Effectiveness of Invisible Backdoor Attacks in Federated Learning

Marinus, Matthijs J.

Award date:
2024

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Interconnected Resource-aware Intelligent Systems Research Group

Effectiveness of Invisible Backdoor Attacks in Federated Learning

Master Thesis

Matthijs Marinus

Supervisors:
Prof. Dr. ir. Nirvana Meratnia
MSc. Vasileios Tsouvalas

Members of the assessment committee:
Prof. Dr. ir. Nirvana Meratnia
MSc. Vasileios Tsouvalas
Dr. Savio Sciancalepore

Eindhoven, June 2023

Contents

List of Figures

List of Tables

List of Abbreviations

FL - Federated Learning
BA(s) - Backdoor Attack(s)
FiBA - Federated invisible Backdoor Attacks
ASR - Attack Success Rate
Acc - Accuracy
ML - Machine Learning
PCA - Principle Component Analysis
I.I.D. - Independent and Identically Distributed
SGD - Stochastic Gradient Descent

Chapter 1

Introduction

Federated Learning (FL) is a Distributed Machine Learning (DML) paradigm that enables collaborative training of deep neural networks across multiple nodes (clients) without sharing raw data. In FL, updates to the deep neural network models (e.g., their learnable parameters) are performed entirely on-device and communicated to a central server, which aggregates these updates to produce a unified global model. Prospective applications of FL include finance [53], medical [27, 7], next-word prediction [40], and commercial sectors [66], where the privacy of clients data has to be respected. For example, in hospitals where patient data is often limited or incomplete, training a separate disease detection model for each hospital results in low performance. However, training a single model across all hospitals' data through FL can significantly improve performance without aggregating data centrally [65].

Concurrently, the distributed nature of FL exposes it to certain risks. Malicious users (referred to as attackers) who infiltrate the training stage of FL can steer the model in a desired direction by manipulating their contributions to the global model aggregation (i.e., their model updates). This vulnerability opens FL to various attack schemes, such as inference attacks, data poisoning, model poisoning, and backdoor attacks (BAs) [8]. These attacks allow malicious users to steal information, misdirect the global model, or implant hidden behaviors for later exploitation. Despite significant efforts to design defensive strategies, more complex attacks continue to bypass these defenses [49, 35].

There is a notable increase in the frequency of BAs [43], where a malicious user implants a "*trigger*" in the global model by sending updates that force the model to learn the "*trigger*" alongside its normal behavior on unmodified data [2, 61, 36, 56, 64, 22]. Once trained, the attacker can embed such "*triggers*" in data, forcing a misclassification in the model's prediction. For example, a small pixel patch in an image corner could steer the model to misclassify all such images as a target class [23]. While the visibility of the patch is not a concern when an attacker has sole access over the data, its importance becomes crucial when the attacker's access to the data is time-constrained (e.g., a temporary breach of a hospital's network) or when multiple parties have access to the data (e.g., doctors in a hospital's database). In both cases, the visual "*stealthiness*" of the "*trigger*" is essential [43, 33, 68]. Hence, the "*trigger*" must effectively steer the model's misclassification as well as remain visually undetectable when inspecting the data – ensuring that the malicious modifications stay hidden.

Developing triggers that are visually hard to detect is well explored in centralized machine learning [42, 14, 54, 55, 41, 57, 34, 3]. However, in the federated setting, attacks and defenses typically rely on simple pixel patterns localized to one area of the image [2, 36, 56, 64, 22]. Neither complex triggers, which are more spread out, nor triggers focusing on visual "*stealthiness*" have been explored in FL. Furthermore, since these attacks are generally much more subtle than obvious pixel patterns, they are already harder for models to learn; this challenge is further intensified in the federated setting, where training the global model with a trigger becomes even more difficult. *In this work, we assess how successful invisible BAs can be in the federated setting as well as ways to improve these types of attacks.*

1.1 Problem Statement

Identifying the importance of visual "*stealthiness*" and effectiveness of trigger patterns in BAs in federated settings, we define our problem statement as follows:

Problem Statement 1: *Visually indistinguishable backdoor trigger patterns are harder for models trained in FL to learn due to the limited local training time and the need to compete with honest clients' updates during model aggregation.*

The primary feature we aim to investigate is the visual imperceptibility of BAs. These attacks must create minimal differences between original and poisoned images, undetectable by humans, yet learnable by models during the FL training stage. In centralized settings, models can undergo extended training, providing ample opportunities for models to memorize these triggers. However, in FL, clients have limited time to train their model locally prior to transmission of model updates to server-side. Apart from the training duration, the aggregation process, which creates a representative model from all submitted models, typically involves some form of averaging. This process further dilutes the model updates from attackers with those from honest clients, causing loss of crucial information about the triggers.

Given the training challenges in FL to learn backdoor triggers and the competition for model updates during server-side aggregation, there is a clear distinction in learnability between simple, concentrated trigger patterns and those with complex, dispersed spatial distributions.

Problem Statement 2: *Subtle backdoor trigger patterns, which involve slight pixel modifications or image warping, are more susceptible to being diluted during the model aggregation process compared to more obvious pixel patterns.*

Unlike simple pixel patterns typically located in a corner, the proposed trigger patterns are deeply ingrained throughout the image, sharing weights with features used for normal class identification. This increases competition during aggregation, as other clients also utilize these weights, likely leading to significant information loss for the subtle triggers compared to simple patterns.

These problems raise questions about about the feasibility of invisible BAs in the federated setting. Therefore we investigate the performance of invisible BAs in varied settings, additionally under common FL BA defenses to be able to determine how successful the attacks can be in FL. Secondly, we investigate a new strategy to embed invisible backdoor triggers in images by leveraging model attention to hide the backdoor more optimally in federated settings. This might provide new insights into optimal ways to hide the backdoor triggers in the federated setting while also increasing the performance of the attacks. Additionally, while we expect the aforementioned problems to impact invisible backdoor triggers more heavily than simple trigger patterns, it is not the case that these problems do not influence simple/obvious triggers either. Techniques have been studied to improve the effectiveness of BAs in the federated setting expressed in terms of attack success rate (ASR). Techniques such as model replacement which can be described as *boosting* the model updates [2] to overpower the global model, or carrying out distributed versions of the attack [64] have been shown to be effective at increasing the ASR of the BAs. We investigate if these techniques can benefit invisible BAs as well.

1.2 Research Questions

Following the description of the problem statements, our two main research questions are defined as follows:

What is the ASR of BAs with visibly undetectable triggers in FL under various settings?

How can the ASR be increased using boosting techniques and trigger generation/distribution methods?

From the aforementioned research questions, we derive the following number of sub-research questions (milestones):

1. Under which federated setting do the selected BAs achieve similar attack success rate and model performance when compared to their centralized counterparts?
2. How do different federated learning defenses affect the attack success rate and model performance of the selected BAs?
3. To what extent can the attack success rate of the selected BAs be increased with model boosting and distributed attacks?
4. What is the effect of hiding the trigger in areas with low global model activations on the attack success rate?

1.3 Report Structure

The rest of this thesis is structured as follows. First, Chapter 2 provides a literature study which categorizes different attacks in FL and discusses relevant recent work on invisible BAs and BAs in FL. Chapter 3 covers preliminary concepts and contains background information on attacks and defenses we intend to use in our work. Chapter 4 discusses our approach, framework and adaptations used to evaluate and improve our selected attacks. We discuss our results in Chapter 5. Finally, conclusions and recommendations for future work are presented in Chapter 6.

Chapter 2

Related Work

Backdoor attacks are an established research field in ML [43]. With the advent of FL, interest in backdoor attacks under federated settings has surged in recent years [43, 7, 66, 53, 49]. These attacks exploit the distributed training nature of FL by manipulating local model updates to steer the model's behavior when a specific trigger is present in its input. This is particularly concerning in human-critical sectors, such as the medical domain [16], where such attacks can negatively impact patients' health. In this section, we discuss both related backdoor attacks and defense mechanisms.

2.1 Backdoor attacks

While backdoor attacks can be classified in various ways, in this work we differentiate backdoor attacks in federated settings based on how their trigger is created [43, 49]: either through data poisoning or model poisoning. Data poisoning attacks involve modifying the data and/or labels used for training to craft local models that exhibit specific behaviors. These models' updates are then shared with the server for model aggregation in FL. Alternatively, in model poisoning attacks, attackers handcraft local model updates directly without requiring any model training. In our work, we primarily focus on data poisoning attacks, with a particular focus on federated settings and the visual stealthiness of triggers. Data poisoning attackers can be distinguished by the appearance and distribution of the trigger, specifically we can have *semantic* and *artificial* triggers.

Semantic triggers. Semantic triggers are part of the original image, for example the labels of all blue shoes are flipped from "shoe" to "pants" in an attempt to teach the model that blue shoes are actually pants. A good example of a semantic backdoor attack is shown in [2]. Cars with a green color have their label flipped to bird. Similarly cars with stripes could also have this same flip applied to them. Even images of cars with stripes in the background have been shown to work in this context. However the effectiveness of this attack is limited by how severely the updates are diluted by other users since other users are also likely to have many images that have these properties. Therefore a better strategy is applied in [61] where properties are targeted which have a low chance of also existing in other users data-sets, i.e. a certain brand of car or airplane that is in the low end of the data distribution, called *"attack of the tails"*. The study gives both theoretical and practical proofs of the attack working. It also shows that defenses like (multi)-Krum are effective against defending against these types of attacks.

Artificial triggers. Artificial triggers are triggers which are imposed on the image, for example a pixel pattern that has been added to the image. Such types of attacks in the federated setting have been studied in [2] and [36]. These two works prove that crafting an artificial pattern, imposing it on pictures, training local models and sending updates accordingly can work in the federated setting, reaching success rates of 100%. Important to note is that the attacks seems to be quickly

Table 2.1: Summary of existing works on invisible Backdoor Attacks.

Attack	Category	Domain	Tested	
			Centralized	*FL*
Blended [11]	Blended	Visual	✓	×
SIG [4]	Blended	Visual	✓	×
ReFool [39]	Blended	Visual	✓	×
Trojan [38]	Blended	Visual	✓	×
FTrojan [62]	Transformation	Frequency	✓	✓
LSB [33]	Transformation	Visual	✓	×
WaNet [42]	Transformation	Visual	✓	×
AdvDoor [69]	Transformation	Visual	✓	✓
FIBA [17]	Transformation	Frequency	✓	×
DUBA [20]	Transformation	Frequency	✓	×
ISSBA [41]	Optimization	Visual	✓	✓
LIRA [14]	Optimization	Visual	✓	×
SATBA [72]	Optimization	Visual	✓	×
FedFIBA [68]	Optimization	Visual	×	✓

forgotten by the model if the attack(ers) are not present in subsequent rounds. A different way to create such a pattern, exploiting the distributed nature of federated learning, is by distributing the pattern over different malicious clients such as in [64]. Here, all the malicious participants get a part of the global pattern for their own data-set and try to teach the global model their local pattern. In the end these local patterns are combined and form the global trigger a malicious user can use to backdoor the model. This results in a better detection evasion rate because the model updates of the attacks are more dissimilar. This can even be improved by having all the malicious users learn an optimal pattern for their local data-set [22]. In the end the global pattern is reconstructed by combining all these optimal local patterns. This results in a more successful attack than previous techniques. However, these kind of artificial backdoor attacks are few and far between and lack extensive bench-marking on various defenses. Secondly, none of these attacks focus on a non-trivial factor of the attacks which is the *ocular* stealth of the triggers, which is an observation supported by [43]. A human could very easily spot these artificial trigger patterns negating the attack.

2.1.1 Stealthiness of Backdoor Attacks

In centralized ML, following the success of early artificial backdoor attacks such as BadNet [23] — later applied in FL [37] — recent efforts have focused on enhancing the visible stealthiness of these attacks [4, 39, 38, 11, 33, 42, 17, 20, 41, 14, 54, 55, 3] (see Table 2.1). Based on the trigger generation process, we classify stealthy backdoor attacks in 3 main categories:

- ***Blend-based BA:*** Blend-based attacks, such as Blended [11], SIG [4], ReFool [39], and Trojan [38], generate predetermined triggers, such as images [11] or sinusoidal strips [4], and merge them with the original images. While stealthier than simpler pixel-based triggers like BadNet, these earlier iterations of backdoor attack triggers still remain visible.

  Transformation-based BA: Transformation-based attacks, such as the LSB Attack [33], WaNet [42] AdvDoor [69], FIBA [17], and DUBA [20] aim to generate triggers through sophisticated algorithms or transformation functions. For example, LSB's [33] trigger modifies the least-significant bits of an image based on a pre-defined string converted to bits, while WaNet [42] uses a warping field with varying intensity to create slight warps in the original image for the model to learn. Alternatively, FIBA [17] and DUBA [20] leverage the frequency spectrum of images to create more effective backdoor attack triggers. Transformation-based

attacks are much stealthier than blend-based attacks, improving performance against post-aggregation defenses like Neural Cleanse [60], as their complex patterns are harder to reverse engineer.

- ***Optimization-based BA:*** Optimization-based attacks generate triggers through a series of optimization steps, generally using deep neural networks and/or optimization metrics. A significant subset of these attacks use auto-encoders to embed information within images [54, 41, 55, 3]. The encoded information can be random noise [57] or entire images [54, 41, 55, 3]. Here, the encoder aims to embed the information in the original image and minimize the perceptual differences between the original and modified images, while the decoder recovers the hidden information. These attacks, similar to LSB [33], are based on steganography. Alternatively, LIRA [14] solves an optimization problem to maximize normal data performance while fooling the model to misclassify between two classes using a transformation function. This way, LIRA generates a small trigger while maintaining a high attack success rate. Lastly, SATBA [72] enhances robustness and stealthiness by leveraging the model's spatial attention, embedding triggers in regions the model focuses on, thereby improving resistance against backdoor defenses.

BA Stealthiness in FL. While stealthiness in centralized settings is well-explored, it has not received similar attention in federated settings. Apart from our work, Federated Invisible Backdoor Attacks (FedFIBA) [17][1] is, to the best of our knowledge, the only other study focusing on the stealthiness of backdoor attacks in FL at the time of writing. In FedFIBA [68] the effectiveness of three transformation based invisible BAs in federated settings were explored, highlighting that increased stealthiness inversely affects the memorization of the trigger (i.e., the stealthier the attack, the harder it is to make it work successfully). Based on these observations, FedFIBA [68] was proposed, using an optimization function to balance stealthiness and memorization ability by evaluating the trigger with a visual perception score, (SSIM) [63], and minimizing the L2 score of poisoned versus original images. If the visual perception score exceeds a threshold, the penalty on the L2 score is increased, leading to a smaller trigger, and vice versa. Additionally, to enhance resistance to backdoor mitigation, L2 regularization was introduced to align the model's attention regions for both the original and the backdoor samples, similar to SATBA [72]. In contrast FedFIBA and SATBA we explore the effect of hiding the trigger in the regions for which the model has the lowest attention since we prioritize the goal of mitigating influence from other clients.

2.2 Defense Mechanisms

As interest in backdoor attacks in FL has grown, there has also been an increasing interest in developing techniques to defend against them.

2.2.1 Model Updates Comparison

A straightforward way to defend against backdoor attacks is to detect differences between model updates among clients. An example of such a defense mechanism is FoolsGold [19], which aims to address orchestrated Sybil attacks (targeted attacks). The core of this defense involves comparing the updates of clients, where colluding clients are expected to have similar cosine similarity in their updates, while honest clients exhibit relatively dissimilar updates. Once detected, suspected attackers' updates are excluded from the model aggregation process. However attackers can easily evade this defense by crafting model updates with near-zero cosine similarity among classes, except for the targeted class. Alternatively, Auror [52] clusters updates over several federated rounds, and flags users as malicious when their updates deviate from the majority. However it assumes i.i.d. data and fails in non-i.i.d. settings, where malicious behavior can appear more common

[1] We refer to it as FedFIBA to avoid confusion with existing FIBA [17] work.

than non-malicious behavior. Furthermore, (Multi-)Krum [9] uses clustering to compare model updates and assumes that the number of attackers remain below a certain threshold across FL training stage. Once the threshold is exceeded, the defense fails to identify attackers and penalize benevolent clients. Contra [1] calculates the cosine similarity between normalized gradient vectors of local updates and uses historical data to identify malicious users based on a scoring function.

2.2.2 Robust Models Aggregation

In addition to comparing clients' model updates, dealing with backdoor attacks during FL's model aggregation step has also been explored [10, 47], where the goal is to address statistically anomalous updates. Here, various aggregation techniques have been proposed as alternatives to *FedAvg* [40]. For example, Median aggregation [10] uses the median of model updates to mitigate outliers. Similarly, Trimmed Mean [10] prunes a percentage of the highest and lowest values for each update to eliminate outliers. However, most aggregation rules fail under an extreme number of attackers or in extreme non-i.i.d. settings [19].

2.2.3 Model Inspection

Model inspection techniques inspect the model during training or after the FL training has been finished. One such category is based on Model Pruning [71]. These kinds of techniques aim at removing unwanted weights from the model which have deviating behavior in some way. One such defense, inspired by model pruning, applied in FL is Lockdown [28]. This defense reduces the model weights each client has access to by determining the most and least important weights for each client every round using a subspace mask. Eventually Consensus Fusion is applied which serves as a voting mechanism such that the clients have to agree on the importance of each weight based on a threshold. If not enough clients find it important the weight is excluded from the final model. Another well known defense is Neural Cleanse [60] which tries to reverse engineer triggers from the final model. By trying different trigger pattern combinations it aims to find anomalous activation's which it can classify as outliers and determine if a model has been poisoned. Otherwise, GradCam [50] can be used as a manual inspection technique to study the activation maps of the model given different data samples to see if it displays strange behavior.

2.3 Non-trigger Based Attacks in FL

Since backdoor attacks share a few characteristics with other types attacks, such as *label flipping* or *model poisoning*, we briefly analyze such works, analyzing their effect with regard to aspects such as detection evasion or success of the attacks.

2.3.1 Label Flipping Attack

Label flipping attacks are a subset of data poisoning attacks based on the premise of flipping labels of a certain class to another class. This can be targeted or untargeted. A very early basic label flipping attack is described in [46]. In this paper an untargeted attack in formulated as an optimization problem which tries to maximise the loss of the global model subject to a loss function which tries to find a poisoned set which maximises this loss. A simple defense proposed is to use K-NN to cluster all the data points and try to relabel samples if they cross a certain scoring threshold since those are probably poisoned. However the defense is too trivial, uses i.i.d data and has only tested for binary classification.
Another example of a label flipping attack is described in [32]. Datasets are clustered using agglomerate hierarchical clustering to classify data and find samples with a unclear classification boundary (based on the value obtained from the clustering). These labels are flipped. TrAda-Boost is proposed as a defense which is used to assign weights to samples from both a clean and contaminated data-set. Weights of poisoned samples should go down indicating that its distribution is not in accordance with the trained model. This defense needs clean as well as poisoned data

to train on and requires extra model training, therefore it is not very practical. A similar approach in a classic ML setting is described in [51] where the defense is focused around detecting Trojans in hardware. This uses a very similar Catboost approach but suffers from the same problem that it needs a clean validation data-set which often is not available. There is yet another one that is very similar which is also used against targeted label flipping based on Adaboost [12]. An Adaboost model is used to assign weights to data points. Badly labeled samples are assigned more weight and are chosen to be re-labeled using a semi-supervised learning technique. It has not been tested in federated learning and it would also require extra model training.

Some interesting experimental results are discussed in [59]. Even though the label flipping attack is only carried out on i.i.d settings, it still has some interesting conclusions. It describes a targeted attack on a selected class and shows that attacks in later rounds have the most effect on the model performance. This is a result of the fact that when attacks are carried out and ended in earlier rounds, the model has an easier time recovering from those attacks. A secondary observation is that finding the vulnerable class is non-trivial since there is not necessarily a correlation between misclassification of non-poisoned samples, performance and attack effectiveness. It also shows that the remaining classes are unaffected in regards to performance proving that the attack can be stealthy in that regard. The proposed defense calculates the difference between the local and global model and stores it in a list. After each round the list is standardized and principle-component-analysis (PCA) is used to create clusters to identify malicious clients.. We choose to evaluate these defence mechanism in our evaluation to analyze its performance under non-i.i.d. settings.

2.3.2 Model Poisoning

Model poisoning attacks directly modify the updates sent to the global model. This usually means that the updates are directly crafted by the attacker itself in contrast to data poisoning attacks where the malicious updates are generated by training on poisoned data. One such attack works as follows [18]. In an attempt to "steal" the global model, which in some settings like a medical or financial setting has a high value, an attacker can try to participate in the federation, thereby obtaining the global model, without actually participating. They can do this by either plain-free riding which is just to re-upload the global model parameters back to the global model or by adding noise to the local updates to try to evade detection techniques. The paper provides theoretical guarantees but no real defense methods are discussed. Other model poisoning attacks worth noting are shown in [6] and [56]. In [6] model updates are crafted by optimizing an attacker objective function which corresponds to the desired malicious behavior. In combination with smart boosting and stealth techniques the paper shows how different defences affect model convergence and performance. While defences like aggregation techniques and Krum can be effective, the paper shows that through choosing the right parameters for their techniques they can still be evaded to a certain extent. In [56] the goal of the attacker is to replace the global model by scaling his updates in such a way that it overwhelms other updates. Even though the attack can be successful when the number of attackers is high enough, techniques like norm-clipping (e.g. making sure updates can only fall within a certain range of each-other) and differential privacy are extremely effective at negating the attack.

2.4 FL Frameworks

There are a couple of frameworks which can be used to facilitate federated learning available. We highlight a few here. TFF (Tensor Flow Federated) [58] is a federated learning framework developed by Google. It is built on TensorFlow and therefore, coupled with its high level API is easy to convert already existing TensorFlow models to federated learning models. However this high level API also means that much of the framework has to implemented by the user and without familiarity with TensorFlow it leads to a steep learning curve. PySyft [45] is a framework build on PyTorch to enable "Remote Data Science" using techniques such as Federated Learning, differential privacy and encrypted communication to maintain data privacy. It is relatively new

and not a lot of examples and documentation is available. FATE (Federated AI Technology Enabler) [15] is a production scale federated learning framework. This framework is meant to build large-scale production ready applications and is very large and rigid and thus has a high learning curve. The Flower framework [5] is a more user friendly smaller scale open source federated learning framework which focuses on ease of use and adaptability. It has a simple and easy to use separation between servers and clients with examples on how to create your own aggregation functions and how to expand the server and clients functionality.

Chapter 3

Preliminaries

3.1 Federated Learning

The goal of federated learning is to learn a global model by training models locally on client devices and sharing model weight updates. This eliminates the need for a global dataset and eliminates privacy concerns that exist when there is a need to collect a global data-set. In each federated round, each client downloads the global model and trains the model for a few epochs locally in parallel. At the end of local training the updated model weights are communicated to the server which aggregates the results and updates the global model. A federated learning sessions can be described by the following steps:

1. **Initialization** The global model θ is created and weights are initialized.
2. **Local Model Training** The server selects K clients for local model training. Each client $i \in K$ downloads the global model θ and copies it as their local model θ_i. Each client then performs a number model update steps to minimize loss function $L_i(\theta_i)$ usually based on SGD.
3. **Aggregation** The server collects the model updates from all K clients and aggregates them. The most common aggregation strategy is FedAvg [40]. This strategy aggregates the weights in the following way: $\theta^{t+1} = \sum_{i=1}^{K} \frac{n_i}{n} \theta_i^{t+1}$ where $\frac{n_i}{n}$ is the fraction of data from each client i over all the data n. After aggregation the global model is updated with the new weights.
4. **Iteration** Steps 2 and 3 are repeated for the selected number of federated rounds.
5. **Model Evaluation / Deployment** Once the federated learning rounds are complete, the final global model θ can be evaluated and deployed for use on live data.

A basic overview of federated learning can be seen in figure 3.1. Federated Learning can broadly be split up into three categories. *Horizontal, vertical and transfer learning.* In horizontal federated learning data between clients shares a feature space but holds different data. In vertical federated learning data between clients differs in feature space but has the same data IDs. In federated transfer learning both the feature space and IDs of the data differ. This study focuses on horizontal federated learning.

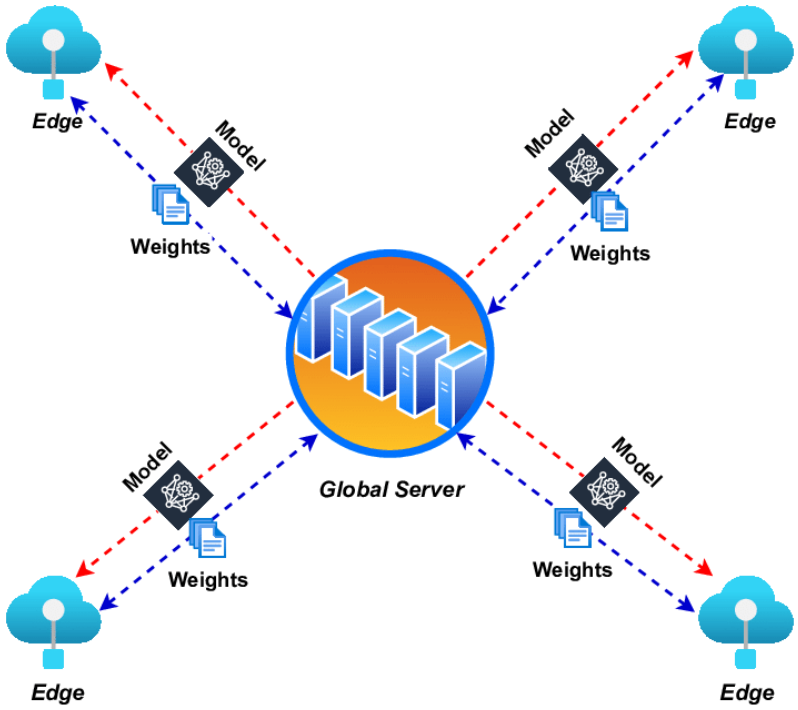


Figure 3.1: Overview Federated Learning

3.2 Backdoor Attacks

LSB Steganography The LSB attack is transformation based BA based on image steganography. Based on the observation that humans cannot observe the differences when the least significant bit of pixels are modified, the attack remains very stealthy to the human eye. To apply the attack first a string is selected. This string is convereted into ASCII characters, which are in turn encoded to 8-bit binary strings. Concurrently each pixel of the image across all channels is also encoded to binary. The string bit sequence can not be encoded into the image by looping over all the pixel values across all channels. In each step the current least significant bit of the current pixel value is changed to the current bit in the string sequence. This sequence of steps is repeated until the entire string is encoded in the image. Since an image contains $W \times H \times C$ pixels, we can encode that much information in the LSBs. If the bit sequence of the string is longer than this $W \times H \times C$, the process repeats but now takes the second LSB. This can be repeated until the entire sequence is encoded.

WaNet WaNet produces images which look natural and invisible by applying "elastic" image warping. Contrary to most other attacks which introduce additional information to an input image, elastic image warping only manipulates existing pixels of the image. While humans are good at spotting inconsistent parts of images, we are bad a noticing small geometric changes. The goal is to create a function B which can be used to poison images x:

$$B = W(x, M) \tag{3.1}$$

where W is the warping function an M is a warping field. M can be used as a backwards sampling grid to assign pixels in the target image based on the sampled locations in the original image. Formally M is defined by:

$$M = \phi(\uparrow (\psi(rand_{-1,1}(k, k, 2)) \times s)) \tag{3.2}$$

$\psi(rand_{-1,1}(k, k, 2))$ creates a random uniform grid of size $k \times k$. This grid is normalized (ψ by dividing the elements by the mean absolute value. These values are multiplied by a scaling factor s to create more or less pronounced warps. $\uparrow$ denotes an up-scaling operation to a $w \times h$ grid using bi-cubic interpolation. Finally a clipping operation ϕ is applied to make sure the sampling coordinates do not fall outside of the image borders. Applying B to an image x create a subtle and smooth warp in the image which can be learned by a DNN and associated with a different class t by flipping the labels of the poisoned samples. Additionally a conditional noise operation can be applied which adds random noise to the poisoned images and does not flip their labels. This help the trained model to learn the warp as the backdoor trigger instead of pixel artifacts.

Encoder Attack The Encoder attack is an optimization based invisible BA inspired by multiple similar works which use some form of auto-encoder to perform image steganography. The general structure of the auto-encoder is similar to [54, 41, 55], in which we have a pre-processing network, a encoder network and a decoder network. The purpose of the pre-processing network is to extract relevant information from the main image and the noise input. This extracted information is merged together and used as input to the encoder network. The encoder network merges the information into a single image. The decoder network tries to recover the noise input from the encoded image. To train this network we use a loss function similar to [3]:

$$\mathcal{L}(I, I', S, S') = \text{MSE}(I, I') + \alpha \cdot \text{MSE}(S, S')$$

where I is the original image, I' is the encoded image, S is the input noise and S' is the decoded noise.

LIRA The LIRA attack propose a new way to create an optimal backdoor attack by using an adversarial training schema between the classification model and a transformation function T which applies the backdoor to images. The transformation function can be based on a simple auto-encoder or a UNet architecture. Training of both the classifier and the transformation model is done in two stages. In the first stages both models are updated adversarially k number of steps. After k steps only the classifier model is updated. This is done because if the loss on the transformation model quickly converges to 0 it might cause the classifier model to get stuck in bad local minima since updates on the classifier depend on changes of T. During the first stage the loss of the classification model is used to calculate the loss given poisoned and clean samples. The resulting model is used to calculate the loss on poisoned samples given their new target label and used to update T. The new transformation function is then used to re-calculate the loss on the clean and poisoned data and used to update the original classifier model. The new classifier model and transformation function is then used in the next step. This process repeats k times after which only the classifier model is updated given clean and poisoned data. The transformation function T_ξ is formally defined as:

$$T_\xi(x) = x + g_\xi(x)), ||g_\xi(x))||_\infty \leq \epsilon \tag{3.3}$$

where $g_\xi(x))$ is a generator function which creates noise in the image. The chosen generator function is a simple auto-encoder. ϵ controls the stealthiness of the attack by limiting the l_{inf} norm of the generated noise to ϵ. An example of LIRAs attack stealth can be seen in Figure 3.2.

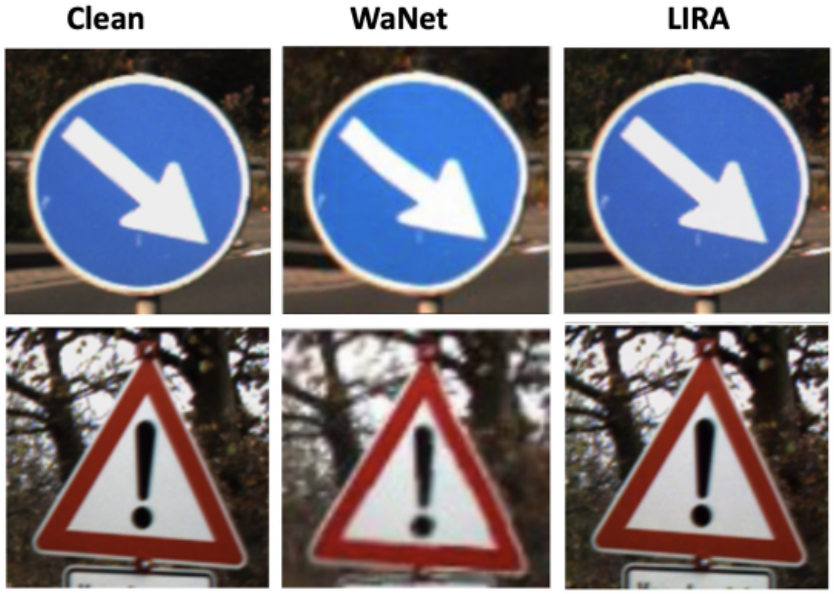


Figure 3.2: Image transformed by LIRA and WaNet, taken from [14]

3.3 Defenses

There are a lot of possible methods to defend against poisoning attacks in federated systems. The ones that are relevant to our type of backdoor attacks generally speaking fall into three different categories; robust aggregation, behavior detection and trigger detection/removal. We will describe these 3 categories together with some specific methods which embody these three categories. The defenses that will be chosen for experimental analysis will each fall into one of these categories.

3.3.1 Robust Aggregation

Defensive techniques which fall under robust aggregation aim at mitigating the impact of updates from attackers. They alter the way the model updates are aggregated in such a way that either updates that are significantly statistically different are either ignored or have very little impact in the aggregation. Two representative examples are the trimmed mean and median [67]. Using trimmed mean as an aggregation rule causes the lowest and highest fraction of updates to be ignored. For the median, simply the median of all weight updates for each weight is chosen. Both techniques aim at reducing the impact of outliers, which malicious updates are expected to have, compared to honest updates.

3.3.2 Behavior Detection

Behavior detection, in contrast to robust aggregation, aims at identifying malicious users instead of mitigating their impact. Defenses like FoolsGold [19], Auror [52] and Contra [1] fall into this category. These techniques generally compare gradient updates using, for example, the cosine similarity between updates to cluster updates into groups. Groups that are statistically different enough to others are chosen as attackers and are excluded from aggregation/training.

FoolsGold. FoolsGold is a model comparison defense which scales down the contribution of suspicious clients by comparing the cosine similarity of the submitted (historical) gradient updates of the clients. Malicious clients are theorized to have more similar gradient updates since they try

to embed a similar specific task into the model. By finding groups of clients which are more similar to each other than most others, this technique can lower their contribution to the global model.

(Multi)-KRUM. KRUM calculates the sum of distances between the update of one client to all others for all clients. Then the first $n - f - 1$ distances (n is the number of clients, f the number of attackers) are summed which represents the score of each client. The client with the smallest score is chosen as the global model update. Multi-Krum extends this by selecting the smallest m clients.

3.3.3 Model Inspection

Defenses in this category aim at finding the trigger patterns and/or removing neurons that have abnormal behavior on particular labels.

Neural Cleanse. One of the most well-known defenses in this category is Neural Cleanse [60]. The intuition behind Neural Cleanse is to reconstruct trigger patterns which cause anomalous classification behavior. It does so by solving the following optimization formula [60]:

$$\min_{m,\Delta} \quad \ell(y_t, f(A(x, m, \Delta))) + \lambda \cdot |m| \quad \text{for} \quad x \in X \tag{3.4}$$

Here f is the prediction function of the selected DNN, A is a function which applies a trigger pattern Δ given a mask m to each image $x \in X$. The mask m controls how much the pixels in the image x may be altered across all color channels. λ controls the weight of the second objective which is to minimize the mask. y_t is the target label which is currently being tested. ℓ is a loss function which measures misclassification. In other words, this function aims to find the smallest trigger pattern Δ controlled by mask m which causes the largest misclassification to target label y_t given all the samples in the dataset X. Once a reverse engineered trigger is obtained for each target class, their L1 norms are calculated. The *Median Absolute Deviation* is then calculated which is used to divide the absolute deviation of each data point. The results are normalized, which results in an Anomaly Index score. Any target label which ends up with an Anomaly Index > 2 is suspected to be an outlier with 95% certainty. Lower anomaly scores indicate that the trigger is well hidden in the model and thus hard to reverse engineer.

Lockdown. The Lockdown defense is based on the principle of subspace training. The essence of the defense is to limit the training of local models to isolated subspaces with the goal of preventing attackers from coupling their poisoning function to normal model behavior. Each client starts with the same subspace which is a mask which indicates for each layer which weights the clients can train on. Then each round the clients follow three steps to update their local subspace. First the client trains his local model using his own isolated subspace. Then at the end of each round and at the start of the following round the following two formulas are applied respectively:

$$m_{i,t+1} = m_{i,t+\frac{1}{2}} - \text{ArgBottomK}_{\alpha_t}\left(|w_{i,t,K}|\right) \tag{3.5}$$

$$m_{i,t+\frac{1}{2}} = m_{i,t} + \text{ArgTopK}_{\alpha_{t-1}}\left(|\nabla f_i(w_{i,t,0})|\right) \tag{3.6}$$

Here m_i indicates the subspace mask for client i. At the end of each round the least important α_t percentage weights are removed from the subspace in each layer (absolute smallest coordinates per layer). Subsequently at the start of the next round the α_t percentage of most important weights is added back into the subspace by looking at the absolute gradient values using the clients local dataset. Higher gradient magnitudes should indicate important parameters. This process repeats itself throughout training until at a certain round Consensus Fusion (CF) can be applied on the server. By collecting all the clients subspace masks and adding them together, Lockdown collects a count for each weight how many clients found it important. By choosing a threshold and eliminating all weights below this threshold, weights deemed important by attackers (which should not be found important by normal clients) are eliminated.

Chapter 4

Methodology

In this chapter, we begin by defining the problem of Backdoor Attacks (BAs) in FL in Section 4.1. Next, we present our framework for evaluating various invisible BAs in federated learning (FL), which we term Federated Invisible Backdoor Attacks (`FiBA`s), in Section 4.2. Section 4.2.3 discusses the adaptation of various invisible BAs in FL, which we enhance with different ASR boosting techniques in Section 4.2.4. Based on our hypothesis that *server-side aggregation can significantly impact the effectiveness of `FiBA`s*, we propose a novel trigger hiding technique specifically tailored for FL, detailed in Section 4.2.5. Finally, Section 4.3 introduces the defense mechanisms we will evaluate against `FiBA`s, focusing on the modifications made to integrate them into our framework.

4.1 Problem Formulation

We focus on the challenge of learning `FiBA`s. While invisible backdoor triggers have proven effective in non-distributed ML, their applicability in FL presents two unique challenges: (i) the need for small, invisible triggers makes them harder for models to learn, and (ii) during FL training, attackers' model updates compete with those of honest clients, limiting the global model's ability to learn the trigger.

To highlight these challenges, we examine *artificial* backdoor attacks, which aim to embed backdoor triggers into the global model by steering it to classify samples with artificial backdoor triggers as a target class, t. Specifically, attackers inject the trigger into locally-stored data, either through a transformation function or via a specialized model that generates a new image, creating the "*poisoned*" dataset ($\mathcal{D}_p$). Subsequently, during the FL training process, as depicted in Section 4.2, attackers' local training is altered to optimize the loss on both the original "*clean*" dataset ($\mathcal{D}_c$) and the "*poisoned*" dataset ($\mathcal{D}_p$). Formally, the attacker's optimization problem is defined as:

$$\min \mathcal{L}\left(\mathcal{D}_c, \mathcal{D}_p, h_\theta\right) = \sum_{i=1}^{|\mathcal{D}_c|} \mathcal{L}_{CE}\left(h_\theta\left(x_i\right), y_i\right) + \sum_{j=1}^{|\mathcal{D}_p|} \mathcal{L}_{CE}\left(h_\theta^*\left(x_j\right), t\right) \ , \tag{4.1}$$

where $\mathcal{L}$ is the supervised loss term over attacker "*clean*" ($\mathcal{D}_c$) and "*poisoned*" ($\mathcal{D}_p$) datasets, given a model h with parameter θ. With θ^* we refer to the model parameter once training on $\mathcal{D}_c$ is complete in a given federated round.

4.2 `FiBA`: Federated Invisible Backdoor Attacks

To validate our hypothesis that *invisible BAs are harder to learn in federated settings compared to centralized ML*, we investigate state-of-the-art invisible BAs within the context of FL. Our study focuses on both transformation-based and optimization-based BAs, specifically LSB, Encoder, WaNet, and LIRA. In our framework, we evaluate these attacks against widely-used BA defense

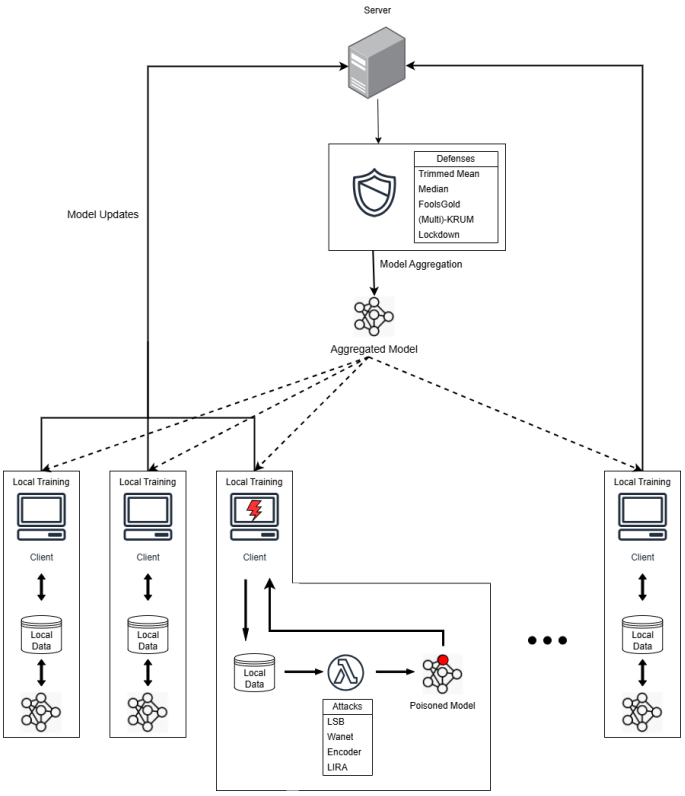


Figure 4.1: Federated Learning training process in the presence of invisible Backdoor Attacks

mechanisms in FL, including model update comparison, robust aggregation, and model inspection. While we initially assess the BAs in their original form, we incorporate with various techniques proven to enhance non-invisible BAs to determine their effectiveness on invisible BAs.

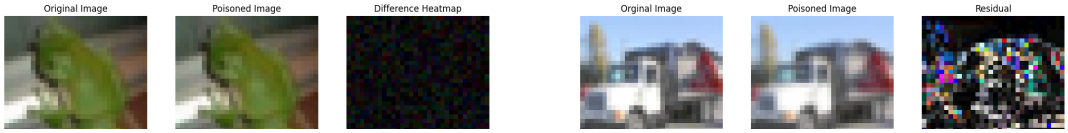
4.2.1 FL Training Process

The training pipeline begins with the server randomly initializing a neural network h_θ, which is distributed to all clients. Non-attacker clients perform local training by executing several local SGD steps on their data in parallel. In contrast, attackers select their invisible BA, modify a proportion of their local dataset with the chosen backdoor trigger, and use this poisoned dataset during local SGD steps. Upon receiving the model updates from all clients, the server aggregates the models. If no defense mechanism is applied, FedAvg [40] serves as the default aggregation strategy. Alternatively, a defense mechanism may be utilized during the server-side aggregation step to mitigate the impact of backdoor attacks by dropping or limiting the effect of specific client updates. An overview of the FL training process with BAs is shown in Figure 4.1.

4.2.2 FIBA Poisoning Design Choices

As BAs aim to force the trained model to learn a trigger that causes misclassification to an intended target class, we identify two crucial design choices in our evaluation:

Poisoning Strategy. An attacker can choose to either poison all classes to a single target class (t) (*all-to-one*) or poison a specific source class to the target class (t) (*one-to-one*). With both poisoning strategies being valid, the ultimate choice depends on the attacker's goals. For example, an attacker might want to prevent an autonomous car from stopping at any traffic sign (*all-to-one*) or only specific traffic signs (*one-to-one*) [24]. While *one-to-one* attacks are stealthier —



(a) LSB Image Steganography. (b) WaNet Transformation.

Figure 4.2: Transformation-based invisible BA effect on images

introducing focused triggers and thus affecting fewer model parameters — our preliminary experiments showed that *one-to-one* BAs exhibit high variance and volatility. In our framework, we adopt an *all-to-one* poisoning strategy, similar to [68, 14, 42, 33, 64] amongst others, to evaluate the effectiveness of invisible BAs. A analysis of target class selection in the *all-to-one* setting is provided in Appendix C.3.

Poisoning Rate. A crucial aspect of all BAs is the attacker's poisoning rate (P), which determines the proportion of locally stored data that the attacker will be poisoning. Following similar works [42, 14, 2, 48, 57, 64], both in centralized and federated settings, we set the poisoning rate to range between 0.1 and 0.5.

4.2.3 Adapting Invisible BAs to FL

As the invisible BAs attacks were initially designed for centralized ML, we performed the necessary modifications, which are discussed in this section.

4.2.3.1 Transformer-based invisible BA

The transformation-based BAs, namely LSB and WaNet, are implemented as data pre-processing steps prior to training, ensuring that invisible backdoor triggers are embedded in the datasets before the FL process begins. An illustration of both transformations on image samples is depicted in Figure 4.2.

LSB Attack. We discuss the core concept of LSB steganography in Section 3.2. In federated settings, the LSB attack remains unaltered, however in case of multiple attackers the underlying assumption is made that attackers have agreed on the encoded string beforehand, ensuring that each attacker creates the same local trigger.

WaNet Attack. In WaNet the input samples are warped using the WaNet algorithm, as discussed in Section 3.2. Similar to LSB, for this attack to work in the federated setting, attackers must agree on algorithm's hyper-parameters k and s beforehand. The same sampling field, s, is drawn from a uniform distribution between -1 and 1, where different s values among attackers would result in varied warping strengths and trigger patterns, reducing the attack's effectiveness in FL.

Apart from the warping transformation, WaNet injects additional Gaussian noise to random samples without label flipping to prevent the memorization of pixel artifacts instead of the warp as the trigger. In our preliminary experiments in federated settings, noise injection had no effect on WaNet's ASR as limited local training time prevented models from adequately learning the distinction; thus we omit it in the FL adaptation of WaNet. However, while the attack can still be effective without the noise injection step, it is important to note that model inspection defenses such as Neural Cleanse [60] can easily counter the BA by reverse engineering pixel artifact triggers. An example of an image poisoned through our WaNet implementation can be seen in Figure 4.2b.

4.2.3.2 Optimization-based invisible BA

Optimization-based invisible BAs are implemented as pre-processing steps before the FL training process begins, ensuring that backdoor triggers are embedded in the datasets.

LIRA Attack. The LIRA poisoning function is presented in Section 3.2, where the trigger is injected as a pre-possessing step on data. In federated settings, when multiple attackers are present LIRA requires additional communication overhead among attackers to coordinate the poisoning function. Specifically, we assume that attackers train the LIRA model on their (shared) data, prior to FL training stage. Subsequently, the model is then distributed amongst attackers to utilize for poisoning their locally stored data.

Encoder Attack. The Encoder-based attack is discussed in Section 3.2, where an auto-encoder is trained such that the encoder hides a trigger in an input image, and the decoder recovers it from the encoded image. In contrast to [54, 41, 55, 3] where an colored image was utilized as trigger - encoder trained to inject an image into sample input - we opt to utilize a single bit-string, similar to [34]; increasing the stealthiness of the BA. Specifically, the bit-string (referred as *noise pattern*) is converted into a grayscale $W \times H$ image and used as input into the auto-encoder, which is trained prior to start of the FL process. Furthermore, to draw conclusion about the attacks effectiveness in FL, we keep the same auto-encoder architecture, i.e, same number of layers, as [54, 41, 55, 3]. For full details about the Encoder's training, architecture and produced triggers, see Section 3.2 and Appendix B.

Once trained, the encoder is used as a pre-processing step. In federated settings with multiple attackers, they either agree on a common *noise pattern* and train their own auto-encoders separately or share a single auto-encoder model. Data is then poisoned by passing images through the encoder, while the decoder can be used to verify the agreed-upon pattern.

4.2.4 Boosting ASR with Existing Techniques

We further adapt widely-used ASR boosting techniques, typically applied to simple pattern triggers, to FIBA. These adaptations aim to investigate whether techniques such as trigger distribution or model replacement will similarly benefit FIBA.

Distributed Triggers. Splitting the trigger into smaller parts and distributing them amongst attackers has been shown to increase ASR of simple-pattern BAs [23, 64, 22]. However, applying this techniques to invisible BAs is more challenging, since the triggers are either image-dependent or spread loosely across the image. To achieve a similar effect in `FiBA`, we divide images into equal-sized rectangles based on the number of attackers, padding any remaining space to the last rectangle if needed. Attackers poison their data samples as usual, then replace the parts of the image outside their assigned rectangle with the original counterparts, giving each attacker a distinct part of the trigger to learn. We provide the algorithm of distributed triggers in `FiBA` in Appendix A.1.1.

Model Replacement. Scaling attackers' model updates to overpower other clients during (weighted) averaging, known as *model replacement* [2], has been shown to increase ASR. For *model replacement* to work, the attackers must discover two parameters: n, the total number of training samples used, and m, the total number of clients. With those two parameters, the attackers model update X can be scaled to replace global model G^t with W_j^{t+1} as follows:

$$W_j^{t+1} \simeq \frac{n}{r_j} X - (m-1)G^t \ , \tag{4.2}$$

where X denotes the attackers local model update from attacker j and W_j^{t+1} is the global model on round $t+1$. Here n indicates the total number of samples used to train the model, t is the current federated learning round, r_j is the number of samples attacker j trained on and G^t is the

Algorithm 1 Activation Mask Extraction

Require: model, dataset ds, input shape $(3, 32, 32)$
Initialize CAM: cam $\leftarrow$ SmoothGradCAM++(model, $(3, 32, 32)$, network.layer1)
Initialize mask: activation_mask $\leftarrow \mathbf{0}_{16\times16}$
for each batch (x, y) in ds **do**
    for each image x_i in batch (x, y) **do**
        logits $\leftarrow$ model(x_i)
        $A_i \leftarrow$ cam(arg max(logits), logits)
        activation_mask += A_i
    end for
end for
Sort and create top half mask:
sorted_indices $\leftarrow$ argsort(flatten(activation_mask))
top_half_mask $\leftarrow \mathbf{0}$
top_half_mask[sorted_indices[$0 : \frac{N}{2}$]] $\leftarrow 1$

current global model. To overpower the global model in the next aggregation step, we scale the attacker's current model updates by $\frac{n}{r_j}$ and subtract $(m-1) \times G^t$, the previous global model. We provide the proof in Appendix A.1.2. Model replacement can an effective ASR boosting technique for FIBA, assuming both m and n are known [2]. However, while m can be easily inferred by listening to the channel or through insider knowledge, the assumption that n can be realistically guessed or inferred throughout the rounds, as stated in[2], is unrealistic. Even a small variation from the true n value can lead to significantly different model scaling factors, resulting in poor model performance, as discussed in Section 5.2.4.2.

4.2.5 Boosting ASR via Low Activation Triggers

Based on the fact that global model attention is distributed across various regions and features, we propose a strategy to increase the ASR of `FiBA`s by hiding the trigger in low-activation areas of the global model. This approach aims to minimize competition during server-side model aggregation, as high-attention regions face greater competition from honest clients' updates. Our motivation stems from observing that invisible BAs like WaNet and LIRA, which often overlap with identifying features, perform worse than simpler attacks like BadNet in federated settings, as seen in Section 5.2.1 and 5.2.2. For instance, in a CNN identifying birds, high activation around the beak could lead to intense competition if a differently colored object is placed on the beak, causing misclassification as dogs. By avoiding such regions and hiding the trigger in less important areas, we can maintain the integrity of the global model's learning process and improve the ASR of invisible BAs.

Finding Low Activation Areas. To identify low activation areas, we utilize GradCAM [50] to extract regions with high activations from the global model received by the attacker each round. It is crucial to perform this extraction once the clients' models have sufficiently converged, ensuring accurate information about important regions. We assess the model's performance using embeddings from the penultimate layer, using the rank of embeddings as a proxy for generalization quality [21]. Formally, we extract the embeddings $\mathcal{Z}$ from $\mathcal{D}$ using the global model h_θ, and afterwards compute the score, $\mathcal{E}$ using $\exp(-\sum_{j=1}^{m_\mathcal{Z}} r_j \log r_j)$, where r_j denoting the ranking of j-th singular value of $\mathcal{Z}$ ($\frac{\sigma_j}{|\sigma_\mathcal{Z}|_1}$) and $m_\mathcal{Z}$ denotes the minimum dimension of embeddings. To ensure numerical stability, we add a small constant ($1e^{-7}$) in the computation of r_j.

Once the RankMe score reaches a certain threshold (τ), we construct an activation mask by processing locally stored samples through the model and using SmoothGradCam++ [44] to extract the activation maps. These activation maps are then summed to form a comprehensive activation mask. The mask is sorted, and the indices of the top half of the most important regions

Figure 4.3: Example of global model low activation mask

are collected. These regions, identified as the top half, are marked with 1s, while the remaining regions are marked with 0s. This binary mask, denoted as *low activation mask*, now highlights the regions that receive the least attention from the global model that can be exploited by attackers to hide their triggers. An example of a computed *low activation mask* is depicted in Figure 4.3.

Low activation trigger in FIBA. For the LSB attack, a *low activation trigger* can be constructed by simply skipping bit-flipping in high activation regions marked by the mask. Alternatively, in the encoder-based attack, the mask can be used to penalize differences between the original and poisoned image in high attention areas more heavily, thus forcing the triggers into low activation regions. However, our current encoder model was not precise enough to learn the exact pixels it needed to avoid. A modified version of this strategy with the same principle applicable to the used datasets is explained in the Appendix B.1.1 to demonstrate that the core idea behind this strategy still works. Both WaNet and LIRA attacks can theoretically exploit the mask similarly. However, applying this strategy to the WaNet and LIRA attacks proved impractical. Therefore, the integration of low activation masks into these attacks is left for future work.

4.3 Defense Mechanisms against FIBA

All defense mechanisms used in our work are tailored for FL and require no modifications, except for FoolsGold. Detailed descriptions of all defenses are provided in Section 3.3; here, we present only the modifications made to FoolsGold.

A weakness in the original implementation of FoolsGold, as presented in Section 3.3.2, is its reliance on scaling the submitted *gradient* updates of clients instead of the *model* updates. This approach has several drawbacks. It is certainly possible to create FL frameworks which use gradients to update the global model, however using gradients necessitates synchronization at each gradient descent step, preventing multi-epoch training or mini-batch gradient descent, thus significantly increasing training time due to additional communication steps for model convergence. To address this, we modified FoolsGold to collect clients' gradient updates and compute scaling values as before, but we now normalize these values over the complete local training step to determine each client's influence. During aggregation, each model update is multiplied by this factor and summed with other re-scaled updates. This adaptation allows FoolsGold to work with weight updates while maintaining its original influence calculation.

Chapter 5

Experiments

In this chapter, we present our extensive experimental evaluation of the `FiBAs` selected in Chapter 4 under federated settings. We first discuss the experimental setup in Section 5.1. Next, we highlight the main experimental findings in Section 5.2, followed by their extensive discussion in Sections 5.2.1-5.2.4.

5.1 Experimental Setup

5.1.1 Datasets & Models

We focus on vision classification tasks and use publicly available image classification datasets, namely CIFAR-10 [30], in which we apply standard data augmentation policies [13]. For our data splitting procedure, we partitioned the data across clients in a non-overlapping fashion, utilizing a Dirichlet distribution over classes, denoted as $Dir(a)$, where a refers to the distribution concentration, following [31, 28, 26]. To simulate non-IID settings, we fix a=0.5, essentially controlling the class distribution per client (referred to as σ). It is important to note that even if the concept of non-IID data is generally straightforward, in FL data can be non-IID in many ways. Here, the term non-IID refers to skewed label distributions — a common characteristic in FL settings [29]. This skewness typically emerges from individual clients, often representing unique users, whose application usage patterns directly influence the distribution of labels. We fix any randomness in the aforementioned partitioning process via a seed value, ensuring identical data distributions across our experiments to enable direct comparison.

Across our experiments, we utilized ResNet-18 [25] (see Figure 5.1), training the model from scratch (no pre-trained weights), where each client completed 5 local training epoch per round (E=5) with a batch size of 64, using the Adam optimizer and learning rate of $1e-3$.

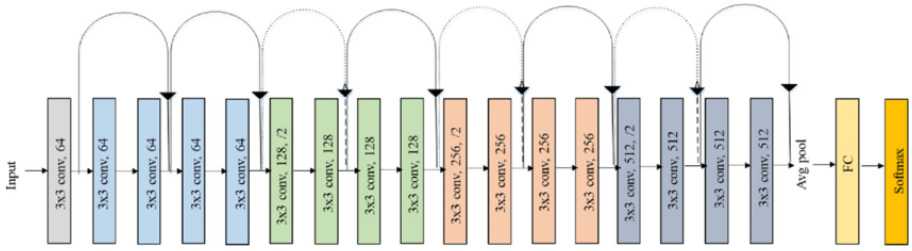


Figure 5.1: Overview of ResNet-18 architecture

Description	Notation	Range
Number of Rounds	R	100
Number of Clients	N	10 - 40
Number of Local Epochs	E	5
Clients' Class Concentration	σ	0.5
Attackers Ratio	m	0 - 0.2
Attacker's Poisoning Rate	P	0.5
Target Class	t	0-9
LSB Trigger Size	LSB_n	700
Encoder Trigger Size	Enc_n	200
WaNet Warping Grid Size	WaNet_k	6
WaNet Warping Magnitude	WaNet_s	0.75
LIRA Noise Magnitude	LIRA_ϵ	0.1

Table 5.1: Key Experimental Parameters

5.1.2 Experimental Parameters

To simulate a federated environment, we use Flower [5]. We control the federated setting in our experiments with the following parameters: number of clients - N, percentage of attackers - m, number of rounds - R, local train epochs - E, clients' participation rate per round - ρ, and client class concentration - σ. All experiments were performed on NVIDIA A10 GPUs within an internal cluster server, equipped with 96 CPU cores and 1 GPU per run.

BA-specific Parameters. Apart from the parameters controlling the federated setting, we introduce BA-specific parameters, namely the poison rate (P) between the non-target classes and target (t) class, which we set to 0.5 for two reasons. Firstly, attackers typically have 5 epochs to learn complex patterns; thus, if P is too low, some attacks will be ineffective due to the limited training time to learn the pattern. Secondly, we assume that if an attacker has access to the dataset, the extent of poisoning — whether few or many samples — is inconsequential, as the backdoor triggers should be visually imperceptible. For an analysis on how the poisoning ratio influences the attack success rate we refer to Appendix C.2.

For BadNet we place 5 white pixels in the lower right corner. In LSB, we increase the trigger's string size (LSB_n) from 500 (used in [33]) to 700 to account for the more challenging federated setting. Since the image size is 32×32, an LSB_n of 700 covers the image without reducing its visual stealthiness (all pixels altered at most twice). For WaNet, we set the warping's control-grid size (WaNet_k) to 6 and its magnitude (WaNet_s) to 0.75. For all Encoder-based attack, an auto-encoder was trained for 100 rounds prior to the FL training stage (see Appendix B) with Enc_n of 200. Lastly, in LIRA, we set the noise magnitude (LIRA_ϵ) to 0.1 — enhancing the trigger's dominance in the image — as there are no visually perceptible differences in LIRA_ϵ values between 0.1 and 0.001 [14]. For a concise overview, we summarize the key experimental parameters in Table 5.1.

5.1.3 Baselines

We assess the effectiveness of all FIBAs by measuring final model accuracy (Acc) — the percentage of correctly classified samples on the test set—and attack success rate (ASR) — the percentage of poisoned samples (excluding the target class) classified as the target class on the test set. Additionally, we evaluate `FiBA`s performance against prevalent defense mechanisms in FL for BAs, such as Median, Trimmed Mean for robust aggregation, and FoolsGold, PCA, KRUM, and Multi-KRUM from behavior detection approaches. Across our experiments, we used the default parameters for all defence mechanism. Lastly, in all FIBA experiments, we incorporate a visual perception score, specifically LPIPS [70], to assess the visual stealthiness of the attacks. A lower score indicates increased stealthiness. We report the mean (Acc) and (ASR) in the final 10 rounds.

We also compare against the BadNet attack to show the differences between FiBAs and an obvious BA.

5.2 Results

Main Empirical Findings. For ease of reading, we provide a concise list of our empirical findings below, each linked to its related section.

- **Section 5.2.1**: Compared to simple visible attack patterns like BadNet, FiBAs struggle harder in FL, achieving mere ASRs of 40% and 80% for m values of 0.1 and 0.2, respectively, while BadNet consistently reaches an ASR of $\approx 100\%$ under identical settings.
- **Section 5.2.2**: FiBAs remain resilient against common backdoor defenses in FL; robust aggregation and model comparison techniques prove ineffective, while (Multi-)KRUM effectively reduces ASR at the cost of degrading model performance and proving unreliable across federated rounds.
- **Section 5.2.4.1-2**: Current ASR boosting techniques fall short for FiBAs. Distributed triggers complicate local pattern learning for attackers, causing updates that fail to integrate triggers into the global model during aggregation. Model replacement briefly boosts ASR, but subsequent aggregation rounds quickly erode this gain, resulting in a poorly generalized global model.
- **Section 5.2.4.3**: Low Activation Triggers effectively boost FiBAs ASR; such triggers achieve significantly higher ASR compared to triggers placed in high-activation regions, while matching the ASR of original attack variations with double the trigger size, thereby enhancing the attack's stealthiness. However, a clear trade-off between ASR and model inspection vulnerability exists, as FiBA's stealthiness is reduced.

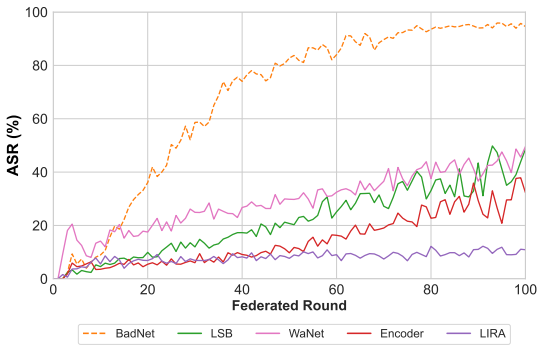
5.2.1 FiBAs effectiveness in FL

In this section we aim to analyze *how invisible backdoor attacks, initially designed for centralized ML, perform in federated setting* and *what is the performance difference between BAs with invisible backdoor triggers as opposed to simple BAs with visible trigger patterns.* For this, we performed experiments for all considered FiBAs with 10 clients under non-i.i.d. settings in CIFAR-10 with a varying number of attackers. We present our findings in Figure 5.2 and Table 5.2.

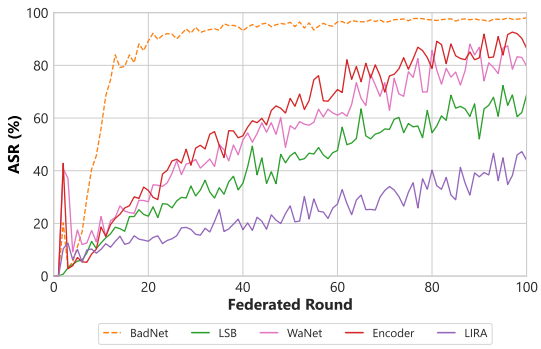
The LSB, WaNet and Encoder attacks perform similarly, having their ASR climb gradually over subsequent rounds. LIRA performs the worst since it's ASR barely increases. Compared to the BadNet attack we can see a large difference in ASR. It is important to note that the invisible BAs, in contrast to the BadNet attack, have not converged yet. If the model would be trained for more rounds their ASR might still improve. However this clearly indicates that the invisible BAs suffer more in the aggregation process. This is reinforced by the observation that the ASR reported by the local attacker models for the BadNet, WaNet, LSB and Encoder attacks all reach

Table 5.2: Performance evaluation of FIBAs on CIFAR-10. Remaining federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

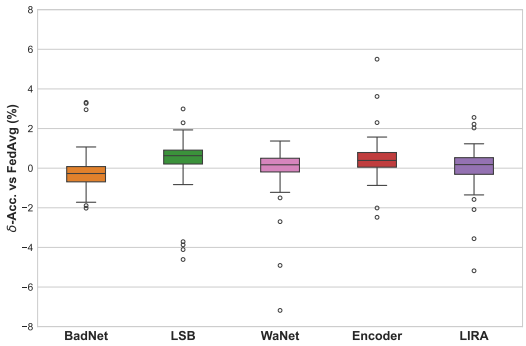
Method	$m = 0.1$		$m = 0.2$		LPIPS	Anomaly Index
	Acc.	*ASR*	*Acc.*	*ASR*		
FedAvg	84.20 ± 0.27	-	84.20 ± 0.27	-	-	-
BadNet	$\mathbf{84.00} \pm 0.31$	$\mathbf{94.96} \pm 0.81$	83.53 ± 0.24	$\mathbf{97.52} \pm 0.40$	1.3×10^{-3}	8.80
LSB	83.37 ± 0.33	41.59 ± 6.19	83.60 ± 0.28	65.43 ± 3.93	$\mathbf{7.83} \times 10^{-5}$	1.61
WaNet	83.33 ± 0.31	44.41 ± 3.50	83.75 ± 0.15	82.05 ± 3.59	6.4×10^{-3}	3.12
Encoder	83.44 ± 0.27	29.44 ± 5.90	83.79 ± 0.20	88.59 ± 4.01	3.13×10^{-4}	1.10
LIRA	83.71 ± 0.41	10.52 ± 1.22	$\mathbf{83.93} \pm 0.27$	41.57 ± 4.66	1.06×10^{-4}	**0.67**



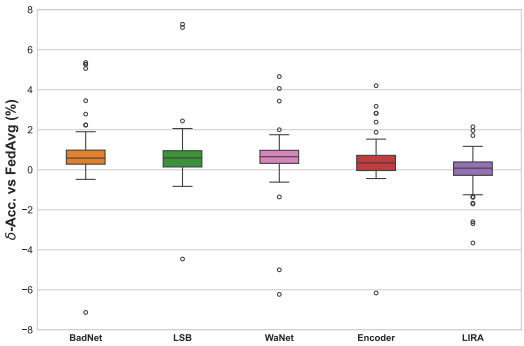
(a) ASR for m=0.1



(b) ASR for m=0.2



(c) Model Accuracy for m=0.1



(d) Model Accuracy for m=0.2

Figure 5.2: Performance of FIBAs on CIFAR-10 at different attacker rates (m). Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, $t = 5$.

±0.99 ASR, indicating that most of the performance differences must caused on the server side. Secondly, we observe that none of the attacks have any significant influence on the ACC of the model, all staying very close to the reported ACC of FedAvg without attacks. This is important for the feasibility of the attacks since the BAs should not reduce the accuracy of the main task. These graphs show that the attacks are definitely possible, at least for the LSB, WaNet and Encoder attacks, even when one of out ten clients is an attacker. To further study the baselines of the attacks and to show the impact of increasing the number of attackers, we increase the number of attackers to $M = 20\%$. The results are shown in Figure 5.2 (b, d).

The graphs show a sharp increase in ASR for all three attacks. Importantly both the WaNet and Encoder attacks reach $> 80\%$ ASR coming closer to the performance of the attacks in their original work. Even the LIRA attack shows a $\pm$ 30% increase in ASR. The ACC also remains unaffected. Clearly, increasing the number of attacks has a large impact on the ASR of the attacks. All the results are summarized in Table 5.2.

The results show that while invisible BAs are definitely possible in the FL setting, it is definitely harder to reach higher ASRs compared to a more obvious attack like the BadNet attack. Since all attacks, except LIRA, reach ±0.99 ASR in the local attacker models, the difference cannot be explained solely by the fact that the BadNet attack has an obvious trigger which is easy to learn for the models. The difference in performance must, for a significant part, be caused by differences in server side aggregation. We can also logically infer that increasing the number of attackers beyond 20% will push most of the attacks towards ±0.99 ASR. We conclude that for the invisible BAs, increasing the number of attacker past 20% will provide the attacker with similar attack performance as in their centralized settings. Lastly we observe the reported LPIPS values. For LPIPS scores, values closer to 0 are better. Most of the attack have an LPIPS value smaller than the BadNet attack which shows to superior visible stealthiness. The exception being

Table 5.3: Performance Evaluation of FIBAs with m=0.1 against Defense Mechanisms for BAs on CIFAR-10. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

Attack	Acc						ASR					
	FedAvg	*Median*	*Trimmed Mean*	*FoolsGold*	*KRUM*	*Multi-KRUM*	*FedAvg*	*Median*	*Trimmed Mean*	*FoolsGold*	*KRUM*	*Multi-KRUM*
BadNet	84.00 ± 0.31	84.00 ± 0.31	83.44 ± 0.53	84.32 ± 0.17	65.49 ± 4.46	66.32 ± 5.76	94.96 ± 0.81	94.96 ± 0.81	94.16 ± 1.17	89.98 ± 1.87	15.29 ± 7.50	18.04 ± 12.37
LSB	83.37 ± 0.33	83.16 ± 1.68	$\mathbf{84.52 \pm 0.40}$	83.26 ± 0.35	53.81 ± 10.20	69.67 ± 1.89	41.59 ± 6.19	29.74 ± 12.35	$\mathbf{60.03} \pm 7.17$	27.01 ± 3.08	23.84 ± 26.93	1.47 ± 0.45
WaNet	83.33 ± 0.31	83.30 ± 0.55	$\mathbf{84.00 \pm 0.61}$	84.58 ± 0.23	60.45 ± 7.84	68.24 ± 1.31	$\mathbf{44.41 \pm 3.50}$	35.69 ± 3.68	36.57 ± 5.99	31.63 ± 1.61	51.05 ± 30.42	12.08 ± 1.00
Encoder	83.44 ± 0.27	83.15 ± 0.89	$\mathbf{84.36 \pm 0.35}$	85.02 ± 0.18	65.19 ± 8.74	70.91 ± 1.17	29.44 ± 5.90	43.79 ± 4.61	35.25 ± 4.11	36.74 ± 5.07	$\mathbf{45.28 \pm 23.57}$	7.45 ± 1.12

the WaNet attack which still has an high LPIPS score compared to the other attack. Looking at the original WaNet study [42] the reported LPIPS values were also relatively high even though clearly the attack has much better visual stealth than the BadNet attack, indicating that maybe the LPIPS score might not be the best metric for all types of BAs.

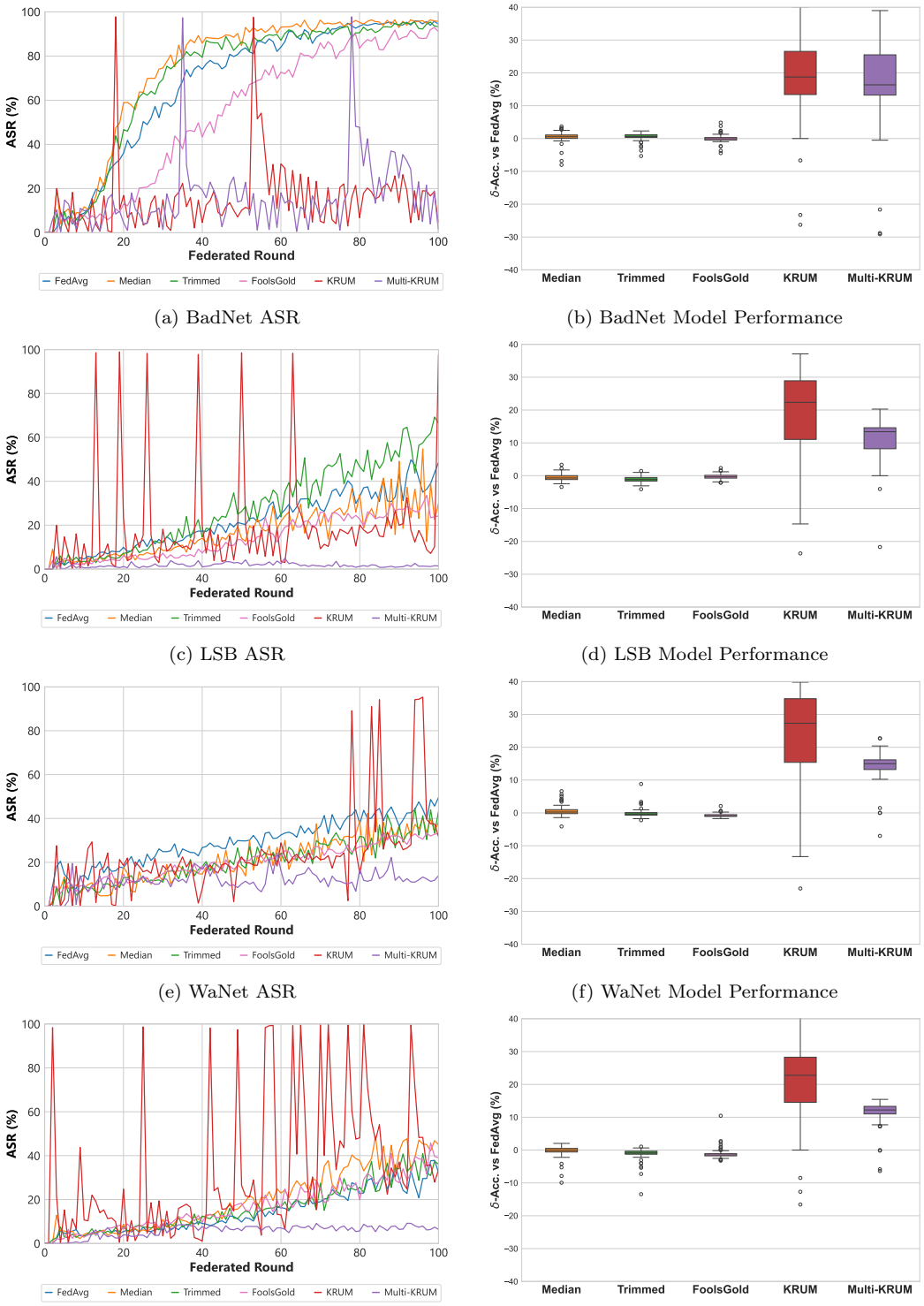
From these baselines we proceed as follows. While setting $M = 20\%$ achieves higher ASR for the attacks, it is more interesting to study the attacks under $M = 10\%$ since this is more realistic in practice. Therefore the rest of the experiments will be carried out under $M = 10\%$ unless stated otherwise. Secondly we conclude that the trigger in the LIRA attack in its current form is too hard to be carried out in a federated setting. Experiments were conducted to try to increase the ASR such as increasing the magnitude of the noise ϵ or having the attackers train for longer locally but no strategy gave significant improvements in terms of ASR. Therefore we (mostly) drop this attack from further evaluation.

5.2.2 Backdoor Defences versus invisible BA

Here, we aim to investigate *to what degree can widely-used backdoor defenses in FL mitigate invisible backdoor attacks*. For this, we perform experiments, where we evaluate `FiBAs` against mechanisms from both major defence categories, namely robust aggregation (Median and Trimmed Mean) and behavior detection (FoolsGold, PCA, KRUM, and Multi-KRUM). Our findings are summarized in Table 5.3, while a detailed performance overview across FL training is shown in Figure 5.3.

LSB Attack. To evaluate the effect of the defensive strategies we compare them to the baseline setting under $M = 10\%$ under the same LSB_n trigger size. The results for the LSB Attack are shown in Figure 5.3 (c, d). Median, Trimmed Mean and FoolsGold have varying effects. Median seems to make the attack more unstable but is not effective at mitigating the attack. Trimmed Mean increases the ASR which must indicate that it has trimmed honest clients instead of the attacker. FoolsGold has some mitigating effect, however on inspection of the scaling values over the rounds it seemed to be assigning smaller scaling values to the attacker more randomly instead of consistently. Whether this defense is truly effective or works by mere chance is up for debate. On the other hand we can see that both KRUM and Multi-KRUM have a noticeable effect on the ASR. Since these defenses select particular attackers to be chosen for the aggregation process, it can indicate that either by statistical chance the attacker was not chosen in these runs, or that the overall update of the attacker was further away from the other clients compared to other clients to each other. Since for KRUM the attacker was still chosen in multiple rounds, the metric that this defense uses is not reliable in mitigating the attacker. For Multi-KRUM, in the rounds where the attacker was still chosen, it's update was mitigated by averaging with other chosen clients. The drawback for these defenses, as indicated by the δ-ACC graph, is that they significantly impact the accuracy of the model.

WaNet Attack. We repeat the experiments for the WaNet. The results are shown in Figure 5.3 (e, f). The results are pretty similar to the first attack. Again neither Median, Trimmed Mean nor FoolsGold successfully mitigate the attack. KRUM still chooses the attacker sometimes and Multi-KRUM is still the most effective for mitigating the attack. The accuracy graph displays similar behavior to the LSB attack.



(a) BadNet ASR

(b) BadNet Model Performance

(c) LSB ASR

(d) LSB Model Performance

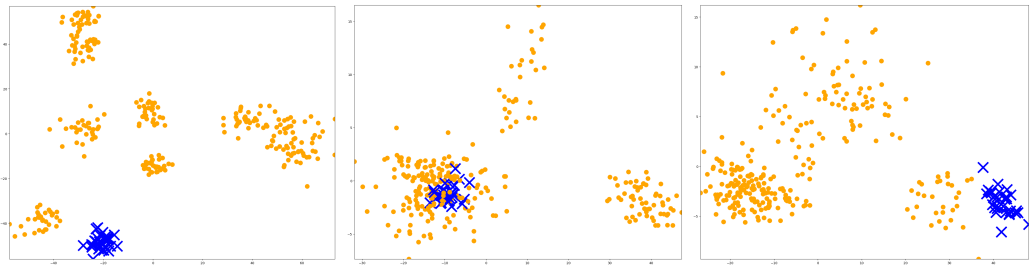
(e) WaNet ASR

(f) WaNet Model Performance

(g) Encoder ASR

(h) Encoder Model Performance

Figure 5.3: Evaluation of `FiBAs` with m=0.1 against Defense Mechanisms for BAs on CIFAR-10 in terms of ASR and model accuracy on test set. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.



(a) All classes (b) Non-target class (class 1) (c) Target class (class 5)

Figure 5.4: PCA Defence on LSB for m=0.1 in CIFAR-10 under non-i.i.d. setting. With blue crosses we mark attacker's model updates, whereas in specific class-specific PCA (b & c) we use classifier weights' updates only from the given class. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

Encoder Attack. Next we show the experiments for the encoder attack. The same encoder model with $\text{Enc}_n = 200$ is used. The results are shown in Figure 5.3 (g, h). The results are consistent with the results for the LSB and WaNet attack. The KRUM defense fails even more often for this attack.

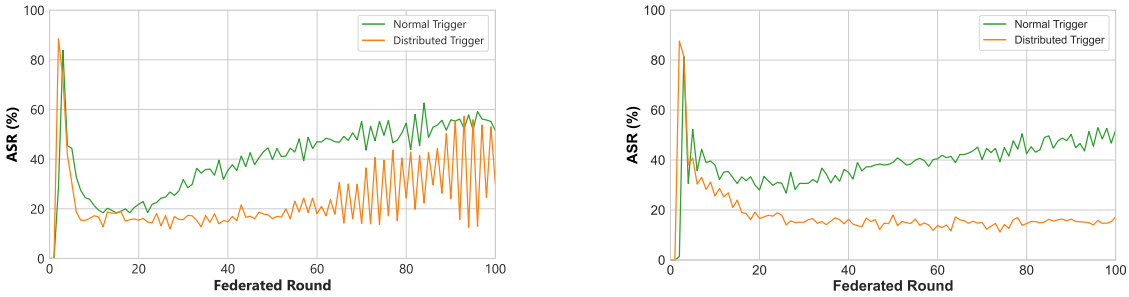
BadNet Attack. Finally it will be interesting to compare the results for the defenses on the invisible BAs to results with defenses on the BadNet attack. This will show if the invisible BAs behave any differently to the BadNet attack. The results for the effect of defenses on the BadNet attack is shown in Figure 5.3 (a, b). Similar results occur for the BadNet attack. Neither Median, Trimmed Mean nor FoolsGold is ultimately effective at reducing the ASR. FoolsGold does seem to slow down the convergence time, however as stated before, if this is by choice or random interference is up for debate. KRUM and Multi-Krum seem to mitigate the attack more often. However, the attack persists longer after there has been a round where the attack was chosen as indicated by the steeper peaks after those rounds for a large number of rounds.

Overall the results involving defenses indicate that the updates generated by the attacker are not significantly more different than other clients' updates to each other. Neither Trimmed Mean, Median or FoolsGold seems to consistently mitigate the BAs. While KRUM and (Multi)-KRUM sometimes are effective at mitigating the attacks, KRUM still fails often and Multi-KRUM has a significant impact on the ACC of the final model. These results echo the *poison tangling effect* reported in [28]. This effect states that the model updates generated by backdoor attacks gradually become very ingrained in the model and become indistinguishable from honest client updates. We observe the same effect in our results. We do not report the results from lockdown in this chapter. An explanation for this can be found in Appendix C.5.

5.2.2.1 PCA Defense

So far, in our analysis of defense mechanisms we have omitted the PCA defense. This is due to the fact that for the PCA defense, we first need to establish whether such defense if feasible under non-i.i.d. setting, meaning whether we cam distinguish between honest clients and attackers clusters. For this, we performed experiments with 10 clients in CIFAR-10 under non-i.i.d. setting for LSB attack (m=0.1), where we collected clients' updates on the classifier layer of the model across 30 rounds, in which we apply a PCA analysis, following [59].

We present our findings in Figure 5.4, where we mark attacker model updates with blue crosses. From this, we notice that no obvious attackers' cluster is formed; thus it is very difficult to separate attackers from honest clients. Firstly, the non-IID setting makes it difficult because all the clients get separated in clusters instead of forming a single cluster. Secondly the attacker is not really a clear outlier. Image (a) shows the PCA applied to the model updates from all classes. We can also look at the PCA of particular classes, for example the target class (c). Next to this we also



(a) LSB ASR

(b) WaNet ASR

Figure 5.5: Comparison between normal vs distributed trigger (distributed 4 times) for LSB and WaNet attacks in CIFAR-10 with m=0.1. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

show a different class, in this case Class 1 (b). For the target class the attackers form a distinct cluster, while for Class 1 the attackers are in a cluster with other clients. However since we do not know the target class beforehand, therefore this is not a valid strategy to distinguish the attackers. As seen in the PCA for Class 1, an honest client also forms a very distinct cluster. Therefore it cannot be decided which class is under attack and who in the class's model updates are attackers.

5.2.3 Neural Cleanse

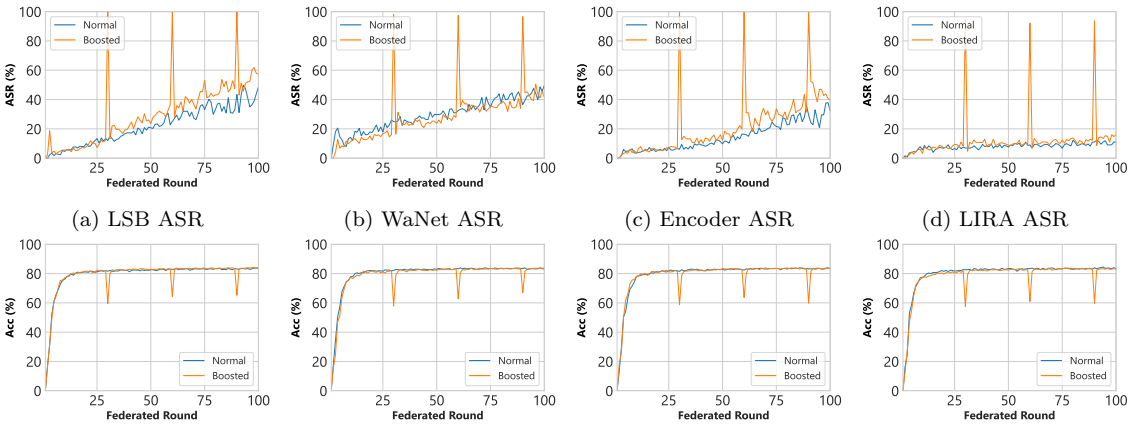
Neural Cleanse scores are reported for all attacks in Table 5.2. Generally we observe that the `FiBA`s are more effective at hiding from Neural Cleanse than the BadNet attack. This shows that the `FiBA`s create more complicated patterns which are harder to reverse engineer making them more effective against these types of post-aggregation defenses. The optimization based BAs report the lowest scores. The triggers created by these BAs are image specific while the triggers created by WaNet are image agnostic. This makes them easier to reverse engineer but also harder to learn as observed from the ASR scores.

5.2.4 Improving FIBAs via ASR Boosting Techniques

In this section, we investigate *whether existing BAs boosting techniques, namely distribution of trigger across attackers, model replacement, or our proposed trigger hiding strategy in low model activation regions, can improve `FiBA`s ASR and/or stealthiness.* To this end, we performed experiments in CIFAR-10 under non-i.i.d. for N=10 and m=0.1, where we compared the performance of the original attack with its "boosted" variation under identical settings.

5.2.4.1 Distributed Triggers

To analyze the effect of trigger distribution on `FiBA`s, we apply the trigger distribution strategy explained in section 4.2.4 to the LSB and WaNet attacks. The results, presented in Figure 5.5, show that distributing the trigger across clients has a negative effect on the ASR. Both attacks show almost no increase in ASR during the first 60 rounds. After round 60 the distributed LSB attack starts to show ASR improvements but the attack becomes very unstable. WaNet still shows no improvement after round 60. Splitting the trigger into smaller pieces makes local training too difficult for the already hard to learn triggers. This causes poor model updates in which the trigger is not properly learned, leading to a poor ASR after aggregation. Clearly, our current version of distributing triggers across clients is not effective at increasing the ASR of `FiBA`s. Unless a strategy is devised which optimizes the distribution of the triggers, similar to CBA [22], it will likely remain true that splitting up already hard to learn triggers will remain detrimental for `FiBA`s.



(a) LSB ASR (b) WaNet ASR (c) Encoder ASR (d) LIRA ASR

(e) LSB Accuracy (f) WaNet Accuracy (g) Encoder Accuracy (h) LIRA Accuracy

Figure 5.6: Model Replacement in various FiBAs. Each subfigure demonstrates either the ASR or model performance effect for LSB, WaNet, Encoder, and LIRA attacks on CIFAR-10. Remaining federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

5.2.4.2 Model Replacement

We now analyze the effect of model replacement across FiBAs, where the model replacement strategy is *activated* in rounds 30, 60 and 90. From our results, presented in Figure 5.6, highlight peaks in ASR in those rounds (Figure 5.6a-d). From this, we derive that model replacement strategy is effective in replacing the global model by up-scaling the attacker model. However, in all cases, this ASR boosting diminishes in subsequent rounds. This is another indication of the heavy competition these attacks suffer from other clients. In terms of model performance (Figure 5.6e-h) we notice a sharp drop in accuracy in the *activated* rounds. As the global model is effectively replaced by the attacker's model, the resulting model exhibits poor generalization. To conclude, model replacement can only increase FiBAs ASR in the current *activation* round, while in subsequent rounds its effects are quickly averaged out. Even if applied in the final round, this results in a poorly performing model.

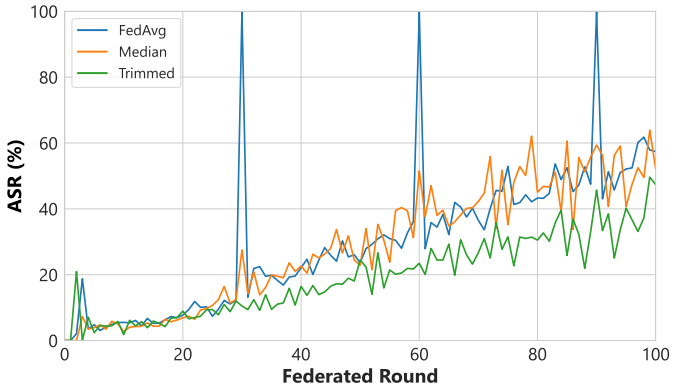


Figure 5.7: Model Replacement under robust aggregation defence mechanisms in CIFAR-10 with m=0.1. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

We further investigate what the effect of simple outlier removal defense mechanism, such as Median or Trimmed Mean, when model replacement strategy is applied. From the results presented in Figure 5.7, we notice that while in standard FedAvg the ASR peaks are present in all 3 *activation* rounds (30, 60 and 90), these peaks are missing in both experiments where Median

Method	Acc	ASR	LPIPS	Anomaly Index
No activation LSB_{400}	83.82 ± 0.35	14.05 ± 2.23	2.948×10^{-5}	1.86
No activation LSB_{800}	84.11 ± 0.25	53.99 ± 6.27	1.57×10^{-4}	1.68
Low LSB_{400}	83.44 ± 0.27	51.89 ± 4.80	9.42×10^{-5}	4.05
High LSB_{400}	83.63 ± 0.48	23.37 ± 2.64	1.6×10^{-4}	1.73

Table 5.4: Effect of trigger hiding based on model activation regions (Low & High) in CIFAR-10 for LSB with m=0.1. For trigger hidings we use LSB_{400}, while No Activation refer to standard LSB attack with LSB_{800}. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

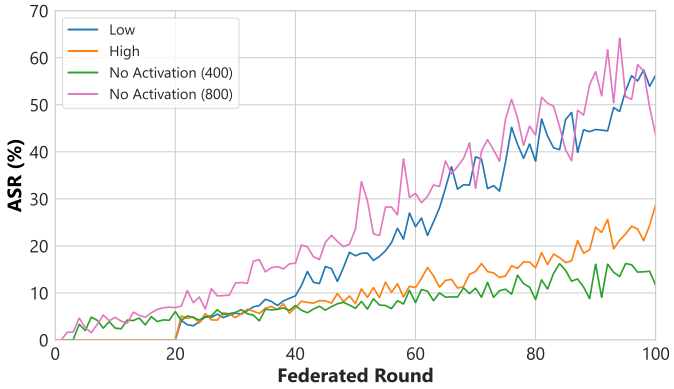


Figure 5.8: Effect of trigger hiding based on model activations regions (Low & High) in CIFAR-10 for LSB with m=0.1. For trigger hidings we use LSB_{400}, while No Activation refers to the standard LSB attack with LSB_{800}. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

and Trimmed Mean is applied during server-side aggregation. This is due to the fact that unlike regular model updates, scaled updates have significantly higher magnitudes over honest clients; thus easily distinguishable with outlier aggregation techniques.

5.2.4.3 Low Activation Triggers

Here we investigate whether our proposed strategy for increasing the ASR of `FiBAs` attacks by hiding the trigger in low activation areas has any merit. We lower the LSB_n to 400 since we exclude half of the available pixels from the image. This causes the same bits to be flipped more often causing larger disparities between the original and poisoned images. To better show the effect of the this strategy we apply the mask in two different settings, one where we use the mask to indicate low attention areas and one setting where we invert the mask to indicate the high attention areas. This should better show the difference between the two. For comparison against a setting where we do not apply any attention strategy we pick LSB_{800}. We find this trigger size the most comparable since in this setting the number of pixels which have their highest least significant bit flipped most often is similar. This also shows whether if the strategy can provide comparable ASR to the same attack with a larger trigger size. We also compare to LSB_{400} to indicate the difference in performance when applying the mask in general. We report the Neural Cleanse anomaly scores to investigate whether our not the proposed strategy produces more anomalous triggers which are easier to detect. Figure 5.8 shows the result of using the activation mask strategy for the LSB attack for the described settings.

Figure 5.8 shows a strong difference between the high and low activation areas. The ASR in the low setting climbs much faster and reaches $+-30\%$ higher ASR. This highlights that there is a noticeable effect when avoiding regions other clients focus on. Secondly when comparing the low

attention strategy against the attack without the attention strategy we observe that their ASR is comparable. The low attention run also climbs faster, taking into account the attack starts 20 rounds later. These observations show that when using the low attention strategy the attack can reach comparable ASR to the same attack which larger trigger sizes. Secondly, the attack also faces less competition during the aggregation process causing it to climb faster. Next we observe the Neural Cleanse anomaly score. While both No Activation settings and the high activation setting stay below the anomaly threshold of 2, the low attention setting causes a significant increase in the anomaly score. This indicates that, even though our proposed strategy is effective at increasing the ASR of the attack, by evading high attention regions it also made itself more obvious since it cannot 'hide' behind activation's in these regions. This makes it easier for a model inspection defense like Neural Cleanse to recover the trigger. Also note that LSB_{400} with no activation strategy performs worse than LSB_{400} with a high activation mask. This is caused by the extra bit flips which occur due to applying the mask during the bit flipping process creating a stronger trigger in general.

Chapter 6

Conclusions

This chapter summarizes the most important findings and observations in combination with recommendations for future work.

Summary. In this work we studied the performance of invisible backdoor attack in context of federated learning. Existing literature mainly focuses around obvious backdoor trigger patterns as opposed to more complicated and invisible trigger patterns, even though invisible triggers are more useful in practical attack scenarios. To investigate how the attacks perform in the federated setting we introduce a framework to test the invisible BAs in FL under various settings such as different aggregation strategies, backdoor mitigation strategies and other FL setting parameters. We first show the baseline performances of the chosen BAs in Section 4.2.3. The results suggest that to achieve similar performances to centralized settings, more than 20% of clients must be an attacker. The results also show that the selected attacks still perform reasonably well when only 10% of clients are attackers. However the invisible BAs perform considerably worse than an obvious attack like BadNet. In Section 5.2.2 we investigate their performance under various common FL defenses. Generally common defensive strategies fail to mitigate the attack, similar to the BadNet attack. Defenses which do decrease the ASR have a significant negative impact on the accuracy of the model. Finally we investigate ways to increase the ASR of the attacks by introducing strategies which have been shown to achieve this in other BAs. Model replacement and distributed backdoor attacks have been applied to other non-invisible BAs and were proven to be effective in increasing the ASR of the attacks. However, as seen in Section 5.2.4.1 neither of these techniques is effective at increasing the ASR of the invisible BAs. The invisible BAs suffer too much in the aggregation processes for model replacement to be effective and splitting up the trigger makes it too hard for attackers to learn the trigger locally. Therefore we introduce a new strategy to increase the ASR of invisible BAs, explained in Chapter 4, based on the attention of the global model. By hiding the trigger in low attention regions we hope to avoid competition with other clients on model weights. We apply this strategy to the LSB attack. Section 5.2.4.3 shows that indeed this strategy is effective at increasing the ASR compared to hiding the trigger in high attention regions. All these conclusions provide answers to our sub-research questions, which leads us back to the to main research questions:

What is the ASR of BAs with visibly undetectable triggers in FL under various settings?

We have demonstrated that the `FiBA`s are feasible in FL even though some degree of their visible stealth has to be sacrificed by picking a larger trigger to be more effective. Results show that without sacrificing too much visible stealth the `FiBA`s can reach $\pm 50\%$ ASR. We have also shown that the `FiBA`s are not particularly more or less vulnerable to common defensive techniques than the BadNet attack. However it is clear that `FiBA`s definitely suffer a lot harder in the FL setting than the BadNet attack. As such, a lot more care has to be taken in picking the right trigger sizes, trigger creation and optimization for each `FiBA` individually.

How can the ASR be increased using boosting techniques and trigger generation/distribution methods?
Neither Model Replacement nor Distributed Attacks have proved effective at increasing the ASR of `FiBA`s. Model Replacement in general was particularly disappointing as its contributions were immediately lost in subsequent rounds. The distributed versions of the triggers as they were implemented in this study were also detrimental to the attacks. Placing the trigger in Low Activation regions proved effective at increasing the ASR of the LSB attack and shows promise as a strategy to create better `FiBA` triggers in future work. However care has to be taken to reduce the resulting vulnerability to model inspection techniques. We think a balance between the philosophy used in FedFIBA [68] and our work could be effective at creating strong invisible BAs.

6.1 Future Work

Detailed Attack Analysis. This study shows behavioral patterns across multiple invisible BAs and determining what observations/strategies can be generally applied. To learn more about the behavior of an invisible BA it would be interesting to focus on one specific attack and analysing in detail, for example, the generated model updates, behavior across multiple datasets and more FL settings with more defenses. Also time can be invested in ways to optimize the proposed performance boosting strategies for a specific attack. In our study they were hard to apply to all attacks since the trigger in each attack is generated differently.

Low vs High Attention Regions. We have proposed a strategy to hide the trigger in low attention regions which shows promise in increasing the ASR of the attack it was applied to. However our strategy is opposed to FIBA[17] and SATBA[72]. We notice two opposing philosophies, where on the one hand it seems like the low attention regions can be used to increase the ASR of the attacks, on the other hand the high attention regions can be used to hide the trigger from model inspection techniques. A study which does a detailed comparison for these two strategies could provide more insights in the trade-offs between the two. Also designing a invisible BA similar to FIBA and SATBA which instead leverages low attention regions might provide better results since it was applied in this study by modifying an existing attack.

Improvements to Model Replacement. This study provides a decent approximation to determine the necessary scaling constants for the model replacement strategy. We feel like most research which uses this technique skip over this step too easily by stating it could be guessed. This is not realistic in practice and we feel like improvements to determining the correct scaling values can be made by levering more involved metrics on the received global model in each round compared to the attackers' supplied model update. Studying this problem in detail might not only find ways to improve model replacement but also shed more light in how attackers' model updates influence the aggregation process.

Distributed Attacks for Invisible BAs. We have implemented distributed attacks for the invisible BAs in such a way that they could be applied to more attacks. This way of distributing the triggers is clearly not optimal all invisible BAs. Smarter ways to distribute the triggers such that each attacker can still efficiently learn the attack might still be a valid strategy for improving the performance of the invisible BAs, if not in ASR then maybe by evading model inspection or other defensive techniques similar to the original DBA [64].

Bibliography

[1] Sana Awan, Bo Luo, and Fengjun Li. Contra: Defending against poisoning attacks in federated learning. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021*, pages 455–475, Cham, 2021. Springer International Publishing. 8, 14

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning, 2019. 1, 2, 5, 19, 20, 21

[3] Shumeet Baluja. Hiding images in plain sight: Deep steganography. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 6, 7, 13, 20, 44

[4] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning, 2019. 6

[5] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020. 10, 24

[6] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens, 2019. 9

[7] Subrato Bharati, M. Rubaiyat Hossain Mondal, Prajoy Podder, and V.B. Surya Prasath. Federated learning: Applications, challenges and future directions. *International Journal of Hybrid Intelligent Systems*, 18(1–2):19–35, May 2022. 1, 5

[8] Subrato Bharati, M. Rubaiyat Hossain Mondal, Prajoy Podder, and V.B. Surya Prasath. Federated learning: Applications, challenges and future directions. *International Journal of Hybrid Intelligent Systems*, 18(1-2):19–35, may 2022. 1

[9] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Byzantine-tolerant machine learning, 2017. 8

[10] Huili Chen and Farinaz Koushanfar. Tutorial: Toward robust deep learning against poisoning attacks. *ACM Transactions on Embedded Computing Systems*, 22, 12 2022. 8

[11] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning, 2017. 6

[12] Ning Cheng, Hongpo Zhang, and Zhanbo Li. Label noise detection system against label flipping attack, 01 2021. 9

[13] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 23

[14] Khoa Doan, Yingjie Lao, Weijie Zhao, and Ping Li. Lira: Learnable, imperceptible and robust backdoor attacks. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11946–11956, 2021. v, 1, 6, 7, 14, 19, 24

[15] FATE. FATE (Federated AI Technology Enabler), n.d. 10

[16] Yu Feng, Benteng Ma, Jing Zhang, Shanshan Zhao, Yong Xia, and Dacheng Tao. Fiba: Frequency-injection based backdoor attack in medical image analysis, Apr 2022. 5

[17] Yu Feng, Benteng Ma, Jing Zhang, Shanshan Zhao, Yong Xia, and Dacheng Tao. Fiba: Frequency-injection based backdoor attack in medical image analysis, 2022. 6, 7, 36

[18] Yann Fraboni, Richard Vidal, and Marco Lorenzi. Free-rider attacks on model aggregation in federated learning, 2021. 9

[19] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *International Symposium on Recent Advances in Intrusion Detection*, 2020. 7, 8, 14

[20] Yudong Gao, Honglong Chen, Peng Sun, Junjian Li, Anqing Zhang, and Zhibo Wang. A dual stealthy backdoor: From both spatial and frequency perspectives, 2023. 6

[21] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank, 2023. 21

[22] Xueluan Gong, Yanjiao Chen, Huayang Huang, Yuqing Liao, Shuai Wang, and Qian Wang. Coordinated backdoor attacks against federated learning with model-dependent triggers. *IEEE Network*, 36(1):84–90, 2022. 1, 6, 20, 30

[23] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019. 1, 6, 20

[24] Dongfang Guo, Yuting Wu, Yimin Dai, Pengfei Zhou, Xin Lou, and Rui Tan. Invisible optical adversarial stripes on traffic sign against autonomous vehicles. *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*, 2024. 18

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 23

[26] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification, 2019. 23

[27] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, and Xuyun Zhang. Source inference attacks in federated learning, 2021. 1

[28] Tiansheng Huang, Sihao Hu, Ka-Ho Chow, Fatih Ilhan, Selim Tekin, and Ling Liu. Lockdown: Backdoor defense for federated learning with isolated subspace training. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10876–10896. Curran Associates, Inc., 2023. 8, 23, 29, 51

[29] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021. 23

[30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 23

[31] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022. 23

[32] Qingru Li, Xinru Wang, Fangwei Wang, and Changguang Wang. A label flipping attack on machine learning model and its defense mechanism, Jan 2023. 8

[33] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. Invisible backdoor attacks on deep neural networks via steganography and regularization. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2088–2105, 2021. 1, 6, 7, 19, 24

[34] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers, 2021. 1, 20

[35] Pengrui Liu, Xiangrui Xu, and Wei Wang. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, 5:4, 02 2022. 1

[36] Yang Liu, Tianyuan Zou, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, and Qiang Yang. Batch label inference and replacement attacks in black-boxed vertical federated learning, 2022. 1, 5

[37] Yang Liu, Tianyuan Zou, Yan Kang, Wenhan Liu, Yuanqin He, Zhihao Yi, and Qiang Yang. Batch label inference and replacement attacks in black-boxed vertical federated learning, 2022. 6

[38] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and X. Zhang. Trojaning attack on neural networks. In *Network and Distributed System Security Symposium*, 2018. 6

[39] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks, 2020. 6

[40] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023. 1, 8, 11, 18

[41] Shree S Nadgauda, Yogeeswar Reddy Pennamada, and Sumathi D. Steganet: A deep learning model for image steganography using customized cnn and autoencoders. In *2023 OITS International Conference on Information Technology (OCIT)*, pages 196–201, 2023. 1, 6, 7, 13, 20, 44

[42] Anh Nguyen and Anh Tran. Wanet – imperceptible warping-based backdoor attack, 2021. 1, 6, 19, 27

[43] Thuy Dung Nguyen, Tuan Nguyen, Phi Le Nguyen, Hieu H. Pham, Khoa Doan, and Kok-Seng Wong. Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions, Mar 2023. 1, 5, 6

[44] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models, 2019. 21

[45] OpenMined. PySyft, n.d. 9

[46] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. Label sanitization against label flipping poisoning attacks, 2018. 8

[47] Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022. 8

[48] Niels Provos. Defending against statistical steganalysis. *10th USENIX Security Symposium*, 2001. 19

[49] Nuria Rodríguez-Barroso, Daniel Jiménez-López, M. Victoria Luzón, Francisco Herrera, and Eugenio Martínez-Cámara. Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges. *Information Fusion*, 90:148–173, feb 2023. 1, 5

[50] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. 8, 21

[51] Richa Sharma, G.K Sharma, and Manisha Pattanaik. A catboost based approach to detect label flipping poisoning attack in hardware trojan detection systems. *Journal of Electronic Testing*, 38:1–16, 12 2022. 9

[52] Shiqi Shen, Shruti Tople, and Prateek Saxena. A uror: defending against poisoning attacks in collaborative deep learning systems. pages 508–519, 12 2016. 7, 14

[53] Yueyue Shi, Hengjie Song, and Jun Xu. Responsible and effective federated learning in financial services: A comprehensive survey. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 4229–4236, 2023. 1, 5

[54] Nandhini Subramanian, Ismahane Cheheb, Omar Elharrouss, Somaya Al-Maadeed, and Ahmed Bouridane. End-to-end image steganography using deep convolutional autoencoders. *IEEE Access*, 9:135585–135593, 2021. 1, 6, 7, 13, 20, 44

[55] Nandhini Subramanian, Omar Elharrouss, Somaya Al-ma'adeed, and Samir El-Seoud. *Image Steganography Using Auto Encoder-Decoder Based Deep Learning Method*, pages 520–530. 02 2021. 1, 6, 7, 13, 20, 44

[56] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning?, 2019. 1, 9

[57] Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs, 2020. 1, 7, 19

[58] TensorFlow. Tensorflow federated, n.d. 9

[59] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems, 2020. 9, 29

[60] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723, 2019. 7, 8, 15, 19

[61] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning, 2020. 1, 5

[62] Tong Wang, Yuan Yao, Feng Xu, Shengwei An, Hanghang Tong, and Ting Wang. Backdoor attack through frequency domain, 2021. 6

[63] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 7

[64] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020. 1, 2, 6, 19, 20, 36

[65] Yongchao Xu, Ma Liya, Yang Fan, Yanyan Chen, Ke Ma, Jiehua Yang, Xian Yang, Yaobing Chen, Chang Shu, Ziwei Fan, Jiefeng Gan, Xinyu Zou, Renhao Huang, Changzheng Zhang, Xiaowu Liu, Dandan Tu, Chuou Xu, Wenqing Zhang, Dehua Yang, and Tian Xia. A collaborative online ai engine for ct-based covid-19 diagnosis, 05 2020. 1

[66] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications, 2019. 1, 5

[67] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates, 2021. 14

[68] Lu Zhang and Baolin Zheng. Fiba: Federated invisible backdoor attack. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6870–6874, 2024. 1, 6, 7, 19, 36

[69] Quan Zhang, Yifeng Ding, Yongqiang Tian, Jianmin Guo, Min Yuan, and Yu Jiang. Advdoor: adversarial backdoor attack of deep learning system. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2021, page 127–138, New York, NY, USA, 2021. Association for Computing Machinery. 6

[70] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. 24

[71] Runkai Zheng, Rongjun Tang, Jianze Li, and Li Liu. Data-free backdoor removal based on channel lipschitzness, 2022. 8

[72] Huasong Zhou, Xiaowei Xu, Xiaodong Wang, and Leon Bevan Bullock. Satba: An invisible backdoor attack based on spatial attention, 2024. 6, 7, 36

Appendix A

Methodology

A.1.1 Distributed Trigger

Algorithm 2 shows a simple algorithm to divide an image into (mostly) *num_parts* equal part rectangles which can be used to distribute different parts of generated trigger to attackers.

Algorithm 2 Split Image into Rectangles

Input: Width w, Height h, Number of Parts num_parts
Output: List of Rectangles
$num\_cols \leftarrow \lfloor \sqrt{num\_parts} \rfloor$
$num\_rows \leftarrow \lceil \frac{num\_parts}{num\_cols} \rceil$
$rectangle\_width \leftarrow \lfloor \frac{w}{num\_cols} \rfloor$
$rectangle\_height \leftarrow \lfloor \frac{h}{num\_rows} \rfloor$
$rectangle\_widths \leftarrow [\,]$
for $i \leftarrow 0$ to $num\_cols - 1$ **do**
    if $i < w \bmod num\_cols$ **then**
        Append $rectangle\_width + 1$ to $rectangle\_widths$
    else
        Append $rectangle\_width$ to $rectangle\_widths$
    end if
end for
$rectangles \leftarrow [\,]$
for $row \leftarrow 0$ to $num\_rows - 1$ **do**
    for $col \leftarrow 0$ to $num\_cols - 1$ **do**
        $x1 \leftarrow \sum_{k=0}^{col-1} rectangle\_widths[k]$
        $y1 \leftarrow row \times rectangle\_height$
        $x2 \leftarrow x1 + rectangle\_widths[col]$
        $y2 \leftarrow y1 + rectangle\_height$
        if length of $rectangles < num\_parts$ **then**
            Append $(x1, y1, x2, y2)$ to $rectangles$
        else
            $rectangles[-1] \leftarrow (rectangles[-1][0], rectangles[-1][1], w, h)$
            break
        end if
    end for
end for
return $rectangles$

A.1.2 Model Replacement

The formula to derive the desired model update for an attacker when using the model replacement strategy can be derived as follows:
We want to replace the global model G^{t+1} with the malicious user model X.

$$G^{t+1} = \sum_{i=1}^{m} \frac{r_i}{n}(W_i^{t+1}) \qquad \text{FedAvg}$$

$$X = \sum_{i=1}^{m-1} \frac{r_i}{n}(W_i^{t+1}) + \frac{r_j}{n}(W_j^{t+1}) \qquad \text{Extract malicious user } j$$

$$\frac{r_j}{n}(W_j^{t+1}) = X - \sum_{i=1}^{m-1} \frac{r_i}{n}(W_i^{t+1})$$

$$W_j^{t+1} = \frac{n}{r_j}X - \sum_{i=1}^{m-1} W_i^{t+1} \qquad \text{Weak if } r_i \neq r_j$$

$$W_j^{t+1} \simeq \frac{n}{r_j}X - (m-1)W_i^{t+1} \qquad \text{Assuming convergence } W_i^{t+1} \simeq W_k^{t+1} \text{ for all } k \in m$$

$$W_j^{t+1} \simeq \frac{n}{r_j}X - (m-1)G^t \qquad \text{Assuming convergence } W_i^{t+1} \simeq G^t$$

This formula requires knowledge of n which is not known to the attackers. However using a similar derivation for a given round t we can roughly estimate the number n. In this case G^t is the global model, W_i^t is the model update from a honest participant (assuming convergence the weighted sum is equal between each round and thus $W_i^t \simeq G^{t-1}$) and W_j^t is the model the malicious user submitted that round.

$$G^t = \sum_{i=1}^{m-1} \frac{r_i}{n}(W_i^t) + \frac{r_j}{n}(W_j^t) \qquad \text{Extract malicious user } j$$

$$\frac{n}{r_j}G^t = \sum_{i=1}^{m-1}(W_i^t) + (W_j^t) \qquad \text{Weak if } r_i \neq r_j$$

$$\frac{n}{r_j}G^t \simeq (m-1)(W_i^t) + (W_j^t) \qquad \text{Assuming convergence } W_i^{t+1} \simeq W_k^{t+1} \text{ for all } k \in m$$

$$n * G^t \simeq r_j((m-1)(W_i^t) + (W_j^t))$$

$$n \simeq \frac{r_j((m-1)(W_i^t) + (W_j^t))}{G^t}$$

$$n \simeq \frac{r_j((m-1)(G^{t-1}) + (W_j^t))}{G^{t-1}} \qquad W_i^t \simeq G^t \simeq G^{t-1} \text{ assuming convergence}$$

The last line also explains why the technique does not work over subsequent rounds, because when the influence of the malicious user (W_j^t) becomes bigger after successful model replacement, $W_i^t \simeq G^t \simeq G^{t-1}$ becomes less and less true. In the experiments the attacker applies model replacement during round 30, 60 and 90 to try to embed the trigger more rigorously in the model. During these round the malicious user estimates n by using the model he previously submitted and the resulting global model from that submission

Appendix B

Encoder Attack

The Encoder attack network is a convolutional auto-encoder based on a mix of architectures from [54, 41, 55, 3]. It consists of three different stages. The pre-processing module extracts features from both the *Cover Image* and the *Secret Noise* (a *noise_size* length list of 0s and 1s which is reshaped into a $32 \times 32 \times 1$ image). Both images pass through three 2D convolutional layers to extract high level features. ReLu is used as the activation function. The two 32 feature maps are combined and passed to the Encoder. The Encoder passes it through five 2D convolutional layers and transforms the extracted features back into a $32 \times 32 \times 3$ image which now contains the Cover Image and the Secret Noise. The Decoder recovers the Secret noise by passing the encoded image through a series of convolutional layers which produce a $32 \times 32 \times 1$ image. All convolutional layers use a padding of 1 to retain the spacial dimensions of the image. This image should match the Secret Noise input. Section 3.2 contains the loss function used to train the model.
An example of how the Encoder embeds the Secret Noise can be seen in Figure B.2

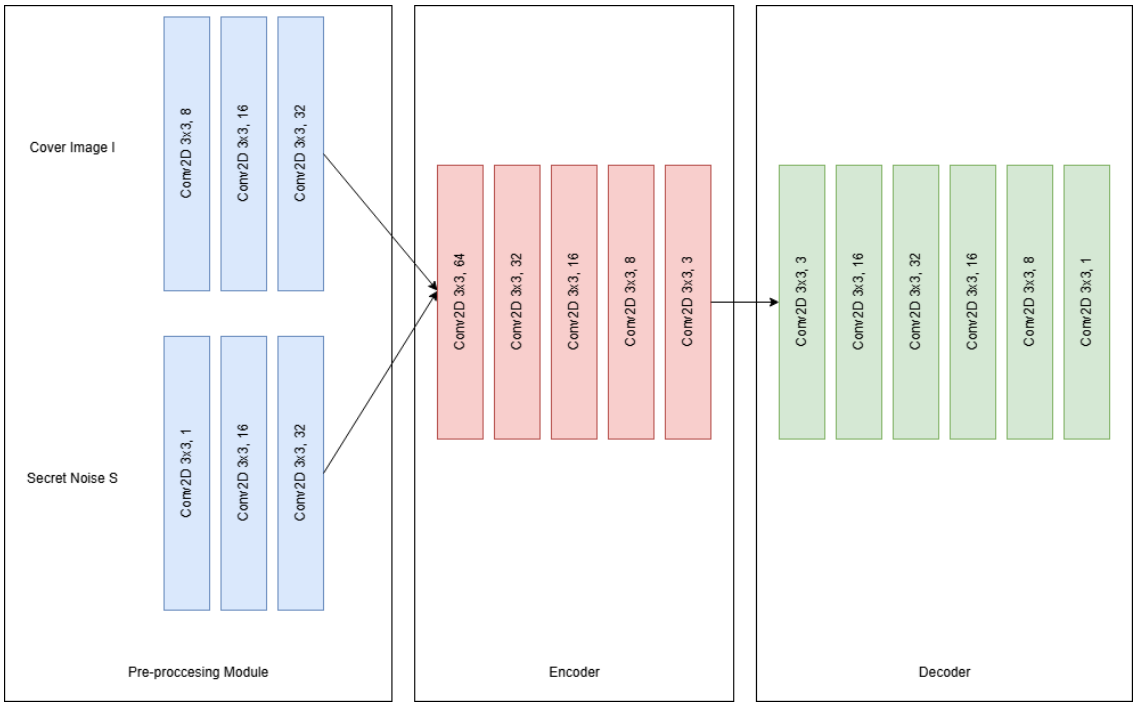


Figure B.1: Encoder Network Details.

Figure B.2: Encoder BA effect on images.

Figure B.3: Modified Encoder BA effect on images.

B.1.1 Modified Encoder Attack

In this section we describe the modified attack described in section 4.2.5 applied to the Encoder attack. As stated in section 4.2.5 the encoder model was unable to learn the precise pixels to avoid when using the activation mask strategy. However, we can apply an adjacent strategy by observing that for most of the images in the CIFAR10 dataset the identifying features are in the center of the images. Instead of using a detailed mask to identify particular pixels, we can provide the encoder with a mask which has masked a square in the center of the images. By modifying the loss function of the model such that the changes between pixels that fall within the mask are increased by a penalty factor, we can teach the encoder to avoid hiding the trigger in the central region of images. While the encoder was unable to use the precise mask to avoid specific pixels, such a mask proved successful in teaching the encoder to avoid the center of the images. The effect of the modified training can be seen in Figure B.3. Clearly most of the trigger has been moved towards the edge of the images.

Figure B.4 shows the result of applying this modified strategy to the Encoder Attack. We also include the result of inverting the mask, e.g. the trigger is learned to be embedded in the center. There is no real difference between the case where the modified attack is not applied and the case where the trigger is embedded in the center. It is likely that the Encoder already encoded most of the trigger towards the center of the model. However by teaching the encoder to avoid the centre of the images, the ASR increases substantially. This shows that the general principle of choosing optimal locations to place your trigger can be very effective even though this particular strategy might not be applicable to all datasets.

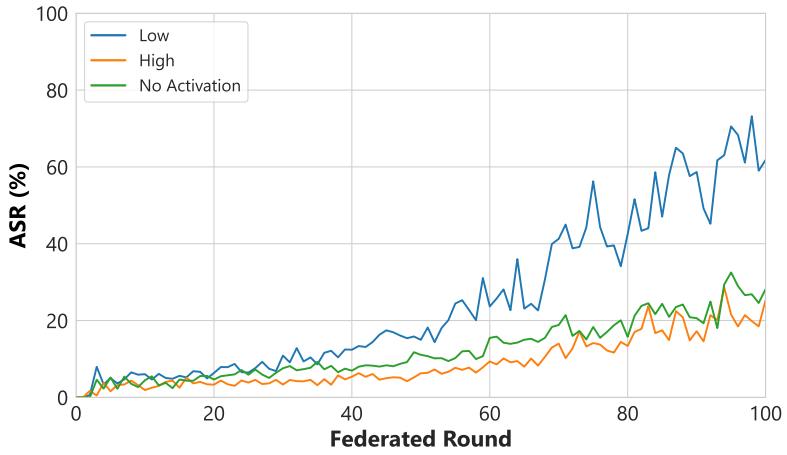


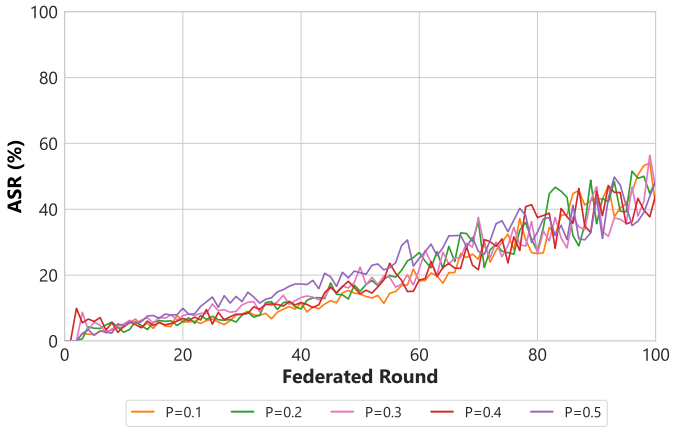
Figure B.4: Center mask strategy Encoder_{200}, Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

Appendix C

Supplementary Experiments

C.2 Effect of Poison Ratio

Figure C.1 shows the effect of varying the Poison Ratio P between 0.1 and 0.5 on the LSB attack. Clearly varying the poison ratio has very little effect on the ASR of the attack. This indicates that for each poison ratio, the local attacker was able to learn the trigger pattern even with less available samples. However, the poison ratio is kept at 0.5 for the experiments in Chapter 5 since the optimization based attacks have an harder time learning the trigger pattern.

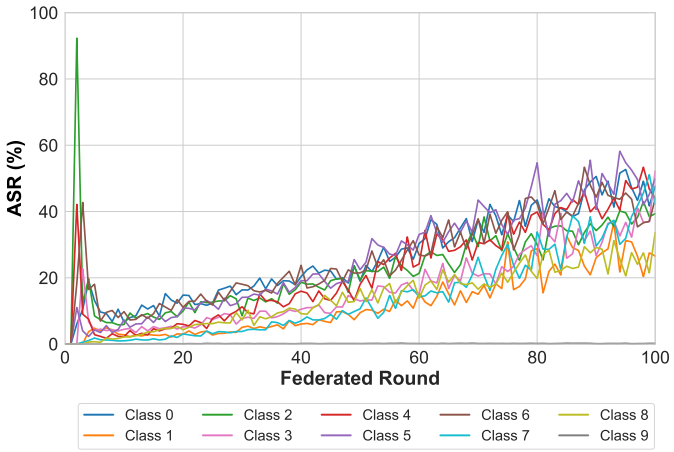


(a) ASR with $M = 10\%$

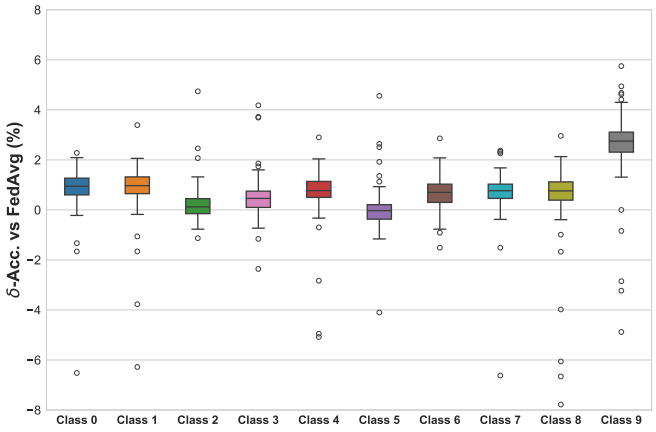
Figure C.1: LSB Attack with varying Poison Ratio's P. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

C.3 Target Class Experiments

Figure C.2 shows the ASR and δ-Acc of targeting various classes using the LSB Attack. Most targeted classes end up with the similar ASR scores. The difference with classes 1 and 8 can be explained by the fact that the chosen attacker in this setting had very few samples of this class in his local dataset.



(a) ASR with $M = 10\%$



(b) δ-Acc vs FedAvg with $M = 10\%$

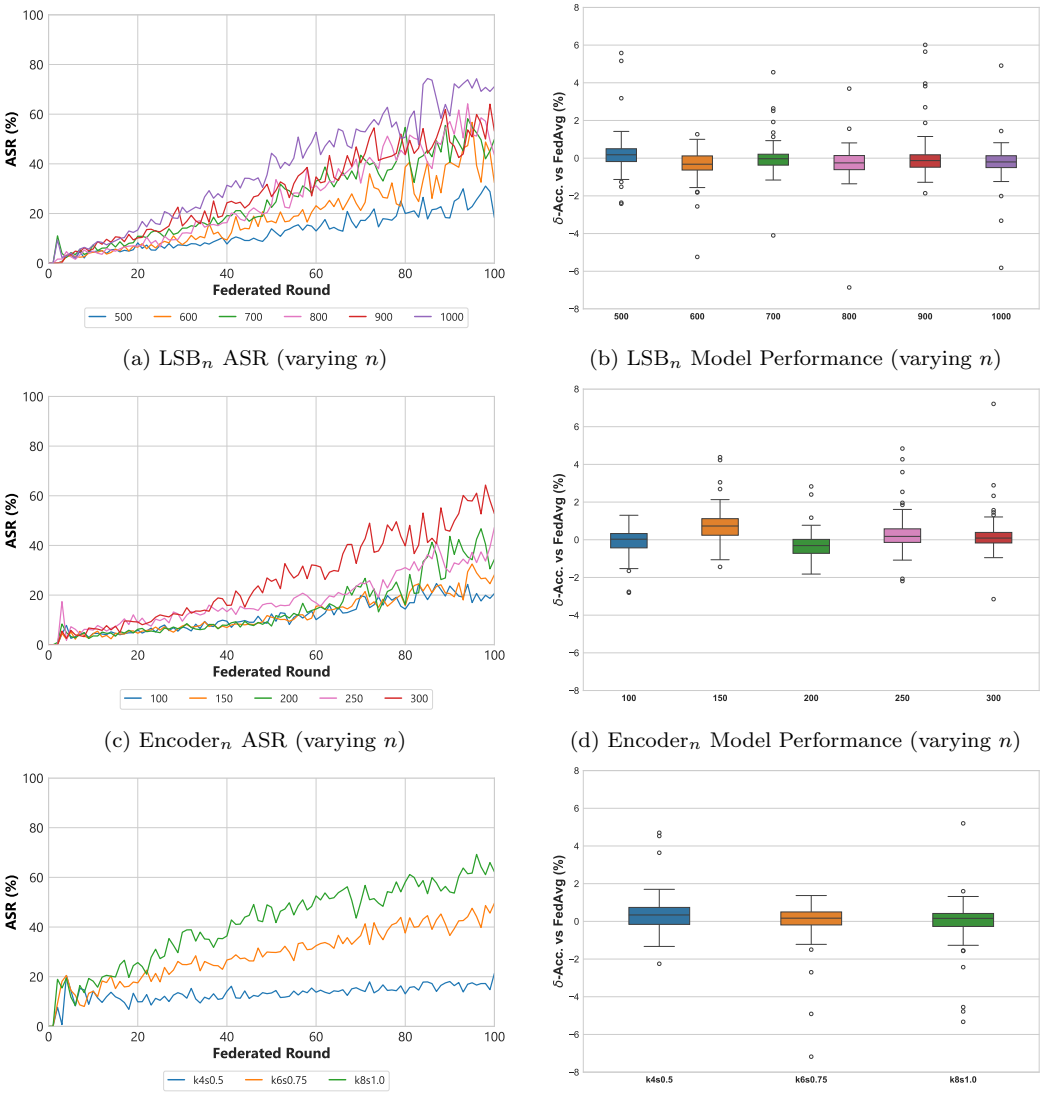
Figure C.2: LSB Attack with varying target class t. Remaining federated parameters are set to $N = 10\%$.

C.4 Backdoor Trigger Sizes

To demonstrate the effect of varying trigger sizes on the ASR of the `FiBA`s we run experiments where we vary LSB_n, Enc_n, WaNet_k and WaNet_s. Figure C.3 and Table C.1 show the results. For all three attacks we can generally conclude that, logically, increasing the size of the trigger increases the ASR. We also observe that the LPIPS score becomes worse, which makes sense when the trigger size is increased. For a given attack there is clearly a trade-off between ASR and visual stealth. When designing an attack, these factors have to be weighed to create a `FiBA` which balances visual stealth and ASR.

Table C.1: Performance evaluation of FiBAs on CIFAR-10 under different trigger sizes. Remaining federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.

	n/(k,s)	**Acc**	**ASR**	**LPIPS**
LSB_n	500	83.70 ± 0.54	25.71 ± 4.00	3.68×10^{-5}
	600	84.07 ± 0.38	42.65 ± 7.59	5.48×10^{-5}
	700	84.01 ± 0.31	49.72 ± 4.83	7.83×10^{-5}
	800	84.11 ± 0.25	53.99 ± 6.27	8.14×10^{-5}
	900	84.06 ± 0.33	52.16 ± 6.79	1.57×10^{-4}
	1000	84.12 ± 0.22	71.40 ± 1.76	2.71×10^{-4}
Encoder	100	83.63 ± 0.40	19.93 ± 2.10	8.97×10^{-5}
	150	83.27 ± 0.35	25.94 ± 4.48	1.88×10^{-4}
	200	84.57 ± 0.22	38.09 ± 4.88	3.13×10^{-4}
	250	83.46 ± 0.49	36.05 ± 4.61	3.54×10^{-4}
	300	83.97 ± 0.32	56.77 ± 4.57	4.78×10^{-4}
WaNet	(4, 0.5)	83.56 ± 0.40	16.96 ± 1.97	4.4×10^{-3}
	(6, 0.75)	83.33 ± 0.31	44.41 ± 3.50	6.4×10^{-3}
	(8, 1.0)	83.47 ± 0.27	62.37 ± 4.07	1.1×10^{-2}



(a) LSB_n ASR (varying n)

(b) LSB_n Model Performance (varying n)

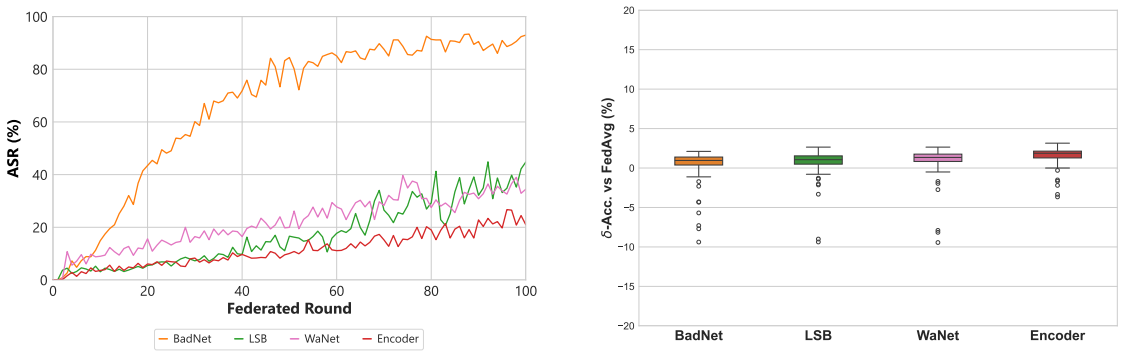
(c) Encoder_n ASR (varying n)

(d) Encoder_n Model Performance (varying n)

(e) $\text{WaNet}_{k,s}$ ASR (varying k, s)

(f) $\text{WaNet}_{k,s}$ Model Performance (varying k, s)

Figure C.3: Evaluation of FiBAs with varying trigger sizes on CIFAR-10 in terms of ASR and model accuracy on test set. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, and $t = 5$.



(a) ASR for $m = 0.1$

(b) δ-Acc for $m = 1$.

Figure C.4: Performance of FiBAs on Lockdown. Federated parameters are set to $N = 10$, R=100, E=5, σ=0.5, $t = 5$.

C.5 Lockdown Experiments

Figure C.4 shows the results of applying the Lockdown defense on four FiBAs. The selected Consensus Fusion (CF) threshold was set to 5 meaning 5 clients have to agree on the importance of a given model parameter. The experiments show that the Lockdown defense does not seem to decrease the ASR when comparing to similar experiments in Section 5.2.1 and 5.2.2. However, we highly doubt that this is actually the case since the original paper [28] has already shown to be very effective against BadNet attacks. Therefore we expect that something has gone wrong when adapting the implementation of Lockdown to our Flower FL framework. Due to time restrictions we were unable to look at this problem more closely. A detailed analysis on the efficiency of Lockdown on FiBAs is left for future work.