable. Directly gathering data could thus raise privacy and communication cost issues. To address these issues, [1] proposed a protocol with privacy-preserving properties and efficiency, known as Federated learning (FL). In FL, the goal is to build a global model with various participants who update model's parameters locally, without them sharing their data. A central entity, known as the server, coordinates this process by averaging clients' updates, without seeing clients' data.

However, by preserving privacy and receiving updates blindly from clients, FL is vulnerable to a variety of training-time attacks from certain malicious clients who want to degrade the global model. For instance, byzantine attacks consist in sending altered updates to the server to decrease the overall model performance or to make the training process completely diverge. On the other hand, backdoor attacks (BAs) instill a specific unwanted behavior into the model for some specific data instances while maintaining a satisfying performance on the rest of the instances. These attacks try to embed a behavior in the model that can cause misclassifications to a particular target class t when desired. This is achieved by poisoning part of the training data with a specific pattern and changing the true labels to the target label t. Contrary to byzantine attacks, such targeted attacks are stealthy as the infected model still has normal behavior on clean inputs. Furthermore, they are effective because an input with the trigger is predicted as the target label with high probability. These two properties make the BAs threatening and hard to detect.

In this study, we propose **FUBA**, a new **F**ederated defense to **U**ncover **B**ackdoor **A**ttacks. We argue that the strength of such attacks lays them open to disclosure. An attacked model can be considered as overfitted for a specific class (the target label t) on data poisoned by a specific pattern. A direct correlation is created between the attacked class and the hidden pattern, which makes the uncovering of the attack pattern possible. Our goal is thus to reveal the hidden pattern and use it as a shield against poisoned models. More precisely, the main steps of our method are to: (1) retrieve noised estimations of the attack trigger without using clean data, (2) define a final patch for the defense and (3) distribute the final patch as a bandage to the end-users who will deploy it during inference. Our method defends against BAs without requiring strong assumptions.

Our main contributions rely on proposing a new defense, well-designed for FL scenarios:

- Unaffected by data distribution: most defenses require data distribution to be homogeneous among clients, which is not the case in our protocol.
- Unaffected by the number of attackers: most defenses require a majority of honest clients, which is not the case in our protocol.
- Built at the server side, without accessing significant information from clients, such as common clean data.

Further, we lead extensive experiments to validate the efficiency of our method on multiple computer vision datasets.

II. Related work

The FL algorithm `FedAvg`, formalized by [1], consists of a succession of rounds coordinated by a central server. At each round t, the server sends a global model w_t to the K clients. According to their availability and computational capacity, a subset C_t of $C \cdot K \leq K$ clients participate at round t. Participants learn a personal task on their devices. More specifically, client k uses its dataset to minimize a proper objective function f_k with stochastic gradient descent (SGD), a number of epochs E and a mini-batch size B.

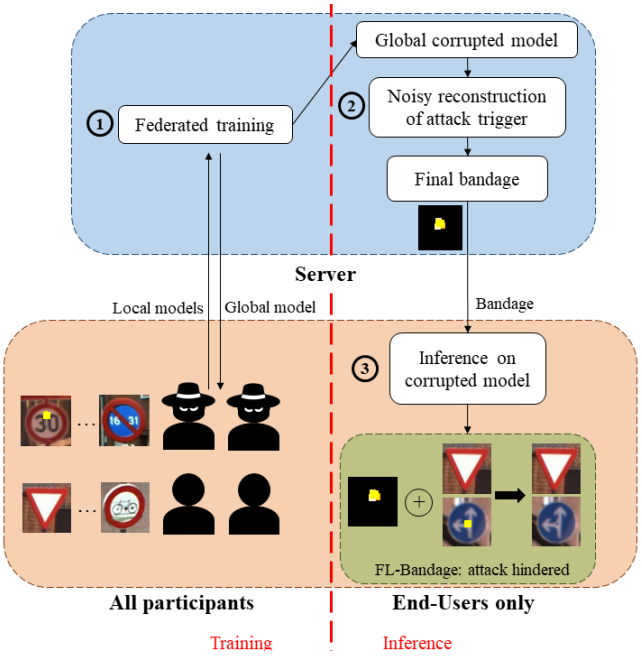


Fig. 1: Overview of FUBA.

Each client k then sends updates w_t^k to the server who averages the received parameters to obtain a new global model $w_{t+1} = \sum_{k \in C_t} \frac{n_k}{N} w_t^k$, where n_k is the number of samples of client k and $N = \sum_{q \in C_t} n_q$. The global objective function f is formulated as the sum of the local objective functions f_k weighted by the proportion of each client k's samples, $p_k = \frac{n_k}{N}$:

$$\min_w f(w) = \min_w \sum_{k=1}^{K} p_k f_k(w) \tag{1}$$

FL is inherently vulnerable to attacks because the remote server has no control over the updates it receives from participants nor over the data participants use to train their models. In this study, we focus on backdoor attacks (BAs) that were introduced by Gu et al. [2] in a centralized setting. Further works [3], [4] have shown that these attacks can also be successful in FL. If a certain number of malicious clients m train their model with a certain proportion p of their own local data poisoned by a given pattern, the BA still persists in the global model, despite averaging loyal and malicious clients' weights.

In FL, most defenses [3], [5]–[7] change the aggregation rule on the server-side to ignore or change potential malicious contributions. Two well-known defenses in FL are Krum [5] and the coordinate-wise median (Med) aggregation [6]. However, these defenses assume that the data distribution among clients is independent and identically distributed (IID) and are robust only when the number of attackers is less than 50%. In practice, clients in FL often have different data distributions [8] due to factors such as personal preferences, geographical location, and device capabilities. Additionally, the server has limited control over the number of potentially

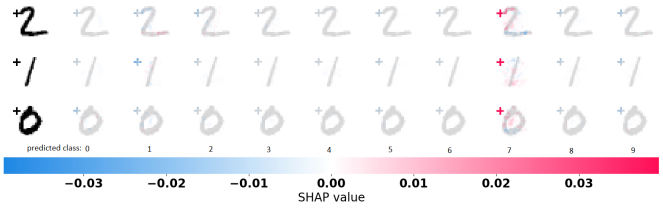


Fig. 2: Shapley values quantify the contribution brought by each pixel to the prediction made by the corrupted model. The cross pattern positively contributes to the prediction of the targeted class 7 and weights negatively towards the prediction of the other classes.

malicious clients. Therefore, it is crucial for a defense to take into account these challenges. Sun et al. [3] and Ozdayi et al. [7] propose defenses that prove to work in such cases since they were applied to non-IID datasets. Sun et al. [3] apply weak differential privacy (wDP) to the federated model to decrease BAs: weights are clipped and noised before averaging. Ozadayi et al. [7] propose Robust Learning Rate (RLR), which carefully adjusts the aggregation server learning rate to mitigate updates that could be harmful to the global model. However, when data is strongly heterogeneous among clients (label skew, see Section IV) these defenses are less efficient. This deterioration can be due to updates that diverge without necessarily being malicious.

In a centralized setting, Neural Cleanse [9] defends from BAs by reconstructing possible triggers, independently from the data distribution or the strength of the attack. However, the method proposed in Neural Cleanse [9] requires access to a clean training dataset. We consider that in FL, the server can not have access to training data since one of the assets of FL is to preserve privacy. In an FL setting, Zhao et al. [10] propose FedReverse (FedR) to reconstruct possible triggers. The trigger reconstruction is performed by clients, who then send their results to the server. However, once again the server is blind to the reliability of the information it receives from the clients. Its defense can be put to failure if the majority of client sends wrong information. Wu et al. [11] propose a defense that prunes "dormant" neurons (FedPrun), which are low activation neurons that presumably encode the backdoor attack. The server prunes neurons through a voting process based on information sent by clients. The defense is thus still dependent on clients, who can send erroneous information to weaken the defense.

Overall, existing defenses have three main weaknesses (see Table I): (1) some are not effective on non-IID data, which is a natural situation in FL, (2) some are not effective if more than 50% of the clients are malicious and (3) some rely strongly on information sent by clients, who can corrupt this information to counter the defense. We propose a defense that addresses these three challenges.

	Krum	Med	RLR	wDP	FedR	FedPrun	Ours
1	✗	✗	✗	✗	✓	✓	✓
2	✗	✗	✗	✓	✗	✗	✓
3	✓	✓	✓	✓	✗	✗	✓

TABLE I: State-of-the-art defenses on three criteria: (1) effective on non-IID data, (2) effective on more than 50% of malicious clients, and (3) low reliability on clients (which can be malicious).

III. Proposed method

A. Overview

We would like to clarify that we tackle attacks in which all malicious users add the *same* trigger to their data. We do not treat semantic attacks [12] nor distributed attacks [13]. Moreover, our attack is targeted: every poisoned sample is labeled as the same targeted class t, regardless of its true label. Malicious clients poise their input data x through a pattern $\mathbf{T}$, a mask M and function g:

$$g(x, M, \mathbf{T})_{i,j} = (1 - M_{i,j}) \cdot x_{i,j} + M_{i,j} \cdot \mathbf{T}_{i,j} \qquad (2)$$

If $\mathbf{T}$ is added to an input sample, then its class is set to the target class t.

Fig 1 illustrates the overall procedure of our method. First, clients participate in a standard federated training process (Fig 1 (1)). Some clients can be malicious and send models trained on corrupted data. The global resulting model will thus be corrupted. After the federated learning process has converged and end-users want to use the global model for inference, the server takes on the role of the defender to warn of the potential dangers of the global model. To uncover the hidden attack trigger, the server computes a noisy estimate of it by inverting the global corrupted model (Fig 1 (2)). Then, it decides on a final defense patch and sends it to the end-users who will use the patch during inference to blur the area where an attack could be present (Fig 1 (3)).

B. Poisoned model

Consider a classification task on a shared dataset $D = \{z_0, z_1, ..., z_N\} = \{(x_0, y_0), (x_1, y_1), ..., (x_N, y_N)\}$ where x and y represent the input samples and the true labels of the samples respectively. Eq. 1 also depends on the inputs of the network. The global objective function can be rewritten as the empirical risk over the clients' local datasets D_k:

$$\tilde{f}(z; w) = \sum_{k=1}^{K} p_k \mathbb{E}_{(z)\sim D_k}[f_k(z; w)] \qquad (3)$$

To successfully attack a model, a malicious client has to optimize the model's weights such that when an input has a backdoor trigger, the loss function is low for the target label. Particularly, if a client k is malicious, then its dataset D_k is the union of clean data and poisoned data $D_k = D_k^c \cup D_k^b$ and its local objective is more precisely:

$$\tilde{f}_k(z; w) = \mathbb{E}_{(z)\sim D_k^c}[f_k(z; w)] + \mathbb{E}_{(x,y)\sim D_k^b}[f_k(x, y = t; w)]$$

The global objective function has thus two supports:

Algorithm 1 Model inversion `MI`

Input: Corrupted weights w

Parameters: Number of samples for optimization n, dimension d of the inputs, set of possible classes $\mathcal{T}$, loss function $\mathcal{L}$, defense learning rate α_d, number of iterations $\mathcal{I}$

for $y \in \mathcal{T}$ **do**
  Let $x_y \sim U(0, 1)$, where $x_y \in \mathbb{R}^{n \times d}$
  for i **in** $1, ..., \mathcal{I}$ **do**
    $x_y \leftarrow x_y - \alpha_d \frac{\partial \mathcal{L}(x_y, y; w)}{\partial x_y}$
  end for
end for
return $\{x_y\}_{y \in \mathcal{T}}$

$$\tilde{f}(z; w) = \sum_{k=1}^{K} p_k \mathbb{E}_{(z)\sim D_k^c}[f_k(z; w)] + \sum_{k \in C_m} p_k \mathbb{E}_{(x,y)\sim D_k^b}[f_k(x, y = t; w)] \qquad (4)$$

where C_m is the subset of malicious clients. Note that every client has part of its objective function over clean data because every client, even if malicious, has clean data.

If the data is heterogenous between clients, the loss function is to be summed over more distributions than in Eq. 4, which is here used simply for explanation purposes.

C. Noisy reconstruction of the attack trigger

As the defender, the server's goal is to uncover the attack. It uses the strength of BAs as vulnerabilities. More precisely, BAs have some characteristics:

- When malicious clients add the trigger to samples, they consistently change their class to the target label t. Thus, the trigger pattern is positively correlated to the attacked class by the corrupted model, but also negatively correlated to all other classes. This is illustrated by Shapley values [14] in Fig 2.
- During FL training, the loss function of the poisoned data and target label t converges to lower values than the loss function of clean data and labels (Fig 3 top).
- An attacked model is very confident of its predictions as the target label t on poisoned data (Fig 3 bottom).

To approximate the trigger pattern, NeuralCleanse [9] optimizes an objective function, in our case (Eq. 4), for which the input is $g(x, M, \mathbf{T})$ (defined in Eq. 2). The optimization is done on $(M, \mathbf{T})$ and requires access to the clean distribution $\bigcup_{k\in[1,K]} D_k^c$. However, in FL the server does not have access to clean data, and in most cases, it does not have access to a shared dataset either. To relax this assumption, our goal is to find an input $\mathbf{z}_\tau$ that has a low minima on the global objective function Eq. 4 and on label τ, which we want to be the target label t. Similarly to [15], the server has to inverse the corrupted model and will optimize.

$$\min_{\mathbf{z}_\tau} \tilde{f}(\mathbf{z}_\tau, t; w) \qquad (5)$$

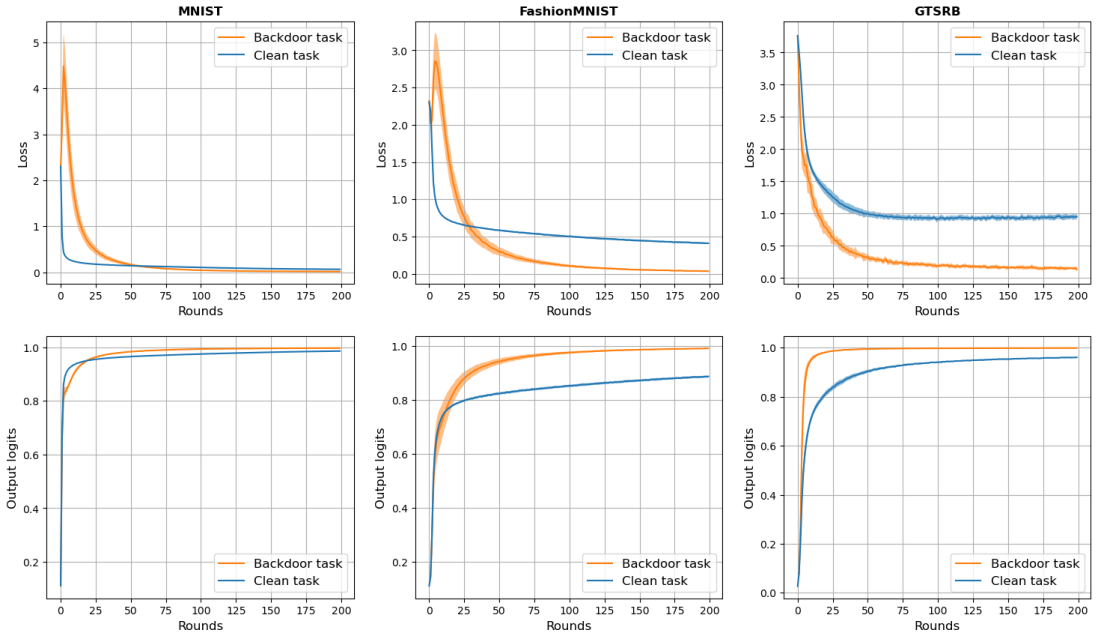


Fig. 3: **(Top)** Loss values during FL training. **(Bottom)** Pseudo probabilities (output logits) on test data during FL training. Only correct predictions were reported. The backdoor task considers a prediction as correct if it is for the target label t on poisoned data. Fig shows mean $\pm$ standard deviation of all experiments. Each experiment has a different class as attack target t so that all classes were at attack targets at least once. Experiments on cross attack for datasets MNIST and Fashion MNIST and on yellow sticker attack for GTSRB (see Section IV).

$\hat{\mathbf{T}}$, the estimation of the trigger $\mathbf{T}$, is obtained through cleaning estimations $\mathbf{z}_\tau$ (see Section III-D): $\hat{\mathbf{T}} = \texttt{clean}((\mathbf{z}_\tau)_{\tau \in \mathcal{T}})$, where $\mathcal{T}$ is the set of possible targets.

To optimize Eq. 5, the server thus needs white-box access to the federated model, but also to the target label t. However, as a defender, it does not have access to the target class t. To circumvent these obstacles and optimize Eq.5, the server will nevertheless use the method proposed in Algorithm 1:

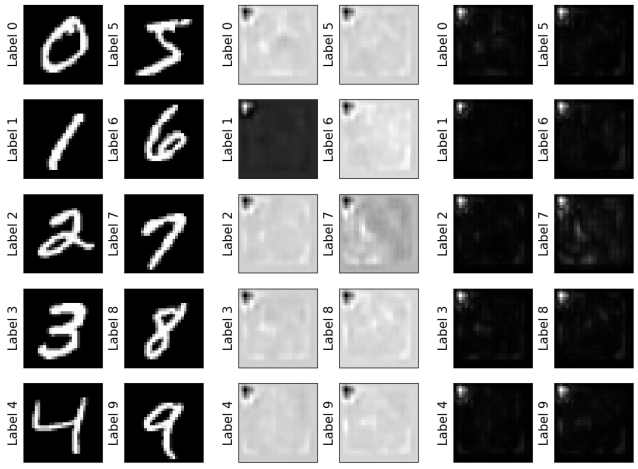
Inputs for the defense objective: To minimize Eq. 5, we sample n random data samples from a uniform distribution (l.2 Alg 1) of the same dimension d as the global model's input. We use SGD (l.4 Alg 1) as the optimizer. Each random sample is treated separately (batch size of 1). The server therefore needs to know the dimension of the input data (d in Alg 1 that is the size of the input layer) and has to define a loss function ($\mathcal{L}$ in Alg 1). In our case, we use the discrete cross-entropy function.

Target class for the defense objective: The server does not know the attacked target t. However, it knows the set of possible targets $\mathcal{T}$ (size of the output layer). It can thus run a few SGD steps of Eq. 5 with random data for *each* class. Two main observations can be made from the reconstructions generated by the server for each class. Firstly, we notice that the class with the lowest loss function value is generally the target label t. This is coherent with the fact that BAs cause the loss function to be remarkably low on the attacked label (Fig 3). Moreover, after these few steps of SGD for Eq. 5, the optimized input for label t is a fairly detectable replicate of the true attack pattern. The shape and edges of the pattern are inexact, but the position and pixel intensity give an idea of the attack that was applied (Fig 4 and 5). However, the specific features of the true attacked class (class 1 in Fig 4 and class 5 in Fig 5) are not distinguishable. This can be explained by the fact that during federated training, since class t is attacked, inputs of many classes are labeled as t. The model thus relates features of various classes with class t. Furthermore, we notice that the reconstructions of other classes also provide information about the attack as previously shown in Fig 2. Since the backdoor pattern is inserted into many classes, it also affects the reconstruction of other classes. When malicious clients train models with clean and poisoned samples of the same label, the models process samples that differ significantly only on the attack pattern but have different labels. Thus, to counter the effect of the attack, the inversion of the model produces opposite values from the attack trigger for classes other than the target t.

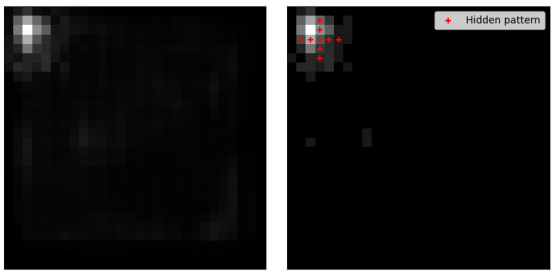
Consequently, noisy estimations of the attack pattern are detectable in the reconstructed inputs of classes other than the target class t (Fig 4 and 5). These estimations are opposite to the estimation of t (label 1 in Fig 4 and label 5 in Fig 5), and therefore to the original trigger. They are also noisier than the one of the attacked class t.

D. Final defense bandage: cleaning the reconstructions

The defender can thus compute the $card(\mathcal{T})$ reconstructions of all possible classes (output of Alg 1). Each reconstruction of class τ, that we name $\mathbf{z}_\tau \in \mathbb{R}^{n \times d}$, can be more or less

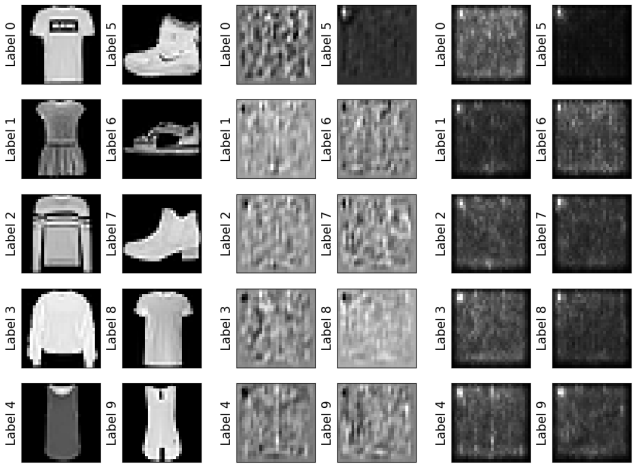


(a) True examples of classes (b) Noisy reconstruction (c) Centered and absolute values

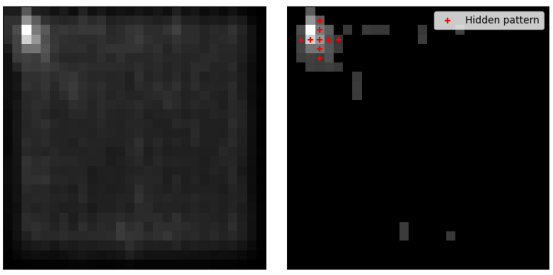


(d) Common reconstruction (e) γ-clean reconstruction

Fig. 4: Example of the different steps of the estimation of the hidden trigger. The dataset is MNIST, the attack pattern is a cross (Fig 7b) and the target label t = 1. Here, γ is set to 0.05.



(a) True examples of classes (b) Noisy reconstruction (c) Centered and absolute values



(d) Common reconstruction (e) γ-clean reconstruction

Fig. 5: Example of the different steps of the estimation of the hidden trigger. The dataset is FashionMNIST, the attack pattern is a cross (Fig 7b) and the target label t = 5. Here, γ is set to 0.05.

noisy (Fig 4b, Fig 5b). According to the class (target t or not), the values of our zone of interest can be opposite to the true pattern, but they represent outlier values within their reconstructions. Moreover, since the attack generally concerns a limited subset of the input, we want to keep minimal information obtained in $\mathbf{z}_\tau$. We therefore only keep outlier values that reflect the effect of the hidden pattern.

We consider the average of the n random optimized samples to get a reconstruction per class (Fig 4b, Fig 5b). To harmonize the reconstructions' distributions and bring out outlier values, assuming they are normal, we centralize and consider their absolute values (Fig 4c, Fig 5c). To assemble our reconstructions of all classes, we average them (Fig 4d, Fig 5d) and decide to put more weight on the reconstruction associated with the lowest loss function because it is generally associated with the loss function of target label t and is thus less noisy (Section III-C). We obtain a common reconstruction $\bar{\mathbf{z}}$:

$$\bar{\mathbf{z}} = \frac{1}{2}\tilde{p}_{t_{min}} + \sum_{\substack{\tau \in \mathcal{T} \\ \tau \neq t'}} \frac{1}{2(card(\mathcal{T})-1)}\tilde{p}_\tau \tag{6}$$

where $\tilde{p}_\tau = (\tilde{p}_\tau)_{j\in[1,..,d]}$ and

$$(\tilde{p}_\tau)_j = \left| \frac{1}{n}\sum_{i=1}^{n}\left((\mathbf{z}_\tau)_{i,j} - \frac{1}{d}\sum_{j=1}^{d}(\mathbf{z}_\tau)_{i,j} \right) \right| \tag{7}$$

Next, we want to clean our reconstruction and only retain the proportion γ of highest values (Fig 4e, Fig 5e). Let σ be a permutation that sorts $(\bar{\mathbf{z}}_i)_{i\in[1,...,d]}$ in the ascending order: $\bar{\mathbf{z}}_{\sigma(1)} \leq \bar{\mathbf{z}}_{\sigma(2)} \leq ... \leq \bar{\mathbf{z}}_{\sigma(d)}$. We choose $\bar{\mathbf{z}}_{\sigma(k)}$ such that such that $k \geq d(1-\gamma)$. Our cleaned final estimation of the trigger $\hat{\mathbf{T}}$ is $\texttt{clean}(\bar{\mathbf{z}}) \in \mathbb{R}^d$:

$$\texttt{clean}(\bar{\mathbf{z}})_i = \begin{cases} 1 & \text{if } \bar{\mathbf{z}}_i \geq \bar{\mathbf{z}}_{\sigma(k)} \\ 0 & \text{else} \end{cases} \tag{8}$$

$\texttt{clean}(\bar{\mathbf{z}})$ should be a fair reconstruction of the attacked target: in Fig 4e and Fig 5e, our final bandage approximates the cross pattern attack (in red).

Finally, after training, the server sends to the participants the trained global model and the final estimation of the attack trigger $\hat{\mathbf{T}}$. The participants use $\hat{\mathbf{T}}$ during inference to hinder the effect of the attack. To predict the class of an input $x =$

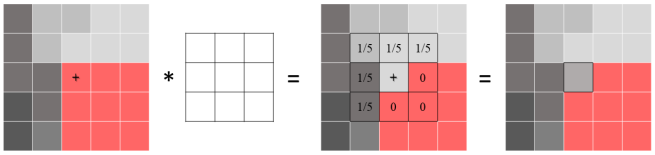


Fig. 6: `KernelMean`: We want to replace the threatening values (in red) with the mean of the neighbor values. A (3×3) kernel is applied and the treat value (marked with a cross +) is changed.

(a) Square pattern attack (b) Cross pattern attack (c) Copyright pattern attack (d) Yellow post-it attack

Fig. 7: Attack patterns on MNIST dataset (a)-(c) and GTSRB dataset (d)

$(x_i)_{i\in[1,\dots,d]}$, instead of directly feeding it to the model, clients change x_i to be:

$$x_i = \begin{cases} x_i & \text{if } \hat{\mathbf{T}}_i = 0 \\ \texttt{KernelMean}(x_i) & \text{else} \end{cases} \tag{9}$$

We consider that the values where $\hat{\mathbf{T}}$ is different than zero are threatening values, that potentially contain the attack trigger. `KernelMean` (Fig 6) is used to mitigate the effect of this threatening zone by replacing the threatening values shown in the bandage with the mean of the surrounding benign values. A convolution between each threatening value and a (3×3) kernel is applied. The kernel will be null for the other threatening values, and sum to 1 for other values. This convolution is done for all threatening values that have at least one benign neighbor value. If a threat value does not have any benign neighbors, the input will be once more scanned, and the threat values that were previously replaced by `KernelMean` will then be benign.

The backdoor pattern is herewith erased from the input. The clients do not know a priori if an input is poisoned, so they will blur the bandage from all inputs they want to infer, whether they are truly harmless or poisoned.

IV. Experiments

An implementation of the method in Pytorch is available as supplementary material. Code can be found at https://github.com/fabiola-espinoza-castellon/Federated_Uncovering_of_Backdoor_Attacks.

A. Model and datasets

As in [7], we use a 5-layer convolutional neural network. It is composed of two convolutional layers of 32 and 64 filters respectively, followed by a max-pooling layer and two fully connected layers with dropout.

We use various datasets of computer vision classification tasks: EMNIST [16] which is the extended version of MNIST [17], FashionMNIST [18], and finally GTSRB (German Traffic Sign Recognition Benchmark) [19] used for digits, clothes (10 classes) and German traffic signs (43 classes) classification tasks. FashionMNIST is a classification task dataset for clothes but remains very similar to MNIST because they have the same number of classes and the same dimensions. EMNIST also shares the same image structure and parameters as the original MNIST, but involves digits and letters as well. EMNIST has more samples than its original version, and also contains the meta-information of the author of each sample. GTSRB is a classification task dataset for traffic signs used in Germany. It contains 43 classes. For coherence, we resize all images to shape $(3 \times 32 \times 32)$.

To simulate a federated context, we split all samples into shards representing clients. According to how we assign samples to each client, we can simulate two cases:

- **IID case**: we split all samples into balanced shards: as far as possible, each client has the same number of samples and classes. We use EMNIST on this case, and equally split samples between users, ignoring the meta-information about the author of each sample. We refer to this decomposition as MNIST in the results Section IV-E.
- **Non-IID case**: we simulate a quantifiable and strong form of heterogeneous distributions among clients that is the *label skew* case [8]. Clients have the same number of samples, however, they only have a certain number s of classes in their local datasets. For instance, for a toy example of only 2 clients with the MNIST dataset, if s is equal to 5, then the first client will have only 5 labels ($0-4$ for instance) and the second client will also have only 5 labels ($5-9$).
  Within the non-IID case, we also include experiments with the raw EMNIST dataset. This dataset contains the information of the writers of each sample and it is assumed to be intrinsically non-IID because each client likely has a different distribution over digits or letters. In our experiments, we use the digits extension and refer to it as FEMNIST.

We split MNIST and FashionMNIST into 100 clients. GTSRB has fewer samples per class and is less balanced than the other datasets, so we split it into 10 users. FEMNIST contains data written by over 3000 writers. For briefness, we only keep the 400 writers with the most samples.

We recap the details of the datasets we used for our experiments in Table III.

B. Federated learning parameters

We fix some federated parameters so that the BA is successful enough. Varying these parameters can have an effect on the backdoor task but due to space constraints, we only show concise results. We set the number of federated training rounds to 200. To minimize the variance in the results, all the clients participate during training i.e. the proportion of

Dataset	Pattern	Before defense		With defense		
		Clean accuracy	Backdoor success	Clean accuracy	Backdoor success	poisoned accuracy
MNIST	Square	0.98 ± 0.8	0.99 ± 0.3	0.98 ± 1.5	0.10 ± 1.2	0.98 ± 1.4
	Cross	0.98 ± 0.9	0.99 ± 0.8	0.98 ± 2.4	0.10 ± 2.7	0.98 ± 2.7
	Copyright	0.98 ± 1.0	0.99 ± 1.0	0.98 ± 1.3	0.15 ± 67	0.88 ± 70
FashionMNIST	Square	0.86 ± 2.5	0.99 ± 1.1	0.85 ± 4.9	0.10 ± 6.0	0.85 ± 4.0
	Cross	0.86 ± 3.6	0.99 ± 2.1	0.85 ± 4.0	0.13 ± 56	0.83 ± 41
	Copyright	0.86 ± 2.1	0.93 ± 11	0.85 ± 3.5	0.22 ± 83	0.74 ± 65
GTSRB	Yellow post-it	0.86 ± 4.4	0.98 ± 2.4	0.83 ± 5.7	0.02 ± 19	0.83 ± 5.5

TABLE II: **IID case**: mean results $\pm$ standard deviation of experiments (std). The middle column shows the best results obtained before the defense.The right column shows results after the defense during inference. Std are multiplied by 10^3.

Dataset	# of labels	Input dimension	# of users	Train samples per user
MNIST	10	$1 \times 28 \times 28$	100	2400
Fashion-MNIST	10	$1 \times 28 \times 28$	100	600
FEMNIST	10	$1 \times 28 \times 28$	400	≈ 82
GTSRB	43	$1 \times 32 \times 32$	10	2664

TABLE III: Dataset details

clients participating in each round C is equal to 1. Each client locally trains its model for $E = 5$ epochs, with a batch size of $B = 50$ and a learning rate of 0.01. For dataset GTSRB, $B = 25$ because the number of samples per client is lower.

C. Backdoor attack

For MNIST and FashionMNIST datasets, $m = 20$ clients out of $K = 100$ are set as malicious. For FEMNIST, $m = 80$ clients out of $K = 400$ are malicious and for GTSRB, only $m = 2$ clients are malicious out of $K = 10$. Malicious clients all corrupt $p = 50\%$ of their local data. We corrupt data with four backdoor patterns (Fig 7), some already used in [7]. Patterns (a), (b), and (c) are applied to MNIST, FashionMNIST and FEMNIST, and pattern (d) to GTSRB. To verify the robustness of our method across all target classes, we perform each experiment setting each class as the target t, and then average the results across all experiments.

D. Defense details and used metrics

For most reconstructions of the attack patterns, we define the number of iterations in Alg 1 to $\mathcal{I} = 2$ and γ to 0.1. The copyright pattern is larger than others and harder to detect because of the complexity of its shape. We thus set γ to 0.3 and a (5×5) kernel in `KernelMeans` for these experiences. Moreover, we set γ to 0.01 for GTSRB because the dataset contains more fine-grained details that can be distorted with `KernelMeans`. Finally, $\alpha_d = 0.1$ and $n = 50$. Increasing the number of samples does not necessarily improve the quality of the reconstructed pattern, and taking a small number of samples n speeds up the computing time of FUBA.

It should be noted that 5% of our experiments did not relate the target class t to the class with the lowest loss function (mainly for FEMNIST, in which samples are not balanced between users). If a class is over represented it can have a greater impact on the minimization of the loss function than the attacked class t and the trigger. Yet, although a higher weight is given to a class different from t in these cases (Eq. 6), the quality of the defense is not affected.

We use the following metrics for evaluation:

- *Clean accuracy*: simply the accuracy of the attacked network on clean samples.
- *Backdoor attack success*: the fraction of samples predicted as the target label t in a poisoned test set. The closer it is to 1, the more efficient the attacks is.
- *Poisoned accuracy*: After the defense, the fraction of poisoned samples predicted correctly as their original class.

We provide overall performances: the test set used for evaluation is the union of all clients' test sets.

E. Results and discussion

Table II and IV sum up the results of over 250 experiments of our method and indicate that it protects against BAs: during inference, the BA success is weakened, and the model correctly predicts clean and even poisoned samples. As expected, for non-IID label skew cases, general performances even before the defense are notably lower than for IID cases. However, we do not focus on attaining state-of-the-art performances for all mentioned datasets, but rather on reducing the attack. The important contribution that our method brings is that it can be applied even in scenarios of heterogeneous data among users, which can arise in real-world applications [8].

Comparison with existing state-of-the-art defenses:

We compare our method with famous defenses RLR [7], wDP [3] and Med [6] (Table V). Med, similarly to Krum [5], is a robust aggregator that ignores outlier updates. RLR is a famous defense that also ignores potential harmful updates, and that was tested on FEMNIST dataset (non-IID). Finally, wDP [3] proved that adding noise and clipping updates could hinder the attacks. RLR and wDP require setting hyperparameters: for RLR, the threshold value to discard clients' updates who are probable outliers and for wDP the norm bound M and the variance σ of the added Gaussian noise. We ran various experiments defending from the square attack by the previously mentioned methods. Table V reports the best compromise we found through 200 federated rounds between robustness to BAs and correct clean input performance while tuning the methods' hyperparameters. A defense is efficient if the backdoor success drops *and* the clean accuracy remains high.

Dataset	Pattern	s	Before defense		With defense		
			Clean accuracy	Backdoor success	Clean accuracy	Backdoor success	Poisoned accuracy
MNIST	Square	1	0.84 ± 0.27	0.99 ± 0.00	0.82 ± 2.20	0.11 ± 3.10	0.82 ± 2.20
		5	0.94 ± 0.51	0.99 ± 0.03	0.93 ± 0.51	0.10 ± 0.25	0.93 ± 0.54
	Cross	1	0.84 ± 0.35	0.99 ± 0.01	0.82 ± 0.95	0.11 ± 1.60	0.82 ± 0.95
		5	0.93 ± 0.43	0.99 ± 0.12	0.93 ± 0.57	0.10 ± 0.20	0.93 ± 0.63
	Copyright	1	0.82 ± 0.60	0.99 ± 0.00	0.75 ± 2.85	0.21 ± 6.10	0.67 ± 4.90
		5	0.94 ± 0.50	0.99 ± 0.08	0.93 ± 0.60	0.13 ± 7.93	0.85 ± 6.85
FashionMNIST	Square	1	0.69 ± 0.64	0.99 ± 0.01	0.66 ± 1.48	0.12 ± 2.22	0.67 ± 0.82
		5	0.81 ± 0.45	0.99 ± 0.15	0.79 ± 0.37	0.10 ± 0.91	0.80 ± 0.52
	Cross	1	0.69 ± 0.60	0.99 ± 0.04	0.64 ± 1.94	0.17 ± 3.63	0.64 ± 1.94
		5	0.80 ± 0.56	0.99 ± 0.33	0.78 ± 1.20	0.10 ± 0.99	0.78 ± 1.24
	Copyright	1	0.65 ± 0.77	0.98 ± 0.44	0.63 ± 1.50	0.18 ± 11.7	0.57 ± 5.34
		5	0.80 ± 0.64	0.95 ± 1.50	0.79 ± 0.64	0.21 ± 8.63	0.70 ± 4.90
FEMNIST	Square	-	0.94 ± 0.20	0.94 ± 0.96	0.92 ± 0.55	0.11 ± 0.53	0.91 ± 0.67
	Cross	-	0.95 ± 0.22	0.97 ± 0.61	0.93 ± 0.82	0.15 ± 1.50	0.86 ± 12.2
	Copyright	-	0.93 ± 0.18	0.89 ± 1.76	0.91 ± 0.99	0.12 ± 5.93	0.79 ± 3.94
GTSRB	Yellow post-it	21/22	0.82 ± 0.53	0.96 ± 0.50	0.78 ± 0.66	0.03 ± 1.97	0.78 ± 0.71

TABLE IV: **Non-IID case**: mean results ± standard deviation (std) of experiments. The middle column shows the best results obtained before the defense. The right column shows results after the defense during inference. Std are multiplied by 10^2.

For the IID case, benchmark methods perform equally to our defense. However, the efficiency of RLR and wDP depend on the tuning of their respective hyperparameters.

For non-IID cases, FUBA performed well in comparison to the other state-of-the-art methods. In other approaches, we encounter two situations. On the one hand, the clean performance can be fairly maintained but the backdoor success remains high. This particularly happens for wDP. On the other hand, the backdoor success can be very low but there is an important drop in the clean accuracy, which happens in RLR and Med, especially for the labels skew $s = 1$ case. Poisoned updates can go undetected by these defenses in the non-IID case because the non-IID distribution of data among clients provides enough space for attackers to hide poisoned updates from being detected throughout the federated process. Our defense does not face the same issue because it does not reject potentially poisoned updates during training. Instead, it leverages the information contained in these updates to reconstruct triggers that can help identify and mitigate the effects of the attack.

Further, we also tested FUBA for 40% and 60% of attackers (Table VI), on one case only for space constraints. Our defense works despite the number of attackers being more than 50%. This is not the case for various defenses [5]–[7], [10] that require a majority of loyal clients.

However, although FUBA is not much affected by the clients' distributions, it is more subject to the complexity of the attack trigger. For both IID and non-IID cases, our defense is highly robust for the square, cross attack and yellow post-it because the backdoor success is strongly decreased whereas the clean and poisoned accuracy remain stable. This is a strong result because it means that our method not only deters the effect of the BA but also classifies data correctly, even if poisoned. However, complex patterns such as the copyright are more complicated to recover by Alg 1. To ensure that the attack is hindered, we recover a larger zone from the reconstructions (Eq. 8). Consequently, some benign values will be blurred, which can explain a drop in the clean accuracy. Further, if most of the pattern has not been estimated, when using `KernelMean`, the values of the pattern that were not detected can pollute the entire reconstruction with incorrect values.

Finally, a concern with our method can be privacy, as we invert global models. However, we argued that the obtained patches represent only a small proportion of the rough reconstructions we perform. Our goal is not to reconstruct the training data but only the attack pattern. If we compare true samples of the clean training dataset (Fig 4a, Fig 5a) to the first results of the reconstruction (Fig 4b, Fig 5b), we notice that the human eye can not really recognize distinct data features in the reconstructions. Finally, after γ-cleaning (Fig 4e, Fig 5e), our final result does not reveal distinctive information about the training datastes.

Moreover, FUBA is built at the end on the FL process, this latter can be run with a secure aggregation algorithm [20] to hide clients' updates from the server. In this way, information about a specific client can not be inferred by gradients inversion [21]. Additionally, we use a simple approach with very few SGD iterations $\mathcal{I}$ to be sure that we only retrieve a pattern highly correlated to the output decision.

V. Conclusion and future work

In this work, we propose a defense for BAs in FL. The defense is performed by the server who has no access to common data. It reveals information about the attack to the users, who use this information to diminish the effects of the attack during inference. FUBA is based on the strength of the attack and is not altered by the distribution of the participants. It has thus the important asset that it can be applied to cases where data is heterogeneous among participants. Extensive experiments have shown that our defense succeeds in mitigating BAs while maintaining high accuracy on clean inputs. To the best of our knowledge, this is the first robust solution to backdoor attacks in an FL scenario, that does not require any information or

		Ours		RLR		wDP		Med	
Dataset	s	CA	BS	CA	BS	CA	BS	CA	BS
MNIST	IID	0.98	0.10	*0.96*	*0.00*	0.90	0.18	0.97	0.11
	5	**0.93**	**0.10**	0.49	0.00	0.73	0.34	0.87	0.15
	1	**0.82**	**0.11**	0.02	0.00	0.80	0.99	0.11	0.10
FashionMNIST	IID	0.85	0.10	*0.83*	*0.07*	0.72	0.41	0.82	0.14
	5	**0.79**	**0.10**	0.50	0.02	0.60	0.35	0.68	0.26
	1	**0.66**	**0.12**	0.08	0.00	0.62	0.99	0.12	0.25
GTSRB	IID	0.83	0.02	*0.89*	*0.00*	0.72	0.44	0.84	0.50
	21/22	**0.78**	**0.03**	0.84	0.52	0.63	0.36	0.71	0.44

TABLE V: Comparison with existing defenses on IID and non-IID cases (label skew) for square attack.

	20 attackers	40 attackers	60 attackers
CA	0.85 ± 0.49	0.81 ± 0.16	0.80 ± 0.72
BD	0.10 ± 0.60	0.10 ± 1.20	0.10 ± 1.46

TABLE VI: Our defense for different numbers of attackers on the square attack for FashionMNIST (mean results $\pm$ 10^2 standard deviation).

data from participants and works regardless of the degree of heterogeneity between clients' local data distribution and the number of malicious clients.

As part of future work, it would be interesting to extend our defense on more subtle yet strong attacks such as [13], [22]. Moreover, we could also apply more sophisticated methods from the image reconstruction literature [23], [24] to delete the threat zones. Moreover, to reinforce privacy, we could also consider robustness against membership attacks [25] and thus validate that our defense still functions when the federated process is done with differential privacy [26].

VI. Acknowledgments

This publication was made possible by the use of the FactoryIA supercomputer, financially supported by the Ile-deFrance Regional Council.

References

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

[2] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[3] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[4] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 070–16 084, 2020.

[5] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[6] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[7] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against backdoors in federated learning with robust learning rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9268–9276.

[8] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[9] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 707–723.

[10] C. Zhao, Y. Wen, S. Li, F. Liu, and D. Meng, "Federatedreverse: A detection and defense method against backdoor attacks in federated learning," in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 2021, pp. 51–62.

[11] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv preprint arXiv:2011.01767*, 2020.

[12] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *CoRR*, vol. abs/1807.00459, 2018. [Online]. Available: http://arxiv.org/abs/1807.00459

[13] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[15] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

[16] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[19] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *The 2011 international joint conference on neural networks*. IEEE, 2011, pp. 1453–1460.

[20] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *CoRR*, vol. abs/1611.04482, 2016. [Online]. Available: http://arxiv.org/abs/1611.04482

[21] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" *CoRR*, vol. abs/2003.14053, 2020. [Online]. Available: https://arxiv.org/abs/2003.14053

[22] K.-H. Chow and L. Liu, "Perception poisoning attacks in federated learning," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2021, pp. 146–155.

[23] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[24] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of graphics tools*, vol. 9, no. 1, pp. 23–34, 2004.

[25] R. Shokri, M. Stronati, and V. Shmatikov, "Membership inference attacks against machine learning models," *CoRR*, vol. abs/1610.05820, 2016. [Online]. Available: http://arxiv.org/abs/1610.05820

[26] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private language models without losing accuracy," *arXiv preprint arXiv:1710.06963*, p. 84, 2017.