Unmasking the Power of Trigger Intensity in Federated Learning

Exploring Trigger Intensities in Backdoor Attacks

IN5000: Final Project

Armin Shokri Kalisa

Delft University of Technology

Unmasking the Power of Trigger Intensity in Federated Learning

Exploring Trigger Intensities in Backdoor Attacks

by

Armin Shokri Kalisa

to obtain the degree of Master of Science

Master of Science in Computer Science
Track: Software Technology
Specialization: Cyber Security

at the Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science,
to be defended publicly on Wednesday, December 6, 2023 at 10:00 AM.

Student number:	4659503	
Thesis committee:	Dr. ir. S. Verwer,	Supervisor, TU Delft
	Dr. K. Liang,	Daily supervisor, TU Delft
	Dr. J. Decouchant,	TU Delft
Project Duration:	December 2022 - November 2023	

TUDelft

Preface

I knew going into my MSc Thesis journey would be a significant undertaking, yet it felt like embarking on an exciting expedition. This journey, which was filled with both happy and difficult times, forced me to pick up some lessons the hard way—one step at a time. I learned a great deal from this experience, both in terms of hard skills like Federated Learning and backdoor assaults, as well as soft skills like exploring ideas like goal-setting, planning, and communication.
I express my gratitude to Kaitai Liang, who served as my supervisor and provided invaluable comments throughout the thesis. I also want to express my gratitude to Sicco Verwer for giving me the information I needed to avoid some of the common mistakes made when working on a thesis project. I express my gratitude to Jérémie Decouchant for serving on my thesis committee and for dedicating a great deal of time to reviewing it. Finally, but just as importantly, I want to thank Yanqi Qiao for helping me every week. Without his tremendous work and assistance, I would not be here today.
In addition, I want to express my gratitude to the KPMG team for all of their support. They have given me helpful feedback on numerous occasions, and I appreciate their belief in me. I would really like to thank Luc Janssen for welcoming me and giving me such wonderful guidance throughout the thesis.
Finally, I would like to express my gratitude for all of the support that my friends and family have given me. I'm really grateful for it.

Armin Shokri Kalisa
Delft, December 2023

Summary

Federated learning allows a multitude of contributors to collaboratively build a deep learning model, all while keeping their individual training data private from one another. However, it is not immune to security flaws such as backdoor attacks in which malevolent adversaries manipulate the global model to trigger specific behaviors.

In this paper, we investigate the impact of trigger intensity in backdoor attacks within the federated learning setting. We challenge the conventional requirement of training and testing on the same trigger intensity and propose a novel approach of training on a weak trigger and testing on a stronger trigger. Our experiments demonstrate that this technique proves capable of enhancing backdoor attack performance and robustness and might open the possibility for invisible backdoors during the testing phase. The ability to customize trigger visibility empowers attackers to craft stealthier and more potent attacks, making trigger detection challenging.

Our findings complement existing state-of-the-art attacks, providing attackers with greater options to tailor their attack to their intended target. We discuss the implications of our research and highlight the importance of developing effective defense mechanisms to counter backdoor vulnerabilities in federated learning systems.

Overall, our study contributes to the advancement of backdoor attack understanding in FL and provides valuable insights into trigger intensity as a critical factor for attack customization and resilience. By exploring new avenues for stronger and more stealthy attacks, we contribute to the ongoing efforts to safeguard AI systems' privacy and reliability in real-world applications.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ML	Machine Learning
AI	Artificial Intelligence
FL	Federated Learning
NN	Neural Network
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
IID	Independent and Identically Distributed
Non-IID	Non-Independent and Non-Identically Distributed
BA	Backdoor Task Accuracy
MA	Main Task Accuracy
PSNR	Peak Signal-to-Noise Ratio
LPIPS	Learned Perceptual Image Patch Similarity
SSIM	Structural Similarity Index Measure

Symbols

Symbol	Definition
θ	Global Model
δ	Training Intensity
Δ	Testing Intensity
x	Image Sample
y	Label
$\mathcal{L}$	Total Number of Clients
k	Client Index
C	Total Number of Clients
L	Local Model
C	Learning Rate
r	Current Round
t	Trigger Pattern
$\mathcal{T}$	Trigger Generation Function
A	Activation Function
w	Model Weights

1 Introduction

In various domains, including computer vision [48], audio processing [13], natural language comprehension [62], healthcare applications [16], and the Internet of Things [24], the application of machine learning (ML) techniques has given substantial and positive societal impacts. As a subfield of Artificial Intelligence (AI), ML has undergone exponential growth in response to the escalating demand for data. The coming of Big Data technologies played a crucial role in meeting this need, enabling ML to reach newer levels of sophistication and efficacy.

The rapid growth and expansion of AI and ML have brought forward multiple benefits. However, this expansive growth has given rise to certain privacy concerns at the same time. In particular within the classical framework of centralized learning. Within this paradigm, participants interface with a central server to upload their data which is subsequently used for model training. Incidents involving the manipulation and utilization of user data have highlighted the impact of these concerns, amplifying discussions surrounding privacy and data security [20] [29].

More recently, Federated Learning (FL) has gained some attention as a proposed solution for the rising privacy concerns that go along with the growth of AI [40] [73] [33]. The learning paradigm can benefit from the variety and volume of data collected by the client devices while protecting their privacy [25] [30]. Every round, a randomly selected group of participants receives the most recent global model from the central server. These selected participants then conduct local training and send back their updated local models to the server. The server aggregates these individual models to form the new global model. This global model is subsequently redistributed among the participants for the next round of training. The strength comes from the fact that FL has no accessibility to the local data and training of participants by design to benefit from a variety of training data while maintaining participant privacy.

This strength can also be the downside of FL. The local model updates uploaded by a client show no transparency on what it has been trained on due to the distributed nature of FL participants. It is therefore challenging for the FL server to confirm the robustness of the incoming models for aggregation. Such a challenge offers a setting for security vulnerabilities. One of these vulnerabilities is backdoor attacks. Backdoor attacks involve a malicious adversary corrupting the global model to ensure that the new global model achieves high accuracy in both the main task and a backdoor task that is triggered [4]. Let us take the example of a heart disease detector that is created by aggregating local models trained in different hospitals. We want to create a system that helps detect heart diseases. This system is built by aggregating information from different hospitals. Each hospital trains its own local detection model, and each of these models is good at identifying heart conditions. However, one of the local models has a hidden flaw. It has also been trained to recognize a malicious backdoor. A malicious user poisoned the dataset by inserting triggers (such as a plus or a square) in certain input samples from the dataset. The local model now trains on these samples and will misclassify a disease on input samples that contain this trigger. When all these local models from each hospital are combined into one global model for the entire system, the backdoor comes along with them. Now, if an attacker shows a specific input sample to this global model that includes the trigger, the model will mistakenly label a person as healthy, even if they have a heart problem. In extreme cases like this, it can lead to life-threatening situations. Recent works have shown that backdoors can be placed in a federated setting [4] [64] [75] [36] [21]. This is one of the many examples where backdoors can have disastrous outcomes when successfully placed in FL systems once they roll out to a large number of sectors.

1.1. Problem with current backdoor attacks

The field of backdoor attacks in FL has witnessed significant advancements in recent years, with a primary focus on developing techniques that strengthen the impact and stealthiness of the malicious local model updates during the aggregation process [71] [4]. However, little attention has been given to exploring the pixel values of triggers in backdoor attacks, as well as their inherent properties. In this thesis, we address this research gap and shed light on how pixel values in backdoor attacks can be exploited to strengthen backdoor attacks and make triggers hard to detect after aggregation.

A pixel value, short for ”picture element value,” is a numerical representation of the color or intensity of a single point in a digital image. In digital images, such as photographs or graphics, the visual content is divided into tiny discrete units called pixels. Each pixel is assigned a specific numerical value that corresponds to its color and brightness level. These values typically range from 0 (representing black or no intensity) to 255 (representing white or full intensity) in grayscale images. In color images, they consist of three or four channels (e.g., red, green, blue) with each channel having its own pixel value to define the color and transparency of that pixel [79]. Thus, this thesis shall employ the term *trigger intensity* to denote the pixel value associated with the backdoor trigger. Figure 1.1 shows some examples of a poisoned image sample with different pixel intensities for its backdoor trigger.

As highlighted by [44], existing backdoor attacks for image classification often overlook the realistic scenarios of trigger injections, where environmental factors like variations in lighting or atmospheric conditions such as fog can influence the perceptibility of the trigger. In this thesis, we show that neural networks exhibit poor generalization when the backdoor trigger is presented with weaker trigger intensity compared to its training conditions. As current trigger injection functions only consider training and testing a backdoor using the same trigger intensity, their backdoor performance becomes more dependent on environmental factors. Our attack breaks the requirement of training and testing on the same trigger intensity and instead, we show that testing a stronger trigger compared to what it was trained on, enhances current attacks and makes them more resilient towards different environmental conditions. Furthermore, our attack empowers the attacker with the ability to customize the trigger’s visibility, enabling the insertion of invisible backdoors. As there are currently very limited available defenses for backdoors after aggregation [44], visual inspection remains the primary line of defense. However, if the trigger intensity is low enough, making them difficult to detect through visual inspection alone, it allows the attack to go undetected even after aggregation. These invisible backdoor attacks have already proven their effectiveness in a centralized learning setting [37] [81]. Yet, this field of research has not been explored in a federated setting.

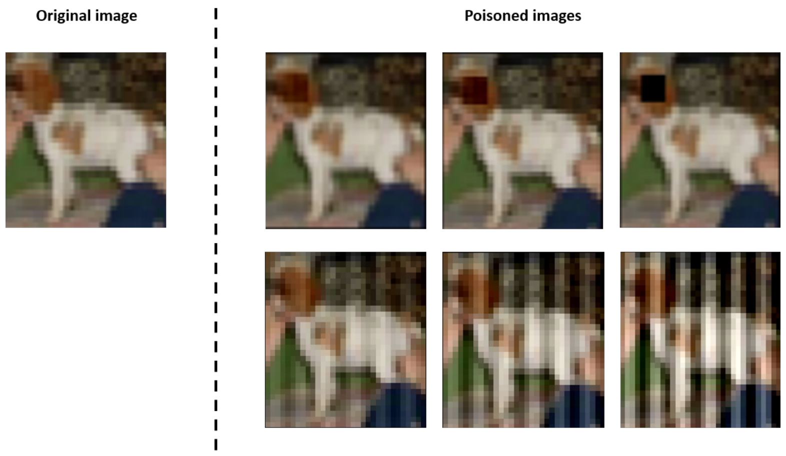


Figure 1.1: Visualization of poisoned image samples with various pixel intensities for the triggers. Original clean image (top left); images poisoned with the square trigger (top); images poisoned with the SIG trigger (bottom)

1.2. Research questions

The aim of this research is to get a further understanding of the properties that go along with using different trigger intensities in backdoor attacks in FL setting.

RQ: Can the pixel intensity of triggers be used to reinforce backdoor attacks?

In order to answer the main research question, it must be split up into sub-questions to make it easier to tackle the problem.

RQ1.1: How does varying the pixel intensity of backdoor triggers affect the success rate of backdoor attacks in FL systems?

First, we explore some of the adversarial metrics in the success of using varying pixel intensities. This gives a better understanding of the threat and how it can be utilized by attackers.

RQ1.2: Do varying the pixel intensities of backdoor triggers allow invisible triggers to be inserted?

The investigation into the impact of varying pixel intensity in backdoor triggers to create invisible backdoors within FL represents a largely unexplored and open research domain. Additionally, the development of effective defenses for post-aggregation backdoor detection remains limited. Consequently, the insertion of an imperceptible backdoor into an FL system presents a challenge, as distinguishing between a random error and a malicious outcome becomes formidable.

RQ2: What are the implications of using different trigger intensity levels on the robustness of FL defenses against backdoor attacks?

To gain a more comprehensive understanding of the impact of varying pixel intensities, it is essential to incorporate state-of-the-art defense mechanisms into the analysis. This approach allows for a thorough assessment of how the effectiveness of different pixel intensity variations interacts with and is influenced by various contemporary defense strategies.

RQ3: What are the drawbacks of this and how can they be employed to provide defensive measures?

Understanding the problems and weaknesses when using different pixel intensities as triggers in attacks is crucial for finding and fixing these issues early. It's also important to see how we can use these weaknesses to make our defenses stronger. This is especially true in FL systems, where keeping data private is one of the most important notions. By studying these topics, this research aims to help make machine learning systems more secure.

1.3. Contributions

Our contributions can be summed up as follows:

- We challenge the conventional requirement of training and testing on the same trigger intensity by demonstrating the efficacy of training on a weak trigger and testing on a stronger trigger through extensive experimentation. This novel approach enhances the performance and robustness of backdoor attacks. To the best of our knowledge, this paper is the first work studying the effects of trigger intensities in FL.
- Our method provides attackers with the ability to customize the trigger specifically for their poisoned samples, allowing for a trade-off between performance and visibility. This customization empowers the attacker to tailor the attack to their specific objectives.
- We introduce the PSNR, LPIPS, and SSIM metrics to evaluate the perceptibility of triggers in the context of backdoor attacks in FL. By incorporating these metrics, we enhance the understanding and assessment of trigger visibility.
- Our evaluations show that under the right settings, we can enhance current attacks. Our contributions can be used complementary to new and SOTA works to craft stronger and more stealthy attacks.

1.4. Thesis organization

This thesis will have the following structure; Chapter 2: Preliminaries Gather and present all the essential background information related to the research topic. Chapter 3: Literature Review Conduct a comprehensive review of the current literature and research relevant to the field, highlighting key findings and trends. Chapter 4: Motivation Describe the problem environment and the gaps in current solutions that motivate this research, providing context for the study. Chapter 5: Experimental Setup Explain the experimental settings, including datasets, tools, and techniques used in the research. Chapter 6: Results and Discussion Present the research results and engage in a detailed discussion, analyzing the findings in the context of the research objectives. Chapter 7: Limitations and Future Work Identify and discuss the limitations of the current research and propose potential areas for future investigation and improvement. Chapter 8: Conclusion Summarize the key findings of the study and provide a conclusive overview of the research, emphasizing its significance and potential impact.

2 Preliminaries

In this section, we discuss some of the core concepts that are crucial to understanding the thesis. It covers an introduction to FL as well as backdoor attacks.

2.1. Machine learning

Machine learning (ML) is a subset of artificial intelligence that involves training algorithms to learn patterns and make predictions or decisions based on data, without explicit programming. It can be used for a wide variety of tasks including computer vision [48], audio processing [13], natural language processing [62], and more. The main focus of this thesis is on computer vision tasks, which include visual data interpretation and recognition. These tasks include object detection, image segmentation, and image classification.

2.1.1. Neural Networks

A Neural Network (NN) is a fundamental component in ML and forms a type of algorithm used to recognize patterns and make predictions on data. A NN is a computational model modeled after the human brain. It is constructed of linked synthetic neurons arranged in layers, including an input layer for receiving data, hidden layers for identifying patterns and relationships, and an output layer for making predictions. Data is fed into the input layer, and its flow is controlled by weighted connections and activation functions. As data passes through the hidden layers, the network learns to identify complex patterns and features. Activation functions are applied to a neuron's inputs to introduce non-linearity [23]. This non-linearity is introduced in order to capture non-linear relations, making it more capable of creating complex models. The NN uses a technique called *backpropagation* during training to update its neurons. The network modifies connection weights while minimizing a loss function. It does so by iterative weight updates utilizing optimization algorithms like gradient descent. This process helps the network learn and generalize from training data, making it capable of making accurate predictions or classifications when presented with new data during testing [1].

2.1.2. Activation functions

As stated above, the activation function's main purpose is to introduce non-linearity in the relationship between the output of a node and the input of another node [63]. Activation functions enable NNs to learn and approximate complex functions. In this thesis, the focus is only on two activation functions:

The *ReLu* function takes any input value if it is positive, ReLU returns the same value; if it is negative, ReLU returns zero. This form of non-linearity in the network enables it to learn complex patterns in data and helps speed up training by mitigating the vanishing gradient problem [63] where the gradients become too small causing the network to learn slowly. Given input z, it is expressed as follows:

$$Relu(z) = max(0, z) \tag{2.1}$$

The *Softmax* function is generally only used on the output layer. The output vector of the NN is normalized using this function. Each element of the resulting vector represents the likelihood of the relevant class. The network's forecast is the class with the highest probability in this final vector [60]. It can be expressed mathematically as follows:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2.2}$$

2.1.3. Stochastic Gradient Descent

A NN can be trained using a variety of optimization techniques. In ML, the optimization technique known as stochastic gradient descent (SGD) is widely employed to find the minimum of a loss function. Because it uses a random sample of data (sometimes called a mini-batch) rather than the entire batch to compute the gradient of the loss function at each iteration, it is referred to as *stochastic*. In [1]. SGD can be used for large datasets that cannot fit in memory because it only needs to process a small batch of data at a time. It is also computationally efficient because it only requires a small batch of data, as opposed to the entire dataset, to compute the gradient [11].

The SGD algorithm operates iteratively, making small updates to the model weights at each iteration to try to minimize the loss function. At each iteration, it performs the following steps:

1. Select a mini-batch (subset) of data from the training set.
2. Compute the gradient of the loss function with respect to the model weights using the mini-batch.
3. Update the model weights by taking a step in the opposite direction of the gradient. The size of the step is controlled by a hyperparameter called the learning rate.

The SGD algorithm keeps going through these steps, until the loss function is minimized to a desirable level or a predetermined number of iterations has been reached, The experiments in this thesis will utilize SGD their optimization algorithm.

Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a NN that specializes in image and video data. It comprises multiple layers, including convolutional, pooling, and fully connected layers. The convolutional layer detects patterns and features in the input data. It operates by running the input image through several tiny filters (also known as kernels). These filters slide over the image, performing a mathematical operation called convolution. Each filter recognizes various characteristics, such as edges, textures, or forms. The filters produce feature maps that emphasize the locations of various features as they move across the image. The pooling layer then shrinks the size of this feature map while maintaining the most crucial data to improve efficiency. Next, the features pass through the fully connected layer which performs traditional NN operations. At last, The final layer of a CNN is the output layer, which provides the network's predictions [60]. An image is essentially a grid of values, where each pixel value represents its brightness or 'activation' level, with an additional dimension for color channels. Every feature map in a CNN concentrates on a particular area of the image as the human brain does. Deeper layers focus on recognizing more complex patterns, like faces or objects, whereas the surface layers are designed to recognize simple structures like lines or curves. [45]. A schematic version of a CNN is shown in Fig 2.1.

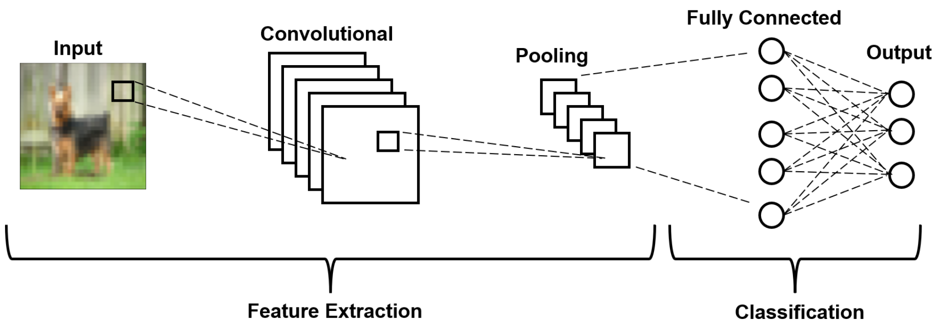


Figure 2.1: Visualization of a typical CNN architecture

2.2. Federated Learning

Federated Learning (FL) is a ML technique [40] that allows a model to be trained on decentralized data, where the data is distributed across multiple participants (e.g., mobile devices, laptops) instead of being centralized on a single server. It makes it possible to train the model without having to centralize the data. This can be advantageous when sharing the data is prohibited due to security or privacy issues. In FL, it is taken for granted that any client can access their own dataset. This may suggest that the data for participants might not be uniformly distributed and do not follow the same statistical characteristics across different subsets or categories [14]. As a result, local models might overfit or deviate from the combined global model, and show poor accuracy. To create a better joint model, local models are averaged to balance out their individual contributions [4]. Learning continues after the model converges. Every round, FL models choose a fresh group of participants. This ensures that the model is adaptable to changes in the data. Fig 2.2 shows a visual representation of FL.

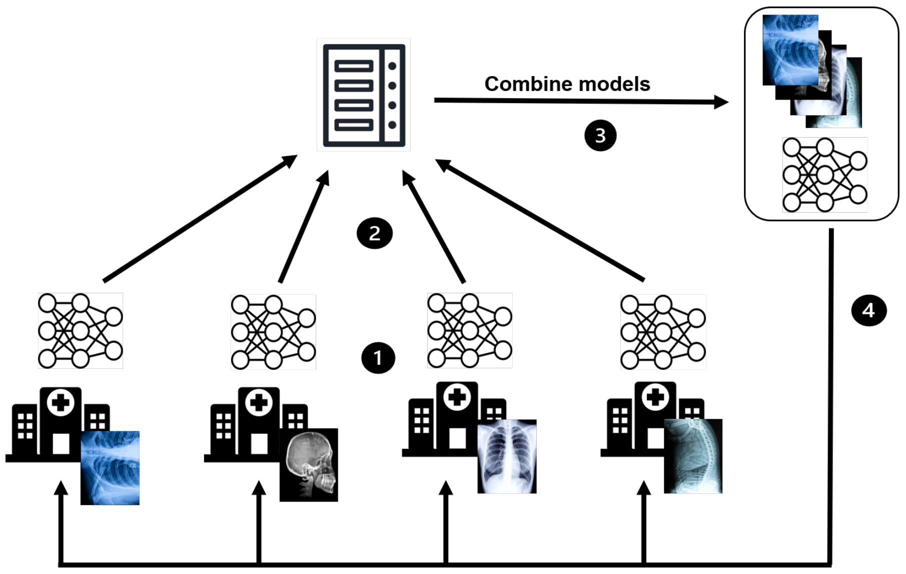


Figure 2.2: Schematic representation of a typical FL setup. 1. Each local user trains a local model. 2. The local models are sent to the central server. 3. The local models are aggregated to create one global model containing all the information from the local models combined. 4. The resulting global model is returned to each local user

The most common implementation of FL is FedAvg [40]. Initially in FedAvg, a global model is initialized on a central server. Participating participants then independently train the model on their local data. This is done in order to prevent privacy intrusion. The central server receives local model updates from the participants and aggregates them using straightforward averaging to produce the updated global model. This procedure repeats itself several times. Algorithm 1 shows the algorithm of FedAvg.

Algorithm 1 Federated Averaging (FedAvg) Algorithm

```
1:  Initialize a global model $\theta^0$ on the central server
2:  Set hyperparameters: learning rate $\eta$, number of communication rounds $R$
3:  for $r = 1, 2, \ldots, T$ do
4:      for each client $k$ in the federated network do
5:          Download global model $\theta^{r-1}$ from the central server
6:          Train a local model $L_k^r$ on client $C$'s local data using $\theta_k^{r-1}$
7:          Compute model update $\Delta L_k^r = L_k^r - \theta^{r-1}$
8:          Upload $\Delta L_k^r$ to the central server
9:      end for
10:     Compute the new global model: $\theta^{r+1} = \theta^r + \frac{\eta}{n} \sum_{k=1}^{C} (L_k^{r+1} - \theta^r)$
11:     Send $\theta^{r+1}$ to all participants
12: end for
```

An instance of the use of FL can be seen in GBoard by Google [41]. Google's virtual keyboard software, Gboard, uses FL to improve its word prediction and autocorrect capabilities whilst putting a lot of importance on user privacy. This method avoids the need to send raw user data to a central server by having each user's device locally build a customized ML model based on their typing history and behavior [41]. On participants' devices, model updates are periodically generated and contain details on the performance of the model without revealing user input. After that, Google's servers combine these changes to enhance the global model that powers Gboard's autocorrect and prediction features.

2.2.1. Horizontal vs Vertical FL

FL can be divided into horizontal FL and vertical FL depending on how data features and samples are distributed among participants. When working with numerous participants holding distinct data instances from the same feature space, horizontal FL is used [57]. This includes people who share the same kinds of data. For instance hospital records of all patients who all have the same data attributes (like age, weight, height, etc.). Vertical FL [69], concentrates on participants that have unique subsets of characteristics from the same collection of instances. This makes it appropriate for situations where customers have complementary data. An example would be merging patient demographics with test results from a medical examination where some patients have data on their weight whereas others have data on their height. Vertical FL collaborates by sharing specific features to improve model performance and horizontal FL commonly shares model updates to train a global model while maintaining data privacy. Fig 2.3 visually shows the difference between horizontal and vertical FL. In this thesis, the focus is on horizontal FL.

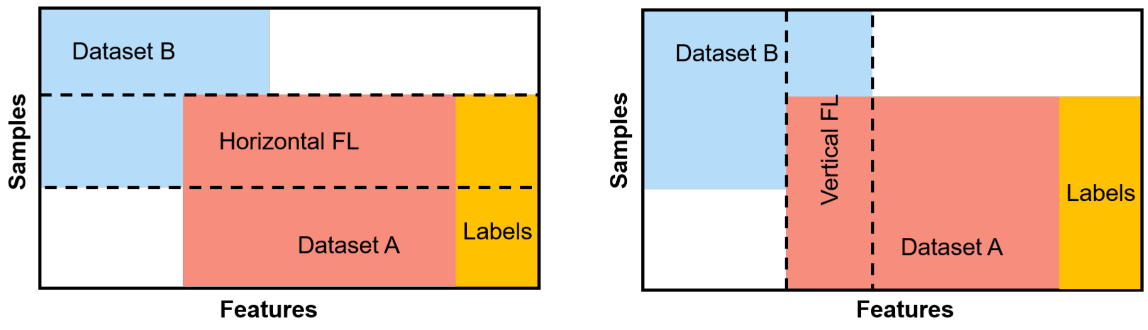


Figure 2.3: Horizontal vs Vertical FL

2.2.2. IID and Non-IID distribution

As each user has its own dataset, the distribution of this data is important as it might fundamentally influence the resulting model. In an Independent and Identically Distributed (IID) data distribution scenario, it is assumed that data samples across different participants or participants are independent of each other. This means that the presence or absence of a particular data point in one participant's dataset does not rely on the presence or absence of that same data point in another participant's dataset [15]. The samples of data are also thought to be identically distributed, which means that the statistical properties of the data hold true for each participant. Every participant's dataset is essentially treated as a sample taken at random from the same underlying data distribution. For instance, let us take the case of participants sharing handwritten digit data. An IID setup assumes that the distribution of digits is homogeneous across all participants and that each participant's dataset comprises a random and representative mix of all possible digits.

On the other hand, a Non-Independent and Non-Identically Distributed (non-IID) data distribution suggests that participant-level data samples are not always independent of one another. This suggests that the independence principle is violated because there can be correlations or dependencies between data samples in various participants' datasets [15]. Additionally, the data distributions across participants can significantly vary. Each participant's dataset may exhibit a particular statistical distribution. This implies that the characteristics of the data, such as the types of data or their frequency, can differ widely between participants [15]. For instance, some participants might have specialized datasets, with one primarily containing even digits and another focusing on odd digits. The non-identically distributed nature of the data introduces complexity to FL. Models must adapt to diverse data characteristics, making it more challenging. Figure 2.4 gives a visual display of what IID and non-IID distributions would look like. The participants with the IID distribution have a similar data distribution if we look at the data points they have in their dataset. On the other hand, The participants with the non-IID distribution do not share the same or a similar distribution but instead have completely different data points. In practice, data in FL settings can often be non-IID due to variations in data sources, participant characteristics, and data collection methods [40].

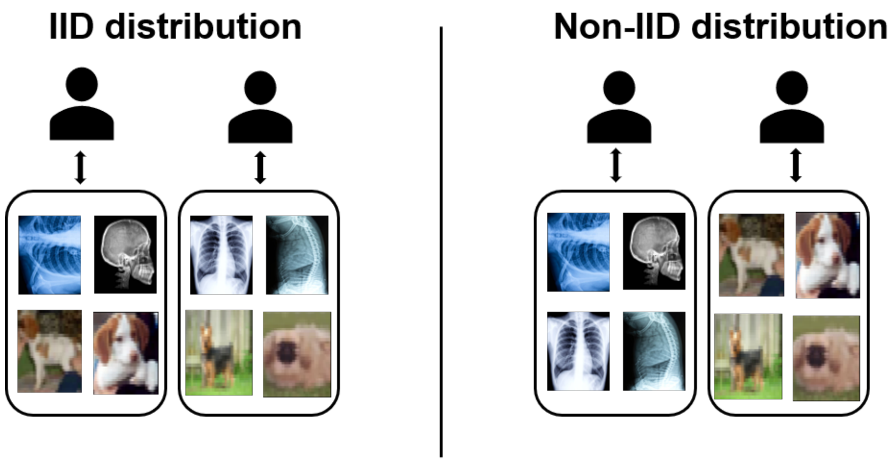


Figure 2.4: IID vs non-IID distribution. (Left) In an IID setting, participants show comparable class distributions. (Right) In a non-IID setting, participants may diverge significantly in their class distributions

2.3. Backdoor Attacks

The emergence of FL did bring its solution but its decentralized design also brought new threats and challenges. One of the challenges faced is the threat of *backdoor attacks*. In a backdoor attack, a malicious participant (or participants) aims to compromise the integrity of the global model by injecting a ”backdoor” pattern into their local training data. These poisoned updates seem harmless to the server, but when the global model includes them, they can cause harmful behavior during testing. This creates a serious security problem, as it enables the attacker to alter or influence the model’s outputs for specific inputs. The attacker wants to maintain high accuracy for the primary objective while simultaneously achieving good backdoor accuracy. As a result, a successful backdoor attack that does not appear suspicious is launched. Mathematically, backdoor attacks can be described as follows: Let there be two datasets D_c and D_p that contain the clean images and images to be poisoned respectively. For each image sample x, $x \in D$ and their respective label y. We denote the poisoned image label as y^*. Then let there be a trigger generation function $\mathcal{T}$ where $\mathcal{T} : \mathcal{X} \leftarrow \mathcal{X}$. $\mathcal{L}$ denotes the loss function.

$$\theta^* = \underset{\theta}{min} \sum_{x_i \in D_c} \mathcal{L}(x_i, y_i) + \sum_{x_j \in D_p} \mathcal{L}(x_j + \mathcal{T}(t, \Delta), y_j^*) \tag{2.3}$$

Having established the fundamental concepts underlying backdoor attacks, these attacks can be categorized into two distinct groups: data poisoning attacks and model poisoning attacks.

2.3.1. Data Poisoning

A data poisoning attack aims to harm the model’s performance during training and testing by inserting deceptive triggers or other manipulations into the training dataset [44]. The model, however, remains unaltered by the attacker. The injected data may be slightly manipulated to cause the model to generate inaccurate predictions or to behave according to the attackers’ wants. Compared to a model poisoning assault, this form of attack necessitates fewer assumptions because the attacker typically just needs access to the data, which is already given in FL [14].

2.3.2. Model Poisoning

In a model poisoning attack, a malicious participant also aims to manipulate the global model's performance by injecting poisoned data during training [44] just like a data poisoning attack. However, unlike data poisoning attacks where hidden patterns are inserted, model poisoning attacks involve stronger adversarial assumptions where the attacker possesses the capability to manipulate the weights of the model they intend to submit for aggregation. The attacker can also modify the hyperparameters that are defined during the training process. In return, the attacker has major influence over the model that will be submitted. By being able to optimize the model for its own objectives, the attacker has a lot of control over the shape and magnitude of the submitted model. In this thesis, we focus on a data poisoning attack. We do not allow the attacker to alter the hyperparameters or the changes of the updates submitted by the attacker.

2.3.3. Measuring success

In this section, we explore the essential metrics that quantify the effectiveness of a backdoor attack. They measure the effectiveness of backdoor attacks in a wide variety of fields. This is important as a backdoor attack has many dimensions that determine its success.

Backdoor Task Accuracy

The *Backdoor Task Accuracy* (BA) assesses how well the model picks up on a hidden pattern or maliciously inserted trigger in the training set. The BA calculates the proportion of correctly categorized cases to all instances in this task, acting as a crucial metric for determining the effectiveness of the attack [71]. A high BA signifies that the model has been manipulated to behave as desired when encountering the trigger. As an example, let there be a dataset that only contained poisoned data samples. A BA of 75% indicates that the model successfully managed to classify 75% of all these samples to the right class. In this case, the right class is the desired label of the backdoor task. Equation 2.4 gives a mathematical expression of the BA:

$$BA = \sum_{\tilde{x}_i \in D^p} \frac{f_\theta(\tilde{x}_i) = y_i^*}{|D^p|} \tag{2.4}$$

Where f_θ is the global model and f is its evaluation function. D^p is the poisoned dataset and $\tilde{x}$ is a poisoned data sample where $\tilde{x} \in D^p$. y^* is the desired backdoor label of the backdoor task.

Main Task Accuracy

The *Main Task Accuracy* (MA) is the accuracy of the primary or legitimate task that the FL model is intended to perform. In the context of backdoor attacks, the main task accuracy in FL analyzes how well the ML model performs on its primary or intended job. This is supposed to be unaffected by the backdoor attack. It assesses how well the model categorizes or forecasts data in the absence of the attacker-introduced malicious pattern or hidden trigger. The model's capacity to maintain its original functionality and generalization despite the inclusion of the backdoor is measured by accuracy in percentages [4]. A high Main Task Accuracy indicates that the model can still perform well on legitimate tasks, even if the model might have been successfully poisoned by the adversary. Conversely, a significant drop in Main Task Accuracy would signal that the backdoor attack has compromised the model. This would give it away and make the attack unsuccessful.

$$MA = \sum_{x_i \in D^c} \frac{f_\theta(x_i) = y_i}{|D^c|} \tag{2.5}$$

Where f_θ is the global model and f is its evaluation function. D^c is the benign dataset and x is a clean data sample where $x \in D^c$. y^* is the correct label of the benign image sample.

Distance of the update

The *distance* of an incoming local model update refers to a measure of dissimilarity or similarity between the local model update sent by a participating client. For instance, let us assume there is an attacker that inserts malicious samples in its dataset as part of a backdoor task. As a result, it can dramatically change the update that the attacker will send in comparison to the incoming updates from benign participants. The update will look very anomalous and defenders will have little trouble in figuring out that the update is malicious. There are different metrics to calculate the distance of incoming updates. In the section below, some commonly used distance metrics are discussed that will also be incorporated into this thesis. The reason for this incorporation is that they are some of the most commonly used metrics in FL clustering defenses [9] [18] [51].

First, there is the cosine distance [53] [9]. The cosine distance is a measure of similarity between two vectors in a multi-dimensional space. It calculates the cosine of the angle between these vectors, and the result ranges from -1 (perfectly dissimilar) to 1 (perfectly similar). The usefulness of the cosine distance is when a malicious update is submitted. Often, it displays a larger distance compared to the rest of the benign updates due to variations in training data. The cosine distance highlights this anomaly when it displays a higher cosine distance than the rest of the submitted updates [53], making it easier to pick the malicious update out of the benign updates. Assume x and y are two individual updates with dimensionality n. The cosine distance is then defined as:

$$\text{Cosine Distance} = 1 - \frac{\sum_{i=1}^{n}(x_i \cdot y_i)}{\sqrt{\sum_{i=1}^{n}(x_i^2)} \cdot \sqrt{\sum_{i=1}^{n}(y_i^2)}} \tag{2.6}$$

The second distance metric is the Euclidean distance. The Euclidean distance is used to measure the straight-line distance between two points in multi-dimensional space. It calculates the length of the shortest path connecting these points. The Euclidean distance is especially useful when the attacker has scaled its update in order to create a stronger attack. As a result of this scaling, the Euclidean distance is significantly larger compared to a non-scaled update, making it easier to reject malicious updates. It is formally formulated as:

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.7}$$

Durability
The *durability* of a backdoor attack refers to the ability of the malicious backdoor or trigger to remain effective and undetected over an extended period of time or across various scenarios. It is an important metric in FL security, indicating the duration during which a maliciously injected pattern or trigger remains effective and undetected within a model [80]. An attack gets stronger the longer it lasts because the FL system finds it difficult to remove the malicious updates through aggregation. To give an example, imagine a traffic sign classification model that is being trained using FL. Now assume a successful backdoor is inserted and as a result gives false results. When the durability of this backdoor is limited, the impact is contained. As a result, the harm done remains relatively minor. However, imagine a scenario where the backdoor persists in the model for an extended duration. In such cases, the backdoor's persistence in the model gives it more potential for harm over time. Because of this, it is essential to consider an attack's ability to persist within the system in the context of security and protection measures because it may increase its potential for harm.

To evaluate the durability of backdoors inserted by different attacks the *lifespan* metric was introduced by [80]. It tracks the maximum amount of rounds the backdoor lasts until the BA goes below a certain threshold. Formally, it can be defined as let t be the epoch index, enumerated starting from the first epoch where the attacker is not present, and let be some threshold accuracy. Then the lifespan l is the index of the first epoch where the accuracy of the model θ on the poisoned dataset D drops below the threshold accuracy, as determined by some accuracy function α:

$$l = \max\{t|\alpha(\theta_t, D)\} > \kappa \tag{2.8}$$

Natural stealthiness
The *natural stealthiness* of a backdoor refers to the inherent ability of a backdoor or trigger to remain inconspicuous or undetectable during the testing phase of the model [44]. It quantifies how effectively the backdoor remains hidden when deployed, with higher values indicating a greater level of natural stealthiness. The natural stealthiness assesses how invisible the backdoor trigger is in its sample images. When a trigger is extremely evident in poisoned samples, detecting the presence of a backdoor

becomes straightforward as it is easily caught by the eye. However, when the trigger is discreetly embedded, its invisibility complicates the distinction between unintentional model misclassifications and intentional incorrect decisions resulting from the backdoor during testing. This will help the impact of backdoor attacks as they are harder to find during the testing phase, enabling longer-lasting and more impactful attacks. Moreover, it dissolves trust in the FL system, as participants may perceive it as unreliable.

To assess the visibility of differences between two images, it is essential to employ a suitable metric. Currently, there is no standard metric used for the visibility of triggers in FL. Therefore, in this study, we propose to utilize the Peak Signal-to-Noise Ratio (PSNR), Structural similarity index (SSIM) and the Learned Perceptual Image Patch Similarity (LPIPS). These metrics are a robust and widely recognized-[27] measure for quantifying the visibility of differences between two images. By leveraging these well-established metrics, we aim to provide a comprehensive evaluation of the visibility of triggers in FL scenarios.

PSNR provides a quantitative measure of the similarity between two images by evaluating the ratio of the peak signal power to the mean squared error between the images. The higher the PSNR value, the more similar the images are considered. PSNR possesses several features that make it effective for measuring visibility differences. First, it accounts for both global and local variations in image content. Thus, enabling a comprehensive assessment of the dissimilarities. Second, it operates in the domain of pixel intensities, making it compatible with various image formats and applicable to both grayscale and color images. Furthermore, PSNR is widely accepted and widely used, making it easier to compare and interpret results across different studies. Mathematically, the PSNR is calculated by:

$$PSNR = 20 * \log_{10} \frac{MAX(D)}{\sqrt{MSE_{x,\tilde{x}}}} \tag{2.9}$$

Where $MAX(D)$ is the maximum pixel value of dataset D and $MSE_{x,\tilde{x}}$ is the mean squared loss between the two image samples x and $\tilde{x}$.

Secondly, the SSIM metric [50] takes into account luminance, contrast, and structure, mimicking aspects of human visual perception. It measures the similarity in local patterns of pixel intensities and returns a value between -1 and 1, with 1 indicating perfect similarity. Unlike PSNR, which primarily focuses on pixel-wise differences, SSIM evaluates image quality by considering how well the structure and texture of the images match. For that reason, it is known as a more sophisticated metric when comparing images. Given two input samples x and y, μ_x and μ_y are its means respectively, σ_x and σ_y are its standard deviations, σ_{xy} is the covariance between x and y and C_1 and C_2 are constants added for numerical stability and to prevent division by zero. The formula for the SSIM can be given as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.10}$$

Given two input samples x and y, μ_x and μ_y are its means respectively, σ_x and σ_y are its standard deviations, σ_{xy} is the covariance between x and y and C_1 and C_2 are constants added for numerical stability and to prevent division by zero.

The LPIPS [78] metric is a perceptual similarity metric used to assess the dissimilarity between two images from the perspective of human visual perception. Unlike SSIM and PSNR, which focus on pixel-wise differences, LPIPS leverages deep NNs that are typically pre-trained on large-scale image datasets to extract high-level visual features from images. The perceptual separation between these features is then calculated, taking structure, texture, and color into account. With this method, LPIPS may catch changes between images that are more subtle and complex making it more similar to human perception. As a result, LPIPS frequently offers more precise evaluations of picture quality and similarity, especially when assessing photos that might have been altered.

Robustness
We define the robustness of a backdoor attack in FL as its ability to withstand a multitude of defensive measures. Due to the evolving landscape of FL, the emergence of backdoor attacks has sparked a parallel development in defensive strategies and countermeasures [43] [51] [52]. These defenses aim to secure the integrity and privacy of FL systems against malicious attackers. This highlights the importance of evaluating not only the efficacy of a backdoor attack but also its resilience in the face of these countermeasures. In essence, evaluating a backdoor attack's effectiveness involves more than just determining if it was initially successful in tricking the FL process. It also calls for experiments on the attack's resilience against a multitude of defenses that employ a wide range of possible techniques. In section 5.5, the defenses employed in this thesis are discussed.

3 Literature review

This chapter delves into the existing literature that forms the foundation of this research. It gives an overview of the current state-of-the-art and within the context, we identify a critical research gap that serves as the focal point of our study.

3.1. Federated Learning

The most common FL system in use is FedAvg. FedAvg [40] starts by initializing a global model and the weights of the model are broadcast to all of the participants. Each client C then trains a local model L^{t+1} using SGD on its own data whilst using the weights of the global model θ^t as the starting point. Once the local model is trained, the client sends the updated local model back to the server. The server then aggregates the weights of all of the local models to create the updated global model. This process is repeated for a specified to an infinite number of rounds. Fig 3.1 shows the mathematical expression for the newly computed global model in FedAvg.

$$\theta^{r+1} = \theta^r + \frac{\eta}{n}\sum_{k=1}^{C}(L_k^{r+1} - \theta^r) \tag{3.1}$$

Whilst this method is very easy and intuitive, new works have been published that tackle a number of challenges commonly seen in FL. These include Aggregation optimization, addressing heterogeneousity, and ensuring fairness. The section below covers what they each mean and what recent advancements have brought forward in each of the fields.

Aggregation optimization

Aggregation optimization refers to the process of improving the way local model updates from participating participants are combined or aggregated at the central server. The intention is to improve the FL process's general effectiveness and efficacy. These aggregate optimization strategies seek to accomplish a variety of goals, including lowering communication costs, protecting privacy, and improving model precision [44]. One such optimizer is FedOPT. FedOPT as proposed by [49], enables a federated setting to generalize the server optimization process to already existing optimizers such as Adagrad, Adam, and Yogi. They showed that the use of adaptive optimizers can significantly improve the performance of FL. Next, a limitation of FedAvg is its coordinate-wise weight averaging. It can negatively impact the averaged model's performance and increase communication overhead. This challenge arises from the permutation invariance of neural network parameters, where various parameter orderings can exist for the same neural network. As a response, PFNM [76] was developed. A Bayesian non-parametric framework for FL with neural networks. It tackles this issue by aligning the neurons of client neural networks before performing the averaging process. [56] offer a layer-wise model fusion approach for neural networks that, prior to averaging the parameters associated with the associated neurons, (soft-) aligns the neurons across the models using optimal transport.

Heterogeneous
Most FL systems contain some level of heterogeneity. This results from differences in data, devices, and data types between entities. It is known as a difficult problem and the development of a unified global model that works on all data and devices may be stopped by these discrepancies. For instance, consider an IoT FL scenario where various smart devices. These can include smartphones, wearables, and home appliances. They all participate in training a global model for a smart home system. These devices exhibit significant heterogeneity from their inherent differences. As an example, one device has significantly more computation power than the other device. As a result, the former can train difficult and complex models whereas the latter is only able to train simple models. Model training and performance are further complicated by variations in device capabilities and class imbalances. As a result, new solutions have been created and are continuously being created [35].

One of these advancements includes FedCurv [55], designed to handle catastrophic forgetting when training on non-iid data and is based on lifetime learning methods. Catastrophic forgetting can occur when the model learns task 1 and then proceeds to learn task 2 however, in doing so it forgets task 1. FedProx tackles the problem of stabilizing training with non-iid data. They do so by generalizing and re-parametrization FedAvg [34]. It is not only the data that can be heterogeneous but the models can be heterogeneous as well. As a solution, HeteroFL was proposed [17]. It addresses heterogeneous participants equipped with highly different computation and communication capabilities. The aggregation is achieved by aggregating parameters on the same location while unlearning the other non-overlapping area.

Fairness
The FL training process as a whole, including client selection, model optimization, incentive distribution, and contribution evaluation, is vulnerable to the fairness problem. Due to the unfairness, both FL participants and FL servers might suffer. participants might be prevented from enrolling in FL training, and servers might be less likely to lure participants who have the potential to be of high grade. Federated Dropout [66] was proposed in order to distribute sub-models with sizes suitable for each client based on their computational resources. It uses dropout procedures and treats neural networks as black-box functions, but it greatly reduces communication and local computation costs.

Adaptive Federated Dropout was introduced as a response to this challenge [42]. An activation score map is kept up to date by Adaptive Federated Dropout to customize sub-models for each client. Federated Dropout and Adaptive Federated Dropout both guarantee that participants with inferior capabilities are included in FL training, but neither of them provides tailored trimmed sub-models for specific participants. In response to this limitation, Ordered Dropout [28] extended Federated Dropout. Ordered Dropout, in contrast to Federated Dropout, dumps just nearby model elements rather than all of them at once. participants with comparable computational capabilities are grouped into clusters, and participants within a cluster drop out at the same pace.

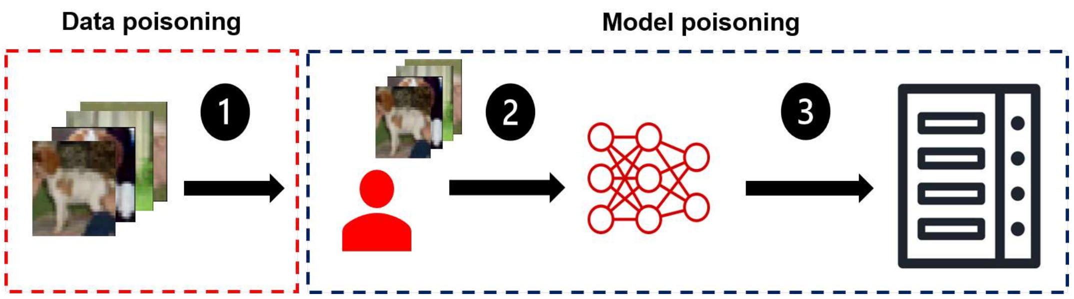


Figure 3.1: Difference between data poisoning attack and model poisoning attack. 1: The data is being poisoned. 2: The user trains a model based on its data. 3: The server sends the trained local model to the central server. As seen, data poisoning attacks are strictly performed in the way the data is being poisoned. Whereas, model poisoning attacks focus on manipulations during training time or after training the local model before sending it for aggregation.

3.2. Backdoor attacks

Recent work on backdoor attacks can be split into two different categories; Data poisoning attacks and model poisoning attacks. Model poisoning primarily focuses on altering the local model itself by changing the parameters and gradients during the model aggregation process. For instance, an attacker can boost the gradients that influence the backdoor task the most in order to retain a stronger backdoor after aggregation. On the other hand, data poisoning techniques target the training data itself. The samples used by the attacker, while appearing legitimate, carry subtle manipulations designed to introduce the backdoor in the global model. Data poisoning emphasizes the influence of model training rather than the model's architecture. A schematic overview of their distinction is shown in figure 3.1.

In light of the distinct characteristics of each attack, we have chosen to categorize them into various groupings and subgroups. This approach aids in gaining insights into the prevailing research directions and the specific areas of emphasis within the field, allowing for a better understanding of prior research. An overview of this is found in figure 3.2

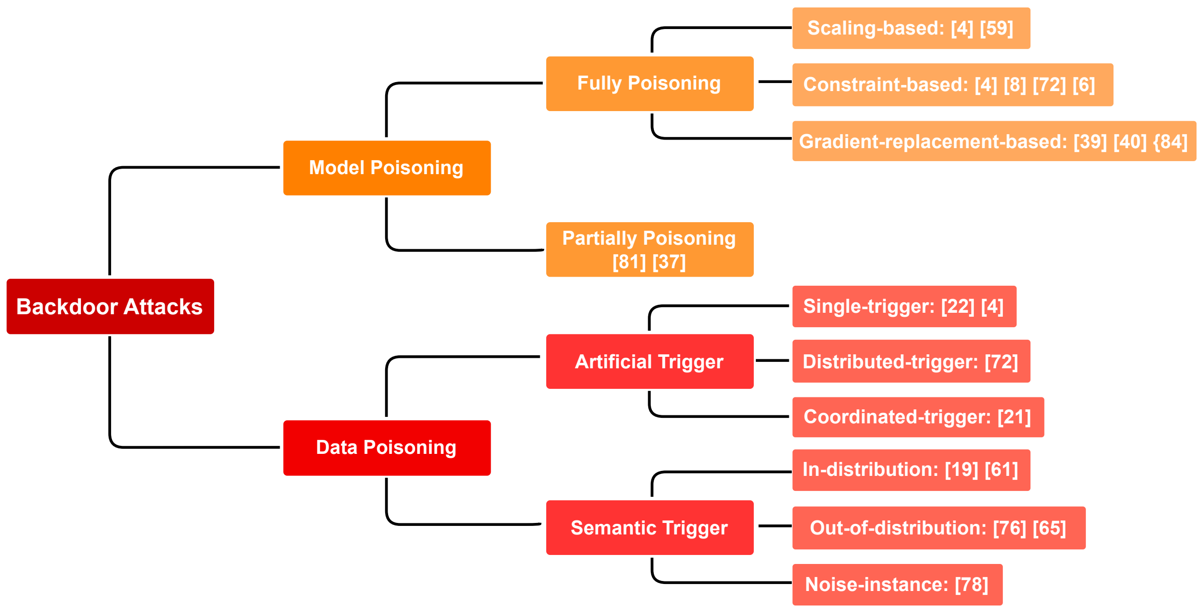

Backdoor Attacks
Model Poisoning
Fully Poisoning
Scaling-based: [4] [59]
Constraint-based: [4] [8] [72] [6]
Gradient-replacement-based: [39] [40] {84]
Partially Poisoning [81] [37]
Data Poisoning
Artificial Trigger
Single-trigger: [22] [4]
Distributed-trigger: [72]
Coordinated-trigger: [21]
Semantic Trigger
In-distribution: [19] [61]
Out-of-distribution: [76] [65]
Noise-instance: [78]

Figure 3.2: Overview and categorization of the latest backdoor attacks

3.2.1. Data poisoning

In the realm of data poisoning attacks, the attacks can be split up into two different categories; Semantic backdoor attacks and artificial backdoor attacks. Fig 3.3 visualizes the difference between artificial and semantic backdoors.

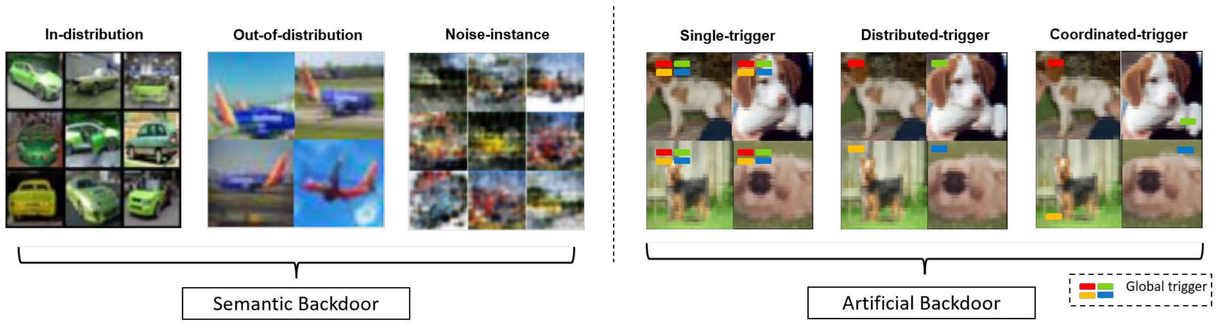


Figure 3.3: Visualization of semantic and artificial backdoors. From left to right: (1) Green cars used in the original dataset to use them as backdoor samples. (2) Planes of Southwest Airlines were used as backdoor samples but were not originally in the dataset. (3) Noise-created samples. (4) Each user has a consistent single trigger. (5) Global trigger split among many participants. (6) Strategic distribution and optimization of a global trigger among participants to enhance backdoor attack performance

Semantic backdoor attacks

In semantic backdoor attacks, malevolent actors manipulate the labels of genuine data obtained from compromised participants. They employ diverse methods to choose particular legitimate samples for manipulation. Often they target samples closely aligned with the benign data distribution to poison the model. Certain strategies, as discussed in [19] and [61], concentrate on modifying samples that closely align with the overall data distribution. Since these samples might contain information from other participants or testing sets managed by the central server, these techniques are referred to as ”in-distribution backdoor attacks.” For instance, all photographs with the label ”1” are carefully reclassified as ”0” [19]. Another example used images with the label ”dogs” and rebranded them as ”cats” [61]. Other methods as detailed in [4], set their sights on samples characterized by certain qualities or characteristics. Such can be the presence of unusual objects in a scene or unusual car colors like ”green.” These techniques frequently include a chosen target word in word prediction challenges. These methods do have a disadvantage, in that updates from good participants can lessen the effectiveness of the backdoor.

[64] and [75] used a unique approach by choosing samples that significantly deviate from the distribution of the global data in recognition of these limitations. It is unlikely that these selected samples will be found in the validation or training sets of benign participants. Their attacks are successful because they deliberately target samples that are frequently found at the edges of the data distribution of benign participants. This ensures that the impact of the backdoor is robust and unaffected by dilution. For example, in [64], they introduced the concept of an ”edge-case” backdoor attack. Edge-cases are image samples that are unlikely to be in the datasets of benign participants, yet do belong to the class they belong to. For instance, benign participants can have images class airplanes but, it is highly unlikely that these images contain airplanes of Southwest Airlines. These ”edge-case” image samples help disguise the backdoor. In [75], they innovatively introduced the ”ultimate rare word embedding” to activate backdoors in the domain of NLP. Again, this is a similar idea of using data samples that belong to a class in the FL system but are highly unlikely to be part of the dataset of benign participants.

It is important to note that the methods described above frequently require some knowledge about the target model, including details about the global data distribution. However, in real-world situations, such information might not always be applicable. For that reason, [77] took a different approach. They suggested a different approach that entails instructing a Generative Adversarial Network throughout the local training procedure. The network is then used by the shared model to create samples similar to other participants’ training samples, which are later used to carry out the backdoor attack. This tactic performs particularly well when the attacker isn’t fully aware of how the data is divided up among benign participants.

Artificial backdoor attacks

This alternate strategy gives greater flexibility than semantic backdoor assaults, where the targeted samples must have particular characteristics and belong to particular classes. The adversary must artificially poison healthy samples before changing their labels. In essence, there is no inherent ”key” in the samples; rather, the backdoor is opened by the attacker inserting a pattern, like a cross into the corner of an image. The basic idea behind this attack approach is to make the model behave abnormally when there is a trigger while still functioning normally in the absence of a trigger.

Normally, backdoor attacks against FL have primarily relied on a single trigger. In this case, all compromised participants inject the same trigger into their local training datasets [22] [4]. The trigger is frequently predefined and consists of patterns like squares or crosses that are put at wasteful pixels in photos. Malicious participants then use the injected trigger to activate the aggregated model during the testing phase. While the effectiveness of this backdoor insertion has been demonstrated [22], Due to the fact that these approaches evenly embed the same trigger across all adversarial participants, they have not fully used the decentralized FL environment. In response to these drawbacks, DBA [71] was developed as a new strategy. DBA breaks down the objective trigger into a number of local triggers. Each of these is related to a specific malicious local agent. After local training, each adversarial agent deploys the local trigger to manipulate the training data and then sends the backdoored update to the server. Instead of employing each of these local triggers separately like in the previous way, DBA combines them to create a global trigger that opens the backdoor. The result was higher attack success rates and more stealth in comparison to a centralized trigger [71].

In previous techniques, the chosen trigger employed by adversaries was often generated independently of the learning model and learning process. It involved elements like logos, stickers, or pixel perturbations. Consequently, these backdoor attacks did not fully utilize the collaborative efforts of multiple malicious participants during the training phase [21]. To address this limitation, introduced the concept of a coordinated-trigger backdoor attack. In this method, the adversary injects the backdoor by using a model-dependent trigger. The model-dependent trigger is determined by a training procedure that optimizes the form, size, and placement of the trigger. It represents the ideal trigger configuration for each malicious participant. The local training dataset is poisoned depending on these local triggers after they have been generated for each malicious client. A global trigger is created during the testing phase by fusing the local triggers as in DBA. As a result, it has been demonstrated that the model-dependent trigger is more effective than the DBA trigger used in earlier publications [21].

3.2.2. Model poisoning

This section provides an overview of the latest developments in model poisoning techniques, shedding light on the evolving landscape in this domain. These advancements can be categorized into several key sections, namely: Scaling-based, Constraint-based, Gradient Replacement-based, and Partial Model Poisoning Attacks. Within each subsection, we delve into the definitions and explore their respective impacts on the nature of these attacks.

Scaling based

Scaling-based attacks are defined by upscaling the model parameters to boost the backdoor performance. Commonly, the global model is trained by simply averaging the incoming local models. A simple method for increasing the backdoor effect is boosting the influence of adversarial participants' updates so that they override those from good participants. The attacker will attempt to replace the new global model with a poisoned model by carefully changing the scaling factor. This strategy was first proposed as the model replacement method in [4]. This strategy requires an evaluation of global parameters and is particularly effective when the global model is nearing convergence [59]. It's worth noting that in the presence of defenses using clipping and restricting methods, simple scaling may become ineffective because of it [44].

Constraint based

To enhance the stealthiness of their attacks, malicious actors may take new strategies aimed at limiting the magnitude and visibility of local updates from malicious participants. This can help them avoid the triggering of anomaly detection mechanisms on the server. That is what is meant by constraint-based attacks. For instance, a defense scheme has been employed that looks into the magnitude of the incoming updates for anomalies. By constraining this magnitude, the attacker reduces its abnormality and is thus more capable of avoiding defenses. In previous works like [4] and [8], an approach was introduced wherein the loss function is adapted by integrating terms associated with anomaly detection. For instance, attackers may use the magnitude of the Euclidian distance of their updates in order to reduce their anomaly. These loss functions are constructed based on the underlying anomaly detection techniques, incorporating metrics such as the p-norm distance between weight matrices and validation accuracy. Additionally, the use of projected gradient descent (PGD) attacks was introduced by [71]. It has emerged as a robust strategy capable of withstanding various defense mechanisms. In PGD attacks, attackers constrain their model's parameters within a radius centered around the previous global model iteration. This ensures that in each FL round, the attacker's model remains relatively close to the global model. Similarly, a technique proposed by [6] defines a permissible range for parameter adjustments by attackers. It aims to prevent undue attention even in IID settings. To further reduce the anomaly of these malicious updates, an additional clipping step is introduced to maximize their effectiveness in avoiding detection.

Gradient replacement based
In the earlier discussed model poisoning attack techniques, participating entities have access to the labels of their respective training data thus always working in a horizontal FL setting. Yet, these methods have not been thoroughly examined within the context of Vertical FL. Addressing this gap, [38] [39] introduced the Gradient-replacement backdoor attack. It is specifically tailored for Vertical FL, even when the adversary has access to only a single benign sample belonging to the targeted class. The intermediate gradients of clean samples from the targeted class are captured using this technique and they are then used to replace the gradients of poisoned samples. The model is then updated using the modified gradients. Furthermore, recent research by [83] has demonstrated the improved adaptability of the backdoor attack. Even in scenarios with communication protected by Homomorphic Encryption. The backdoor task is built on the label inference attack, where the malicious party learns the corresponding true labels of each sample. The malicious party then replaces the true label with a target label by carefully designing the encrypted communicated messages which in turn is used to perform the backdoor attack.

Partial model poisoning
Partial model poisoning techniques do not fully utilize the full model space but instead, only influence a subsection of the model space. In the concept of ”Neurotoxin,” [80] the adversary adopts a strategy where they target coordinates that benign participants are unlikely to modify. The aim is to embed the backdoored model and extend the duration of the backdoor's impact. In their framework, the attacker doesn't directly update the poisoned model using gradients computed on poisoning data. Instead, they project the gradient onto coordinate-wise constraints, specifically focusing on the bottom-k% coordinates derived from the observed benign gradient. The objective behind partially poisoning attacks is to achieve two key goals. Firstly, they aim to prevent catastrophic forgetting of the adversarial task. It ensures that the model retains its original capabilities. Secondly, these attacks seek to prolong the persistence and effectiveness of the backdoor, allowing it to maintain its impact over an extended period. A novel two-phase backdoor attack strategy was proposed by [36]. It incorporates an initial convergence-expediting phase preceding the poison phase. This new approach puts the spotlight on the early stages of the attack where the backdoor effect is less susceptible to dilution from regular model updates. As a result, the injected backdoor attains heightened effectiveness by leveraging the initial phase to its advantage. This two-phase strategy offers improved efficiency and potency in achieving the attacker's objectives.

3.3. Backdoor Defenses

Many works regarding defenses for FL have been conducted in recent years with many different proposed solutions each having their own respective pros and cons. We sum up some of the recent solutions proposed in this section by each category it works in. Figure 3.4 shows an overview of some defense techniques published thus far. We decided to split them up into the following subsections: pre-aggregation defenses, in-aggregation defenses and post-aggregation defenses.

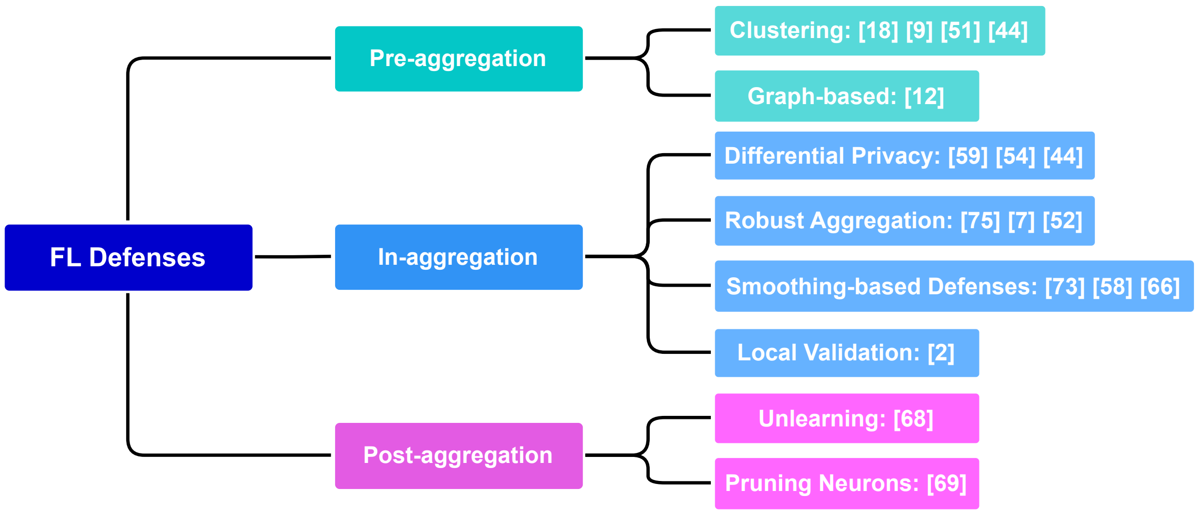

FL Defenses
Pre-aggregation
In-aggregation
Post-aggregation
Clustering: [18] [9] [51] [44]
Graph-based: [12]
Differential Privacy: [59] [54] [44]
Robust Aggregation: [75] [7] [52]
Smoothing-based Defenses: [73] [58] [66]
Local Validation: [2]
Unlearning: [68]
Pruning Neurons: [69]

Figure 3.4: An overview and categorization of some of the latest defenses

3.3.1. Pre-aggregation defenses

In the realm of pre-aggregation defenses, the focus is on identifying and mitigating the potential threat posed by malicious updates before they become part of the aggregation process. This proactive approach is primarily initiated by the server upon receiving incoming updates. Our exploration of pre-aggregation defenses is divided into two distinct sections for clarity. The first section encompasses clustering methods, which form the predominant group of pre-aggregation defenses. In the second section, we delve into graph-based defense techniques.

Cluster methods

Clustering defenses refer to techniques and strategies employed to identify and group together similar or related updates received from participating participants. Usually, these changes are connected to the corresponding local models or datasets. To distinguish between legitimate client contributions and those that could be malicious, clustering defenses seek to identify patterns or irregularities in incoming updates [44]. Once these clusters have been identified, it could then isolate the updates that show irregularities and prevent them from being aggregated. Commonly, identifying malicious updates is based on some distance metric. For instance, the Cosine Distance and the Euclidian Distance are two frequently used distance metrics.

Such protection techniques are used by FoolsGold [18]. FoolsGold uses the cosine distance to spot unusual client updates during model aggregation. It computes a weighted average of updates and identifies deviations from expected behavior, marking potentially malicious updates. By downweighting or excluding these anomalies, Fool's Gold mitigates the impact of backdoor attacks, making the security of FL more robust without detailed knowledge of the attack. Multi-Krum [9] is a defense where each round all the Euclidian distances are calculated with respect to all other local models. Based on the resulting distances it creates a subset of local models that are the closest to the majority of other models, thus excluding any outliers.

In [51] the authors proposed Deepsight, a novel model filtering method. DeepSight's approach involves using an ensemble of different techniques in order to identify poisoned updates. It does so by analyzing the data distribution used to train model updates and assessing anomalies in the internal structure and outputs of neural networks. It then combines it with clipping to identify and eliminate incoming poisoning attacks. FLAME [43] employs the HDBSCAN algorithm to find harmful updates. HBDSCAN builds a hierarchy of clusters with varying densities and identifies the most stable clusters at different levels. It then combines it with update clipping and noise addition to get rid of backdoors. However, FLAME calls for more computer power than conventional FL methods and could deteriorate benign accuracy.

It is important to acknowledge that clustering defenses usually perform less when presented in non-IID scenarios. Due to the highly heterogeneous distribution of the data trained on by local participants, the local models will significantly differ from one another [4] [64]. As a result, it becomes hard for clustering techniques to recognize the malicious update out of all updates.

Graph-based methods
Graph-based defensive methods use graph structures to analyze and identify potentially malicious model updates before the aggregation process. In [12], the authors introduce a graph-based approach to anomaly detection. They construct a graph that represents local model updates and employ a maximum clique problem-solving technique to identify benign models. This method is specifically tailored for the IoT (Internet of Things) domain. It employs an innovative approach to anomaly identification by representing network packets as symbols in a language, thus facilitating the application of language analysis methods to effectively detect and flag irregularities.

3.3.2. In-aggragation defenses

In-aggregation defenses are defenses that prevent backdoors during the aggregation process. By making this process more robust, which can be accomplished through a range of techniques, these defenses successfully maintain effective model aggregation while fending off unwanted influences from potential attackers. This section categorizes these defenses into four main areas: robust aggregation, differential privacy, smoothing-based methods, sparsification, and local validation defenses. Each of these areas offers distinct strategies and tools to reinforce the aggregation process.

Robust aggregation
Robust aggregators perform the aggregation in a different way such that it becomes more robust against backdoor attacks. For instance, commonly used aggregators such as FedAvg do not have any defenses in place against backdoor attacks. By changing the way of aggregating, one could mitigate poisoning attacks. These robust aggregators try to prevent backdoors from entering during the aggregation phase. An early solution proposed was the use of the geometric median as a form of aggregation for the global model [74]. Due to its strong robustness to outliers, the geometric median is resistant to the influence of potentially malicious participants who may have manipulated the data or model updates. The use of the geometric median can lessen the effects of outliers and make sure that the final model represents a more accurate consensus of the benign participants' contributions. A different strategy is the usage of the Median and α-trimmed mean, which substituted the median of model updates for the arithmetic mean to boost robustness against attacks [74]. SignSGD [7] puts the focus on the direction rather than the magnitude of gradients. It helps maintain the integrity of the FL process, making it more resilient to backdoor attacks while preserving model performance. This solution was further worked out by [52] with their implementation of a Robust Learning Rate, which adjusts the server's learning rate in accordance with client updates.

Differential privacy
Differential privacy is a framework that protects the privacy of individuals in a dataset by adding noise to the data before it is released or used for analysis. The effectiveness of differential privacy against backdoors has been demonstrated [59]. However, it can impair model performance in the presence of data imbalance [3]. Weak-DP was proposed by [59], which maintains task accuracy while adding enough Gaussian noise and weight clipping to prevent backdoors. This noise disrupts any hidden backdoor signals, making it harder for attackers to manipulate the global model's behavior through model poisoning. Weak-DP also ensures that no single client's contribution overly influences the global model, enhancing model robustness and privacy. Norm clipping alone has proven itself to mitigate poisoning attempts based on recent attacks in [54]. Malicious updates are found to have a higher Euclidian distance. By constraining this distance using a fixed value, the effectiveness of the backdoor attack can be removed. Additionally, FLAME [43] combines the perks of clustering and DP in order to catch malicious models. It is a defense framework that determines how much noise will be needed to ensure that there are no backdoors. FLAME employs a model clustering and weight clipping strategy to reduce the necessary amount of noise. As a result, FLAME may continue to maintain the aggregate model's benign performance while successfully closing down adversarial backdoors. Although differential privacy has potential as a protection strategy, it does have some drawbacks. Notably, it becomes most successful

when a sizable number of participants are involved. This is due to the requirement that the additional noise needs to be of a sufficient scale to mask the characteristics of each client's data, making it more difficult to identify backdoor attempts. However, the substantial noise that was added throughout this process can have an adverse effect on the general performance of good participants.

Smoothing based defenses
Smoothing-based defenses work by smoothing the aggregation results and thereby mitigating the impact of any harmful updates. These defenses help protect against backdoor attacks by reducing the chances of a malicious update significantly affecting the global model. By effectively "smoothing out" the contributions of individual client updates, they make it more challenging for a backdoor to manipulate the model or introduce a harmful bias. One of such defenses is CRFL [72]. Introduced a new defense framework, CRFL is designed to train certifiably robust FL models resilient to backdoor attacks. It leverages techniques like cropping and smoothing of model parameters to control model smoothness and thus generate robustness against backdoor attacks. The incorporation of smoothness and perturbation methods further constrain the Euclidian distance of individual updates, making the defenses more robust against backdoors. FL-WBC [58] focuses on identifying vulnerable parameter spaces within FL and perturbing them during client training. It provides FedAvg with convergence assurances as well as robustness guarantees against backdoor attacks. FLARE [65] assumes that the majority of the participants are benign. It suggests a method for assessing model updates' trustworthiness based on differences in how the penultimate layer represents them. Lower trust scores are assigned to updates outside of the benign cluster, and these values are weighted during model aggregation to properly update the global model. At last, the sparse nature of NN's is proposed as a defense mechanism against backdoor attacks. Sparsefed [46] is a framework that incorporates sparse updates. Sparse updates are a technique where only a subset of the model parameters are updated by each client during training. It selectively updates a small fraction of the highest-magnitude elements. Due to attackers moving in different directions from the benign devices, the coordinates that attackers typically target for model poisoning often remain untouched. The advantage is that it is a relatively easy and cheap method of aggregation. Yet, it has failed to show its effectiveness in non-iid distributions.

Local validation
Local validation-based defenses are characterized by evaluating the credibility and validity of individual client updates before they are included in the global aggregation. BaFFle [2] presents a decentralized feedback-driven approach designed to be resilient against backdoors in FL. This method uses client data to perform an additional validation step on the global model. To check the integrity of the global model, a subset of participants applies a validation function to sensitive data. If they discover any indication of backdoors, they alert the server. The server then decides whether to accept or reject the global model based on the variance in misclassification rates between the local model and the global model. While BaFFle presents a secure aggregation technique, it does come with certain limitations. For instance, it relies on trigger data to activate the backdoor. It also may not be suitable for non-IID data scenarios with a limited number of participants and may not be highly effective against continuous attacks that disrupt FL training processes.

3.3.3. Post-aggregation defenses

Post-aggregation defenses introduce an extra step to cleanse the global model of any poisoning after it has been aggregated. One of the few post-aggregation defense strategies for FL against backdoor attacks was introduced in [68]. Their method focuses on locating and eliminating neurons that show low activation when exposed to benign samples. Without the trigger, these neurons are likely to remain inactive. A distributed pruning technique was suggested to deal with the issue of the server not having access to personal training data. In this approach, the server asks participants to create a local pruning list by recording neuron activations using their local data. Then, a global pruning sequence is created using these lists. Additionally, in order to improve the pruning list, the server can dynamically modify the pruning rate based on how well the current model performs on a validation dataset. The disadvantage of this attack is that is not always able to remove the backdoor as benign and backdoor behaviors utilize the same set of neurons in some cases. For this reason, pruning has the potential to trigger backward and inflict more harm on benign behaviour.

Recent developments in machine learning, particularly the concept of ”unlearning,” have gained prominence in [67]. They explore its application as a defense mechanism against backdoor attacks in FL. To mitigate single-trigger backdoor attacks without drastically degrading overall performance, they introduced federated unlearning. It erases the historical parameter updates from the target client and then recovers the damage through a technique called knowledge distillation. However, before this technique can begin unlearning, it must first identify malicious participants, which is a significant challenge already. This strategy could be employed for backdoor removal upon the identification of a malicious participant, yet its effectiveness is constrained in the absence of such knowledge.

3.4. Research gap

In prior studies, most of the research has mainly focused on situations where backdoored models are trained and tested using the same trigger intensity. This often overlooked the possible effects of changing the trigger's intensity. Our thesis sets out to bridge this gap in understanding by delving into the world of trigger intensity manipulation and its potential impact on backdoor attacks. We are curious about how tweaking the trigger intensity can be utilized by attackers to strengthen their backdoor attacks.

Additionally, we have observed that there is a significant lack of research when it comes to detecting backdoors during the testing phase in FL. So that means that after aggregation, only little defenses exist that could eradicate the backdoor. By playing around with different pixel intensities, we might find methods to create backdoors that are nearly invisible. Once these hidden backdoors are in place, detecting them can be a real challenge because they can easily be mistaken for regular system errors. This could mean that getting rid of these backdoors, once they've infiltrated the system, becomes a much more complex challenge. Our research aims to dive deep into these important questions and shed light on the world of backdoor attacks and the influence of pixel intensities, expanding the knowledge on the topic.

At last, using the trigger intensity can work alongside other model and data poisoning techniques, potentially making backdoor attacks even more effective. This adds an extra dimension to our research, as it suggests that our attack can be used in combination with other attack methods. It can serve as an extra tool in the shed for attackers to create more powerful backdoor attacks.

Each point summed up:

- **Trigger Intensity Exploration:** The research addresses a gap in understanding by investigating the impact of varying trigger intensity in backdoor attacks in FL, breaking away from the conventional focus on consistent trigger intensity during training and testing.
- **Detection Challenges:** The study highlights the limited research on detecting backdoors post-aggregation in FL. Existing defenses are insufficient, and manipulating pixel intensities to create nearly invisible backdoors adds complexity to detection.
- **Stealthy Attack Potential:** By manipulating trigger intensity, the research explores how attackers can enhance the effectiveness of their backdoor attacks. This insight complements existing attack methods, potentially making backdoor attacks even more potent and challenging to detect.

4 Methodology

4.1. Motivation

Common strategies for incorporating backdoor triggers into machine learning models typically involve maintaining a single, unvarying intensity level throughout both training and testing phases. In our research, we specifically define the training phase as the initiation of distributing a global model to individual devices. Each device independently refines the model through localized training, generating gradient updates that are subsequently sent to a central server for aggregation. Following this aggregation, the testing phase begins as the model is deployed and subject to evaluation. Figure 4.2 illustrates this process. This approach presents inherent limitations that our research found. Our findings indicate that when the models confront complex or challenging visual conditions, they often struggle to recognize the presence of the backdoor trigger. As a result, the model's performance tends to degrade significantly.

This challenge becomes more concerning in real-world scenarios, where practical applications of ML models are deployed [44]. In such settings, the visibility and effectiveness of a backdoor trigger can be influenced by a wide range of external factors, including variations in lighting conditions, dust particles, or other environmental elements. These factors can have a substantial impact on how well the trigger is detected and utilized by the model. For instance, consider a surveillance system that employs facial recognition technology to grant access to secure facilities. If the backdoor trigger used for training and testing the model is not robust against variations in lighting conditions, it could lead to a reduced impact from the attacker. An abrupt change in lighting, such as strong sunlight or dimmed indoor lighting, may render the trigger undetectable for the model. This could potentially remove the effectiveness of the backdoor.

In essence, our research underscores the attackers' need for more adaptive and resilient methods of injecting backdoor triggers into ML models. These methods should account for the dynamic and unpredictable nature of real-world conditions, ensuring that the trigger remains effective and reliable even in the face of challenging visual scenarios. To address these challenges, we came up with a method involving pixel intensity that is more robust against these challenges and could help the attacker create a more resilient attack.

Understanding backdoor attacks in the context of FL systems requires taking into account both the subtlety with which these attacks can be carried out and the real-world conditions in which they might arise. We've talked a lot about how important it is to imagine scenarios in which backdoor attacks could be dangerous, but creating invisible backdoors is just as important. As elaborated upon in Section 3, the existing landscape of defense mechanisms predominantly focuses on thwarting backdoor attacks during the model aggregation phase. This means that once a backdoor has been stealthily planted into the global model, its detection becomes much more challenging and often relies on manual human inspection. Consequently, it becomes clear that attackers stand to gain a substantial advantage if they can master the art of concealing their backdoors for as long as possible. Thereby, allowing them to bypass defenses for an extended period.

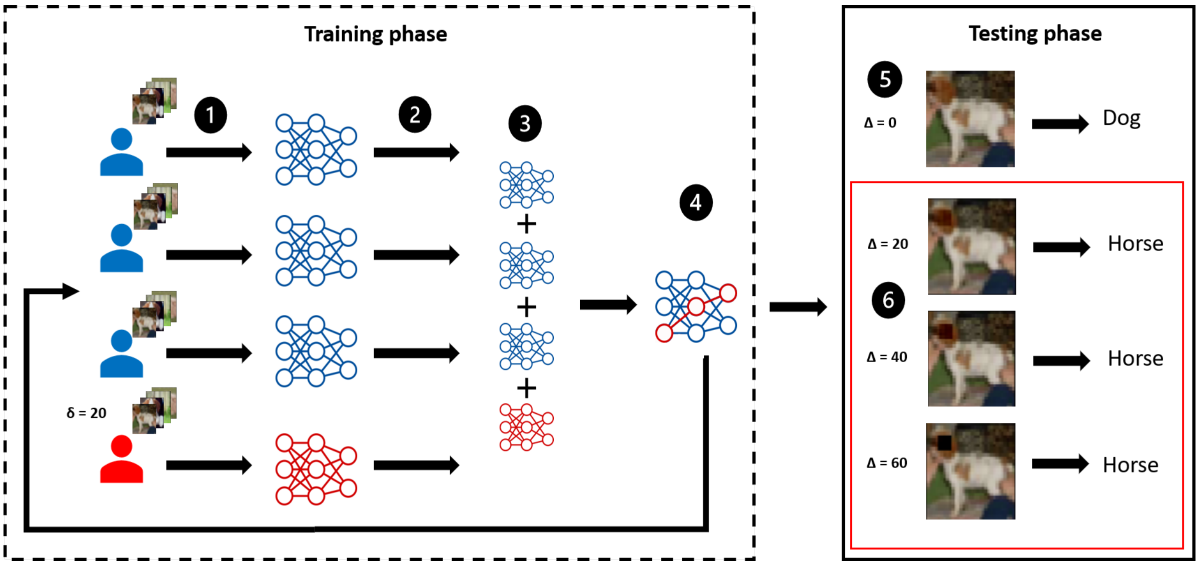


Figure 4.1: Oversight of Backdoor Attacks in FL and using different intensities values for training and testing. **Training Phase**: 1. The participants train on their own dataset and produce a local model. 2. Each agent sends their local model to the server 3. The server aggregates all incoming local models 4. The aggregated model gets sent back to the participants **Testing Phase**: 5. The local benign participants can get correct predictions when they feed benign samples into the final global model 6. The local malicious participants can activate the backdoor by using a stronger trigger and get a target prediction

To address this challenge, we propose a novel approach in which we train the backdoor task using a specific training intensity and subsequently confirm its effectiveness using a higher testing intensity. This approach enhances the generalization capability of the backdoor, making it more robust and resilient in real-life scenarios where trigger visibility can vary. By allowing the attacker to adeptly select the trigger intensity after the aggregation process, our method provides an additional layer of customization and adaptability, enabling the attacker to handcraft the backdoor to specific images or target scenarios.

The ability to exercise manual control over the trigger intensity grants significant advantages to the attacker, both in terms of attack performance and detection evasion after aggregation. By taking this control, the attacker gains the flexibility to optimize the trigger intensity to achieve desired levels of attack performance. Furthermore, the attacker can tailor the trigger visibility to specific levels, making the backdoor less conspicuous and harder to detect during the test phase. This invisibility complicates the distinction between an unintentional misclassification by the model and a deliberate incorrect decision resulting from the backdoor in the test stage. All of this enhances the backdoor's stealthiness. Moreover, our method complements SOTA backdoor attack techniques, further enhancing their efficacy and impact in terms of performance, lifetime, and stealthiness. This newfound power gives attackers the means to devise more potent and sophisticated backdoor attacks, presenting a substantial threat to the integrity and reliability of FL systems. Fig 4.1 gives an overview of how our attack works.

4.2. Our attack

4.2.1. Defining the trigger intensity

To create a formal framework for understanding and manipulating backdoor triggers in the context of FL, we introduce the concept of "trigger intensity." Trigger intensity represents a critical attribute of these triggers, dictating the pixel values used to construct them. In essence, it determines how visible or imperceptible the trigger is within an image.

To illustrate this concept further, consider the analogy of adjusting a dimmer switch on a light fixture. In this analogy, a higher trigger intensity corresponds to a "brighter" trigger, one that is highly visible and conspicuous within an image. On the other hand, a low trigger intensity equates to a "dimmer" trigger, making it nearly indistinguishable from the image's visual content. Trigger intensity essentially controls the extent to which the backdoor trigger is perceptible to both the human eye and ML models. In the context of FL, where the model is trained collaboratively across multiple devices, we introduce two distinct levels of trigger intensity:

Training Intensity (δ): During the training phase of FL, this parameter (δ) represents the intensity level of the trigger applied to the training data. It defines how prominent or subtle the trigger appears in the training images and influences how the model adapts to recognize it.

Testing Intensity (Δ): In contrast, the testing phase employs the testing intensity (Δ) to determine the trigger's visibility during the evaluation or testing stage. This intensity level dictates how detectable the trigger is when the model is subjected to new, unseen data.

The interplay between these two intensity levels is central to the effectiveness and security of backdoor attacks in FL. The choice of δ and Δ can significantly impact the success of the attack, as well as the model's ability to recognize and mitigate it. For a visual representation of how these triggers manifest at different δ values, please refer to Figure 4.2. This figure provides concrete examples that demonstrate the varying degrees of trigger intensity and their influence on trigger visibility within images.

4.2.2. Trigger generation

We propose a backdoor attack in which the training intensity of the local model will be different compared to the testing intensity during the testing phase. For that reason, our definitions use two different delta values, where one is for the training set and one for the testing set. We define a trigger t as follows:

$$t = [t_1, t_2, ..., t_n] \tag{4.1}$$

where t_i represents the pixel values of the trigger, and n is the number of pixels in the input data. Furthermore, for a training sample x let there be a trigger generation function $\mathcal{T}$ where $\mathcal{T} : \mathcal{X} \rightarrow \mathcal{X}$.

$$\mathcal{T}(t, T) = t * T \tag{4.2}$$

Where t is the trigger itself (a square shape for instance) which has all its values set to 1 and T is the trigger intensity given as a parameter. Thus, we can define the trigger generation for the training dataset as:

$$\tilde{x} = x + \mathcal{T}(t, \delta) \tag{4.3}$$

Where $\tilde{x}$ is the poisoned training sample and δ is the training intensity. We define the trigger generation for the testing dataset as follows:

$$\tilde{x} = x + \mathcal{T}(t, \Delta) \tag{4.4}$$

Where Δ is the testing intensity.

4.2.3. Explaining the attack

To make our attack effective, the trigger intensity of the training dataset (δ) must be set to a lower value compared to the testing intensity (Δ) used during the testing phase. The effectiveness of our backdoor attack technique is built on a remarkable phenomenon: only a select few neurons within the model play an important role in triggering the malicious behavior [82]. Thus, by increasingly stimulating these neurons, our backdoor's effectiveness increases simultaneously.

To elaborate on this concept, let's consider a scenario in which we employ two distinct triggers, denoted as (t, δ) and (t, Δ), with the key distinction that δ is less than Δ. This differentiation in trigger intensities is fundamental to our approach. In our strategy, we start by training a neural network using data that has been subtly poisoned with the weaker trigger, (t, δ). This initial step introduces an ineffective backdoor within the model to respond to the presence of the weaker trigger. However, when the model encounters the more potent trigger during the testing phase (t, Δ), something interesting

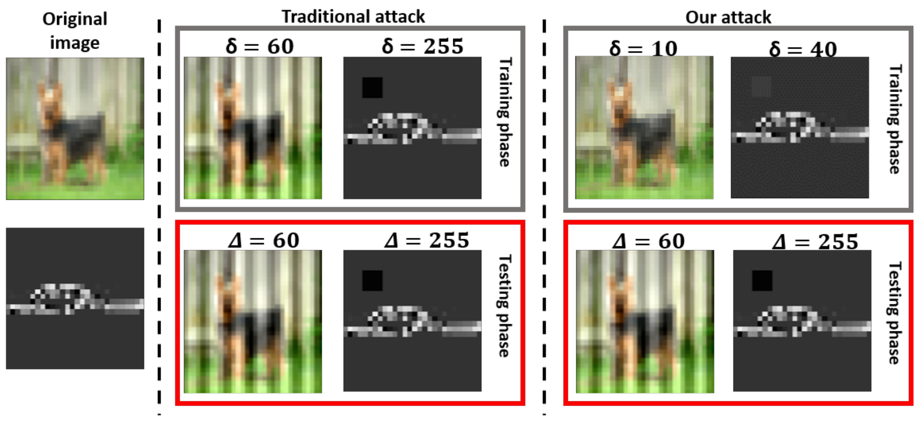


Figure 4.2: Visualization of poisoned image samples. **From left to right:** original clean image; traditional attacking method of training (top) and testing (bottom) using the same trigger intensity; our attack of training (top) with a lower trigger intensity compared to testing (bottom)

happens. The neurons that were previously exposed to the weaker trigger (t, δ) exhibit an amplified activation response. In other words, they become significantly more ”excited” when triggered by the stronger stimulus. This heightened activation within the previously poisoned neurons has a big effect on the model's decision-making process. It further reinforces the model's inclination to output the predetermined backdoored class when it encounters the stronger trigger (t, Δ). As a result, the overall performance of the backdoor attack is enhanced.

In essence, this small selection of malicious neurons combined with the strategic use of trigger intensities, allows the reinforcement and adaptability of modern backdoor attack techniques. By leveraging this behavior in NNs, attackers can infiltrate ML models with adaptable levels of stealthiness and strength.

More formally, let there be a neural network F_L with L layers defined as:

$$F_L = f_l * f_{l-1} * ... * f_1 \tag{4.5}$$

Where $0 < l < L$ and f_l are the set of neurons in each layer l. Each neuron in F_l outputs: $A(wTx + b)$ where A is the activation value, w are the weights, x is the input given to the neuron and b is the bias. Now based on [82], let there be a small set of neurons $K \in F_L$ that contribute a lot to the backdoor behavior. Given the two triggers (t, δ) and (t, Δ), we find that:

$$\begin{aligned} & A(w^T\delta + b) < A(w^T\Delta + b), \forall A \in K \\ s.t. \quad & \delta < \Delta \end{aligned} \tag{4.6}$$

As seen in equation 4.6, the elevated values produced by the activation functions across the layers of the neural network propagate through the compromised neurons K, thereby amplifying the model's inclination to produce the backdoored class.

4.2.4. Strength-Visibility tradeoff

Lowering the trigger intensity offers a significant advantage in backdoor attacks by affording attackers greater flexibility and adaptability in their strategies. As we've previously discussed, in a realistic scenario it is highly improbable that all deployed triggers will exhibit the same level of visibility as those they were initially trained on. Various external factors such as fluctuations in lighting conditions, the presence of dust particles, and other environmental elements, can significantly influence trigger visibility. For this reason, it becomes crucial for attackers to have the capability to activate their backdoors using triggers of different intensities. Essentially, the attacker can select any trigger intensity for activation, provided that the testing intensity surpasses the training intensity. This strategic choice empowers the attacker to make a trade-off between the visibility and the strength of the backdoor.

Our empirical research underscores the profound impact of trigger intensity on the performance of the attack. Increasing the testing intensity leads to an enhancement in attack effectiveness. However, this comes at a cost of increased visibility during the testing phase of the attack. It's important to note that currently, there is a lack of robust defenses capable of detecting backdoors after the aggregation phase. For that reason, visual inspection remains the primary line of defense against such attacks. Given the attacker's control over trigger intensity, this parameter serves as a crucial lever for optimizing the trade-off between the visibility and strength of the backdoor. This adaptability ensures that backdoor attacks can be tailored to specific scenarios, making them more challenging to detect and emphasizing the need for more sophisticated defenses to protect ML systems from these evolving threats.

4.3. Delta optimization

As expected, the adversary may not possess prior knowledge about the ideal attack delta value. An attack delta set too low results in suboptimal outcomes, while an excessively high value fails to fully optimize the attack's invisibility. Consequently, enabling the attacker to fine-tune this variable is essential for maximizing the attack's effectiveness. To achieve this, the attacker pursues three primary objectives: (1) Maximizing the main task accuracy score; (2) Maximizing backdoor accuracy; and (3) Minimizing trigger visibility. In pursuit of these goals, we propose the following equation:

$$\begin{aligned}\delta^* = \min_{\delta}(\beta_1 * \sum_{x_i \in D_c} \mathcal{L}(x_i, y_i)) + (\beta_2 * \sum_{x_i \in D_p} \mathcal{L}(x_i + \mathcal{T}(t, \Delta), y_i^*)) \\ + (\beta_3 * \sum_{x_i \in D_p} \mathcal{L}(x_i, x_i + \mathcal{T}(t, \delta)))\end{aligned} \tag{4.7}$$

δ^* denotes the optimal delta that we want to find. D_c and D_p are the datasets with poisoned and clean image samples x respectively. x_i is an image sample and y_i is its corresponding label. y_i* indicates the label is altered to the adversary's desired target class. $\mathcal{L}$ is the loss function. $\mathcal{T}$ is the trigger injection function that takes in the arguments t and δ, where t is the trigger pattern and δ is the trigger intensity. At last, β_x indicates the 3 variables in this equation. The attacker can choose these variables to his liking to give more importance to one of the three tasks. For instance, if the attacker finds the natural stealthiness of the attack very important, he could opt to increase β_3. Doing this makes the equation more sensitive towards changes in trigger intensity causing it to consider it more in order to establish the optimal delta.

5 Experimental setup

5.1. Experimental setup

The implementation of all the compared attacks and the FL framework is founded on the robust PyTorch library [47], which provides a powerful and flexible environment for conducting deep learning research. To ensure the reliability and scalability of our experiments, we executed them on a dedicated cluster specifically provisioned for research purposes. This cluster efficiently manages computational resources using the SLURM protocol. It allows for seamless parallel processing and efficient job scheduling. Our experiments were executed on a robust server equipped with substantial hardware resources, including a high-performance NVIDIA A40 GPU, an Intel Xeon E5-2620 CPU, and a generous 32GB of RAM. These robust hardware components collectively facilitated the execution of our experiments, enabling us to explore the nuances of backdoor attacks in FL across varying scenarios and configurations.

5.2. Datasets

Our experimental evaluations are conducted on three distinct datasets that have gained significant recognition within the research community and are prominently featured in prior studies [4, 64, 71, 80]. These datasets have been selected for their relevance to our research domain and their ability to provide comprehensive insights into the outcomes of our attack methodologies. By utilizing datasets that have been leveraged by other researchers in the same field, we ensure a degree of comparability and fairness in assessing the results of our attacks. These datasets serve as a robust foundation for our experimentation. It enables us to draw meaningful conclusions and contribute valuable insights to the existing body of knowledge in the field. Figure 5.1 shows some image samples of each of the datasets and in appendix B one can find different poison image samples for the CIFAR-10 dataset to get an intuition of the difference a delta makes for the poisoned image.

5.2.1. Fashion-MNIST (FMNIST)

Originally designed as a substitute for the original MNIST dataset, the FMNIST dataset contains photos of fashion goods like bags, shoes, and apparel. FMNIST is better appropriate for a wider range of image classification applications because it is composed of grayscale photos of different fashion items, whereas MNIST is composed of handwritten numbers. The FMNIST dataset contains 60,000 training images and 10,000 test images, similar to the MNIST dataset. Each image is 28x28 pixels in grayscale (1 channel) and is labeled with one of 10 class labels. Each class has 6000 images in the training set and 1000 images in the test set. Each image is represented by a 28x28 pixels image, and each pixel is represented by a single 8-bit integer in the range of 0 to 255 [70], where 0 represents black and 255 represents white. Table 5.1 highlights all the features of the FMNIST dataset. FMNIST dataset is similar in structure to the MNIST dataset, but it's focused on fashion items, which are more complex than handwritten digits [70]. This makes it a more challenging dataset than MNIST and it's more suitable for evaluating the performance of more complex models. It's also commonly used as a benchmark dataset for image classification tasks, as it's similar in structure to the MNIST dataset, but with more challenging classes.

Table 5.1: Features of the FMNIST Dataset

Feature	Description
Dataset Name	Fashion-MNIST (FMNIST)
Number of Classes	10
Number of Samples	70,000
Image Size	28x28 pixels
Image Channels	Grayscale (1 channel)
Sample Types	Fashion items (e.g., T-shirt, dress, shoe)
Data Split	Training (60,000 samples), Testing (10,000 samples)
Data Distribution	Balanced classes
Data Augmentation	Not provided (typically applied during preprocessing)
Purpose	Fashion item classification, benchmarking machine learning models
Source	Zalando Research

5.2.2. CIFAR-10

The CIFAR-10 dataset is a collection of images that are commonly used for training computer vision and image classification models. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class [31]. There are 50,000 training images and 10,000 test images. The images in the CIFAR-10 dataset are relatively small (32x32 pixels) and the objects in the images are centered and occupy most of the image. The dataset is designed to be challenging for current computer vision and machine learning algorithms, as the images are low resolution and the object classes are highly similar [31]. Table 5.2 shows the general features of the CIFAR-10 dataset.

Table 5.2: Features of the CIFAR-10 Dataset

Feature	Description
Dataset Name	CIFAR-10
Number of Classes	10
Number of Samples	60,000
Image Size	32x32 pixels
Image Channels	RGB (3 channels)
Sample Types	Objects and animals (e.g., cars, birds, cats)
Data Split	Training (50,000 samples), Testing (10,000 samples)
Data Distribution	Balanced classes
Data Augmentation	Commonly applied during preprocessing (e.g., random flips, rotations)
Purpose	Object recognition, image classification
Source	Canadian Institute for Advanced Research (CIFAR)

5.2.3. CIFAR-100

CIFAR-100 consists of 100 different classes, with each class containing 600 images. These classes cover a wide range of object categories, making the dataset suitable for training and evaluating machine learning models in various image recognition tasks. Each image in CIFAR-100 is of size 32x32 pixels and is colored, represented in RGB format [31].

Table 5.3: Features of the CIFAR-100 Dataset

Feature	Description
Dataset Name	CIFAR-100
Number of Classes	100
Number of Samples	60,000
Image Size	32x32 pixels
Image Channels	RGB (3 channels)
Sample Types	Various fine-grained categories (e.g., insects, trees, vehicles)
Data Split	Training (50,000 samples), Testing (10,000 samples)
Data Distribution	Balanced classes
Data Augmentation	Commonly applied during preprocessing (e.g., random flips, rotations)
Purpose	Fine-grained object recognition, image classification
Source	Canadian Institute for Advanced Research (CIFAR)

5.2.4. Tiny ImageNet

The Tiny ImageNet dataset, a derived subset of the expansive ImageNet dataset, serves as a valuable resource for scaled-down computer vision research. Comprising 200 distinct classes, each class boasts 500 training images and an additional 50 validation images, all contained within compact 64x64-pixel frames. Represented by WordNet IDs, these classes are thoughtfully organized in a hierarchical structure, facilitating more organized classification [32]. This dataset has become a favored benchmark for evaluating the efficacy of deep learning models in various computer vision tasks. While considerably smaller than its parent dataset, ImageNet, Tiny ImageNet still presents computational challenges, making it an ideal playground to assess the performance of backdoor attacks in larger and more complex datasets. In Table. 5.4 you can find an overview of the Tiny ImageNet dataset and its features.

Table 5.4: Features of the Tiny ImageNet Dataset

Feature	Description
Dataset Name	Tiny ImageNet
Number of Classes	200
Number of Samples	Approximately 120,000
Image Size	64x64 pixels
Image Channels	RGB (3 channels)
Sample Types	Various objects and categories
Data Split	Training (100.000 samples), Testing (10,000 samples)
Data Distribution	Balanced classes
Data Augmentation	Commonly applied during preprocessing (e.g., random flips, rotations)
Purpose	Image classification, fine-grained recognition
Source	Created from the ImageNet dataset

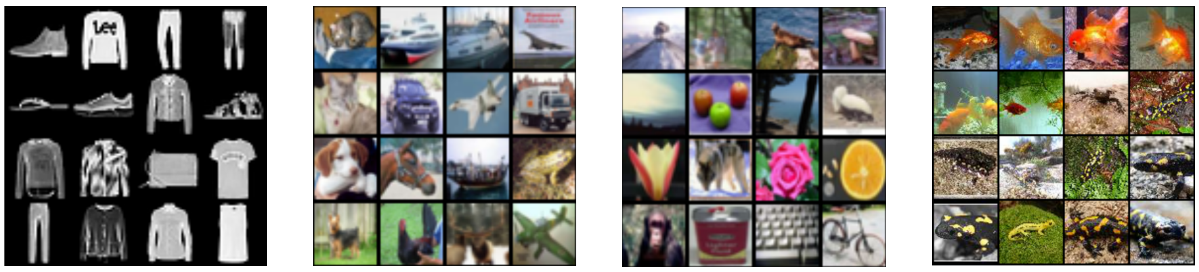


Figure 5.1: Data samples of each of the datasets used in the experiments. From left to right: FMNIST, CIFAR-10, CIFAR-100, Tiny-ImageNet

5.3. Tasks and attack settings

5.3.1. Goal of the attacker

It is the goal of the attacker to minimize the loss of the backdoor accuracy whilst also minimizing the accuracy of the main task. Equation 2.3 shows the optimization problem for the attacker. In this paper, we utilize two different trigger patterns in order to determine the properties of our attack. We use a simple square pattern and a sinus pattern (SIG) as in [5]. One important difference to keep in mind when comparing square and SIG triggers is that their operating delta value ranges are different. While the SIG trigger limits its delta to a range between 1 and 100, the square trigger operates within the range of 1 to 255. This difference results from the special properties of the SIG trigger, which can create significant changes to the original image. These changes can be so big that instead of maintaining the backdoor's intended effect, they force the global model to adjust to a new class. For that reason, it is decided to stay within the bounds of 0 to 100. Both trigger patterns were picked because they are straightforward and demonstrate how our attack enhances existing SOTA attacks. To have an idea of what these image samples look like, Appendix B provides an overview of what these poisoned image samples would look like with a given trigger intensity.

5.3.2. Attack settings

To evaluate the performance of our method in different settings, we conducted experiments using varying settings for local aggregation. Table 5.5 shows the standardized attack settings for our basic experiments. Furthermore, to understand the robustness of our attack method in response to variations in hyperparameter settings, we carefully tested it under a spectrum of attack configurations. Any deviations from the original settings in specific sections will be explicitly noted for clarity and transparency. It's worth noting that we conducted each experiment with a minimum of five repetitions. We adopted this practice to enhance the reliability and consistency of our results while mitigating the impact of inherent randomness in the outcomes.

Similarly to other works [4] [71] [80], we use FedAvg [40] in our experiments which simply performs an average of the device updates. Due to its effectiveness and simplicity, it has emerged as the optimization standard for FL deployments at scale and has thus become the de facto standard for FL [10].

Table 5.5: Standard experiment settings for each dataset

	FMNIST	**CIFAR-10**	**CIFAR-100**	**Tiny ImageNet**
Total participants	3000	1000	1000	1000
Malicious participants	1	1	1	1
participants per FL round	10	10	10	10
Phase to start the attack	From beginning	From beginning	From beginning	From beginning
Poison fraction	0.5	0.5	0.5	0.2
Default trigger pattern	SIG	SIG	SIG	SIG
Frequency of SIG pattern	6	6	6	6
Trigger size square pattern	4	5	-	-
Local epochs	2	2	2	5
Data distribution	IID	IID	IID	IID
Classification model	Classic CNN	ResNet18	ResNet18	ResNet18
Learning rate	0.1	0.1	0.1	0.001
Momentum	0.9	0.9	0.9	0.9
Type of attack	Fixed frequency	Fixed frequency	Fixed frequency	Fixed frequency

5.3.3. Models

Our evaluation setup consists of two different models. First, we use the classic CNN model as provided in [52]. This model works on the FMNIST dataset. For the other datasets, we use the ResNet18 model [26]. Table 5.6 shows the layout of our classic CNN model. The layout of the ResNet18 model can be found in [26].

Table 5.6: Layout of the classic CNN model

Parameters	Shape	Hyperparameters of Layer
Conv2d	1*32*3*3	stride = (1, 1)
GroupNorm	32*32	ϵ = 10^{-5}
Conv2d	32*64*3*3	stride = (1, 1)
GroupNorm	32*64	ϵ = 10^{-5}
Dropout2d		p = 0.5
Linear	9216*128	bias = True
Linear (For Fashion-MNIST)	128*10	bias = True

5.3.4. Threat model

In this section, we establish our threat model. It aligns with the standards outlined in prior research [4] [64] [71] [80]. Our threat model encompasses several key considerations: Firstly, the attacker possesses the capability to manipulate the weights of the model they intend to submit for aggregation. This manipulation can involve altering model parameters to inject backdoors or other malicious modifications. Secondly, the attacker has the authority to modify hyperparameters associated with the local training process. These hyperparameters include settings like the number of training epochs and learning rates. This allows the attacker to fine-tune their approach to achieve their goals. Lastly, the attacker retains the flexibility to adapt their local training strategies from one round to another, enabling them to optimize their attack approach as needed. Importantly to note is that the attacker's influence does not extend to benign participants' training processes or the aggregation technique utilized to merge updates from all participants into a consolidated global model. We operate under the assumption that benign participants adhere to standard FL training methods while applying them to their local data to construct their local models.

5.4. Metrics

Attack effectiveness

We quantify our attack's efficacy in terms of Backdoor Accuracy (BA). It evaluates a model's capacity to identify accurately data that has a backdoor pattern or concealed trigger, which is a crucial consideration in adversarial and security contexts. High backdoor accuracy means the model is detecting and reacting to the trigger correctly. Conversely, low backdoor accuracy denotes a more robust model that is less vulnerable to these kinds of attacks. The mathematical expression of the BA can be found in 2.4

Besides attaining a high BA, the second goal is to not influence the benign accuracy as well. For that reason, we also measure the Main Task Accuracy (MA) for our experiments as expressed in 2.5. MA indicates how many of the benign input samples are correctly classified. A good backdoor attack does not or only slightly degrades the MA. A high loss in MA usually indicates that the local update is malicious and thus makes it easier for the defenders to defend against such updates.

5.4.1. Irregularity of the update

Most clustering defenses in FL perform some type of distance metric. In this thesis, the two most discussed metrics are used. First, there is the cosine distance [53] [9]. The mathematical expression can be found in 6.13a. The cosine distance, which goes from -1 to 1, is used to evaluate the consistency of updates and can help separate trustworthy updates from ones that might be tainted. The process of collaborative learning can be interfered with by malevolent entities purposefully or inadvertently introducing poisoned updates. FL systems can detect outliers and deviations, which suggest the presence of poisoned updates, by tracking the cosine distance between model updates from various devices.

The second distance metric is the Euclidean distance as found in 6.13b. It calculates the straight-line distance between two updates in multidimensional space. When the attacker tweaks the weights of its own update in order to make it stronger, it will increase its Euclidean distance. Again, this will give away the poisoned updates among the benign updates and render the attack ineffective. Therefore it is important for the attack to make both their cosine distance as well as its Euclidean distance as similar to benign updates as possible.

5.4.2. Durability of the backdoor

The durability of the backdoor is expressed by the *lifespan* metric as expressed in 2.8. It is important to measure as a longer-lasting attack can also do more damage and thus prove important for an attacker. The lifetime tracks the maximum amount of rounds the backdoor lasts until the BA goes below a certain threshold. We follow the guideline of [80] to put threshold κ at 50%. This means we monitor the backdoor's effectiveness until its accuracy drops to half of its initial level.

5.4.3. Measuring natural stealthiness

Evaluating the distinguishability between two images requires the use of an appropriate metric. In the context of FL, there isn't a universally accepted standard for measuring the visibility of triggers. Therefore, in this study, we propose to utilize the PSNR, SSIM, and LPIPS. These metrics each use a different strategy to measure the visibility of differences between two images. Our goal is to deliver a comprehensive assessment of trigger visibility in FL scenarios by leveraging these well-established and reliable metrics.

PSNR provides a quantitative measure of the similarity between two images by evaluating the ratio of the peak signal power to the mean squared error between the images. The higher the PSNR value, the more similar the images are considered. It quantifies the dissimilarity between two images on a scale from 0 to 80, with a score of 80 implying no difference, while a score of 80 signifies a complete dissimilarity between two images. Its expression can be found in 2.9.

Secondly, the SSIM metric [50] takes into account luminance, contrast, and structure, mimicking aspects of human visual perception. It measures the similarity in local patterns of pixel intensities and returns a value between 0 and 1, with 1 indicating perfect similarity. Unlike PSNR, which primarily focuses on pixel-wise differences, SSIM evaluates image quality by considering how well the structure and texture of the images match. For that reason, it is known as a more sophisticated metric when comparing images. The disadvantage of using SSIM is that it was only compatible with colored images, restricting our evaluation to the CIFAR-10 dataset. The expression for the SSIM metric is found in 2.10

The LPIPS [78] metric is a perceptual similarity metric used to assess the dissimilarity between two images from the perspective of human visual perception. The metric quantifies the dissimilarity between two images on a scale from 0 to 0.7235. A score of 0 implies no difference, while a score of 0.7235 signifies a complete dissimilarity between the two images. Unlike SSIM and PSNR, which focus on pixel-wise differences, LPIPS leverages deep neural networks that are typically pre-trained on large-scale image datasets to extract high-level visual features from images. The perceptual separation between these features is then calculated, taking structure, texture, and color into account. With this method, LPIPS may catch changes between images that are more subtle and complex making it more similar to human perception. As a result, LPIPS frequently offers more precise evaluations of picture quality and similarity, especially when assessing photos that might have been altered. The disadvantage of using LPIPS was again that it only works on colored images and is only being utilized on the CIFAR-10 dataset.

5.5. Defenses

We assessed our attack against a variety of recent defensive methods. These included techniques like vector-wise scaling defenses such as in [54]. We also evaluated cluster defenses such as FLAME [43] and multi-Krum [9], as well as a robust aggregator, the Geometric Median [74]. In addition, our assessment covered sparse defenses, as found in [46]. This evaluation encompasses various defense mechanisms, providing us with valuable insights into the effectiveness of our attack strategy. For each defense, we highlight whether it was applied in IID or non-IID settings. Success in IID settings implies that the defense proved ineffective in the most straightforward scenario. If our attack is effective in performance in IID settings but encounters challenges in non-IID scenarios, it signifies that the defensive measure offers some level of protection but falls short of halting our attack in more realistic conditions. Non-IID settings were obtained by using a Dirichlet distribution with $a = 0.8$.

6 Results and discussion

6.1. Experimental results

In this section, we first experiment with some of the basic properties regarding the pixel intensities in backdoor attacks. After our first experimental results, we use these results in order to craft a new attack using the pixel intensities. We compare our attack against a baseline of a traditional attack under no defenses. The aggregator in all the experiments is FedAvg. The results provide a detailed overview of the outcomes achieved with different trigger intensities. This exploration serves as a reference point, showcasing the advantages and drawbacks associated with diverse trigger intensities within the context of FL.

6.1.1. Results under low test- and training intensities

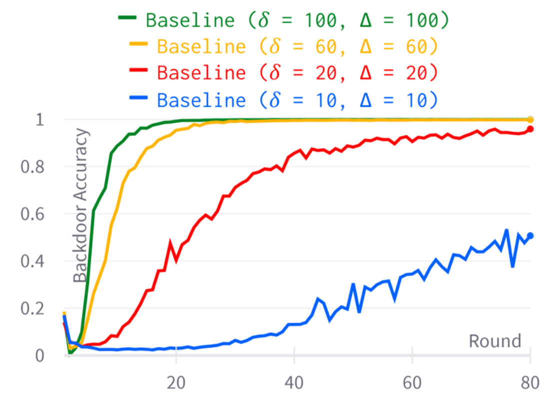
First, we conduct a basic exploration experiment to test how the BA is affected by lowering the trigger intensity in different datasets. In figure 6.1 we observe the results in BA for different trigger intensities whilst keeping the training and the testing intensity the same. This lets us explore the effectiveness of our attack on trigger intensities that are considerably lower compared to most attacks. We can observe the following. First, let us take a look at the results from the FMNIST dataset in figure 6.1a. We observe that a trigger intensity of 100 results in a BA of 100%. It quickly gets to this point with a steep line and after about 10-15 rounds we see that it has achieved its maximum BA. We see a similar pattern happening when the trigger intensity has a value of 60, with the difference being that its increase in BA at the start of the attack is less steep in comparison to the former. A clear decrease in BA emerges whenever we poison our model with trigger intensities below 60. We can see that a trigger intensity of 20 leads to a BA of about 90-95%, however reaching this score takes many rounds. For the trigger intensity of 10, we can see that the backdoor is unable to reach a BA of 50%, showing us that a lot of performance is lost by lowering this trigger intensity.

Now, if we take a look at the results under the CIFAR-10 dataset (fig. 6.1b), we can see that both attacks using trigger intensity of 60 and 100 manage to reach about 100% BA in a relatively short time, the former being slightly below the latter's BA. From analyzing the results for trigger intensities below 60, we can see that both runs, using trigger intensities of 20 and 10 respectively, have a difficult start. It takes about 20 rounds before their BA starts a rapid incline up to their maximum performance reached after about 25-30 rounds. It can be observed that a trigger intensity of 20 manages to attain a BA of 95% and a trigger intensity of 10 shows a loss in performance with its maximum BA being about 80%.

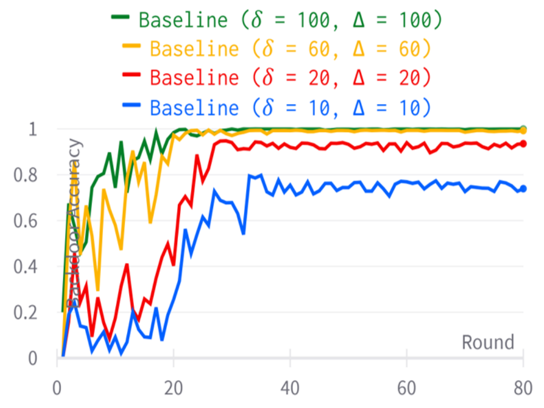
Next, the results for the CIFAR-100 dataset are up in figure 6.1c. We observe that all runs have difficulty with this dataset, with no run managing to reach 100% and all runs needing at least 20 rounds before the BA starts to rise steeply. We observe that trigger intensities of 100 and 60 manage to reach a BA of 97% and 90% respectively, whilst the experiments with trigger intensities of 20 and 10 achieve backdoor accuracies of 35% and 8%.

At last, we observe that there are no very significant differences in the Tiny ImageNet dataset in figure 6.1d. The runs with 100, 60, and 20 trigger intensities show a similar pattern in terms of growth and maximum performance in BA. We observe that it takes them all about 30 rounds to reach their maximum BA of 100%. The only slight observable difference is that the run with a trigger intensity of 10, seems to show a lower BA during its climb to its maximum in comparison with the other runs.

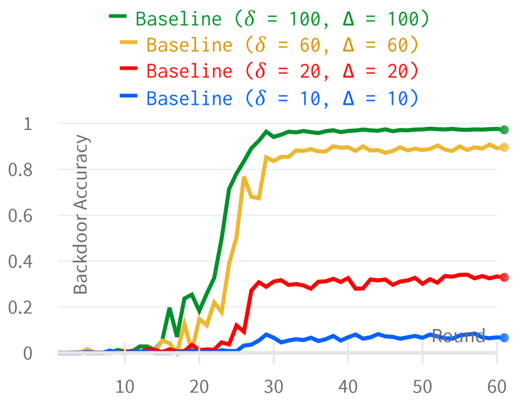
The loss of performance from lowering the trigger intensities can be attributed to the loss in effectiveness from the activation values. Due to the reduced trigger intensities, the model has a harder time recognizing the trigger compared with the benign features of benign samples. As a result, the influence of benign features increases in comparison to the influence of the backdoor trigger, making it less and less likely the model is able to classify the backdoor correctly. It also becomes apparent that some datasets are harder to poison in comparison with others. This happens due to differences in resolution, model used, and the amount of samples per class in the dataset.



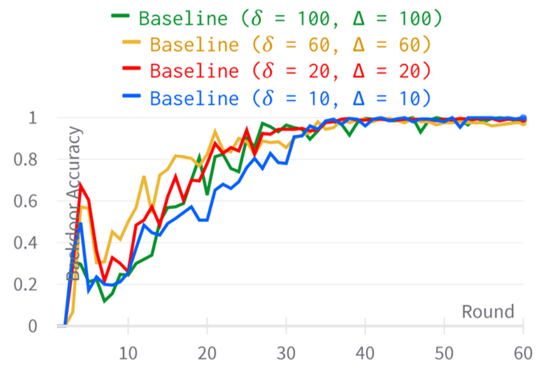
(a) FMNIST



(b) CIFAR-10



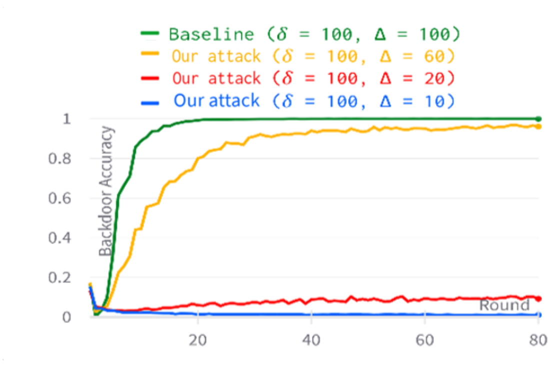
(c) CIFAR-100



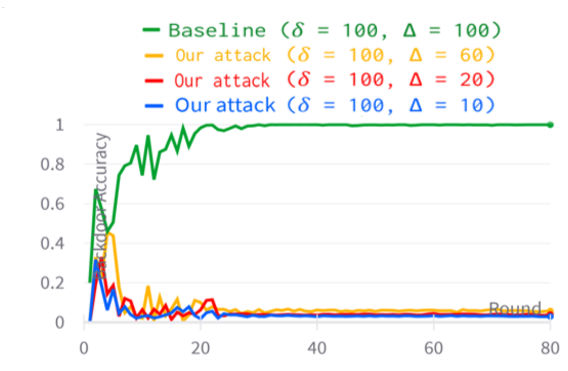
(d) Tiny ImageNet

Figure 6.1: Attack performance under different trigger intensities. Both the training and testing intensity are equal in this experiment

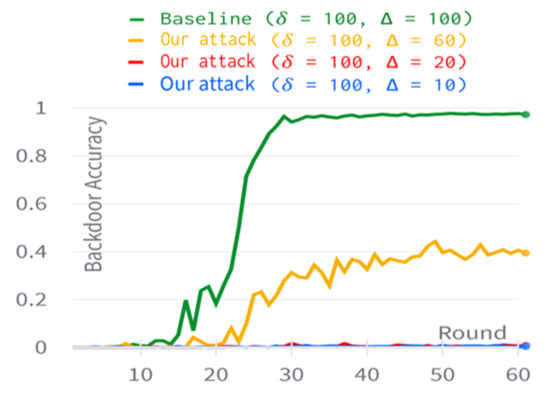
6.1.2. High training intensity and low testing intensity



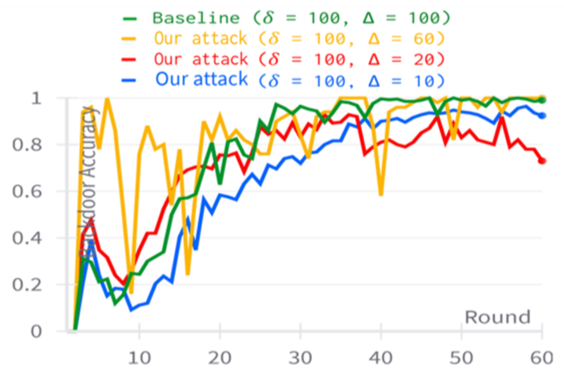
(a) FMNIST



(b) CIFAR-10



(c) CIFAR-100



(d) Tiny ImageNet

Figure 6.2: Attack performance under a high training intensity and a low testing intensity

In this section, we delve into experimenting using a high training intensity and a lower testing intensity. The results are depicted in Figure 6.2. In all experiments, we train our local model with a training intensity of 100, whilst subsequently using a testing intensity of 100, 60, 20, and 10. When focusing on the FMNIST dataset in figure 6.2a, the results indicate diminishing effectiveness as we lower the testing intensity. We see that using the same training- and testing intensity leads to a strong BA of 100% that quickly increases over rounds. However, when presented with a lower testing intensity, we notice that the backdoor quickly loses performance. A testing intensity of 60 results in a BA of 95% which also takes longer to reach. When using a testing intensity of 20 and 10 we see that the BA quickly drops, being unable to get a BA that surpasses 10%. We observe a similar pattern when looking at the results from CIFAR-10 in figure 6.2b. We observe that using 100 for both training and testing intensity leads to a strong result. However, when we lower the testing intensity in comparison to what it was trained on, we notice that the BA completely vanishes, being reduced to 0%.

Examining the outcomes for CIFAR-100 and Tiny ImageNet datasets, we observe a similar trend. In the CIFAR-100 results, depicted in Figure 6.2c, we notice a rapid decline in performance when validating with a testing intensity lower than the training intensity. Specifically, using a testing intensity of 60 results in a BA of about 40%, while testing intensities of 10 and 20 yield no BA at all. For Tiny ImageNet, this performance drop is a bit subtler. As seen in Figure 6.2d, evaluating with an intensity of 60 leads to an unstable BA. Checking the results for testing intensities 10 and 20, we notice a slight decline in performance compared to training and testing with the same intensity.

In all datasets, we notice that there is a poor generalization that could essentially eradicate the backdoor's efficacy whenever the testing intensity is lower than the intensity it was being trained. This phenomenon can be attributed to the model's neurons expecting a robust trigger, rendering them ill-prepared to recognize a weaker trigger. Consequently, in real-world scenarios, it becomes crucial to ensure that there is no loss of trigger intensity, as any reduction could rapidly render the backdoor ineffectual.

6.1.3. Low training intensity and high testing intensity

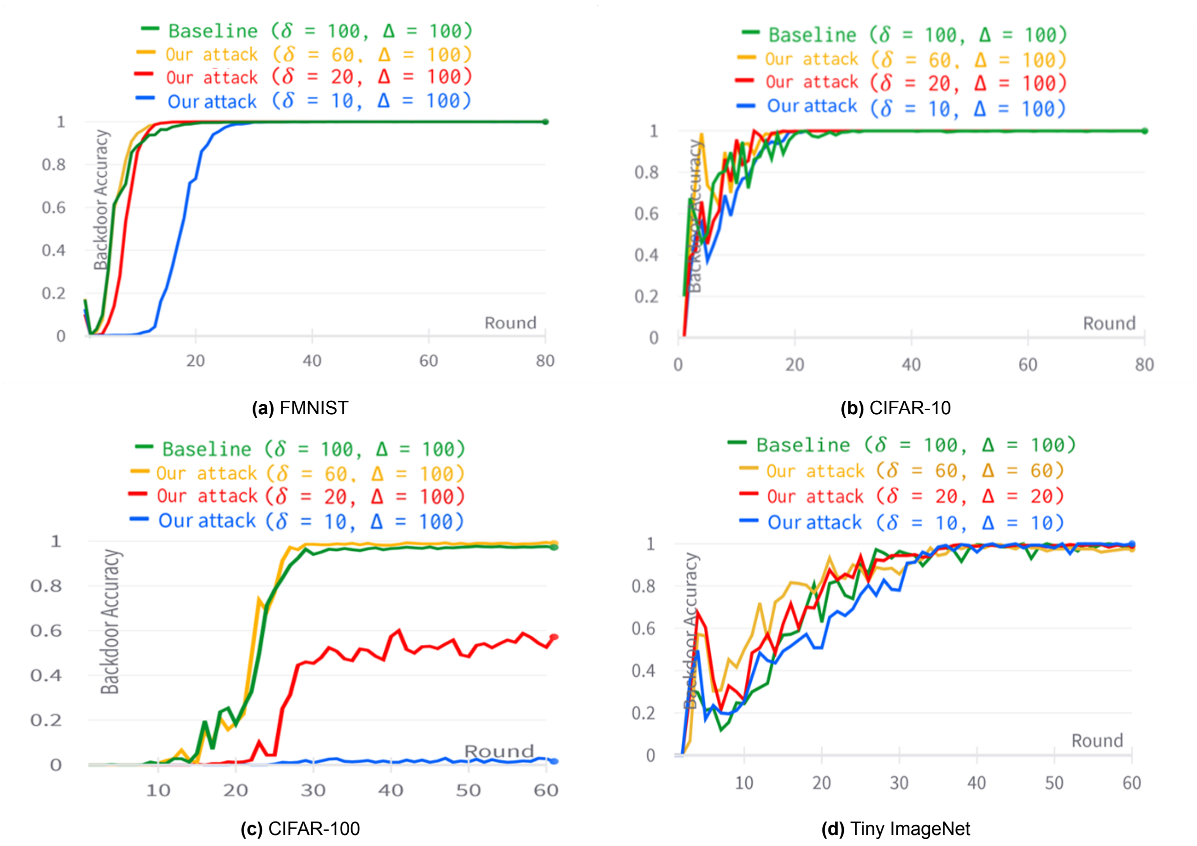


Figure 6.3: Attack performance under a low training intensity and a high testing intensity

Next, we turn our attention to the effects of training with a lower trigger intensity than that employed during testing. This experiment is visually presented in Figure 6.3. In contrast to the previous section's findings, training with a low-intensity trigger and subsequently testing with a higher-intensity trigger yields interesting results in terms of generalization. We observe this result in both the FMNIST and the CIFAR-10 datasets. In FMNIST, all runs, except the run with training intensity 10, start with a steep incline in BA which eventually results in a BA of 100%. We see that the run with training intensity 10 takes longer and kicks off after about 13 rounds. Quickly after, it reaches 100% in BA as well. Taking a look at the results for CIFAR-10, we can see that essentially all runs quickly come into effect and reach 100% BA after 10-15 rounds. They all follow a somewhat similar pattern and reach maximum performance at the same time.

Now, turning our attention to Figure 6.3c and 6.3d showcasing the results for CIFAR-100 and Tiny ImageNet, respectively. Notably, when training with an intensity of 60 and testing on intensity 100 in CIFAR-100 (Figure 6.3c), we observe impressive outcomes with a perfect BA of 100%. Even training with an intensity of 20 results in a respectable BA of about 50%. However, it's worth highlighting that this dataset proves to be the most challenging to poison, as training with a trigger intensity of 10 leads to no effective BA. Shifting our focus to Tiny ImageNet, we observe a consistent pattern across all runs in Figure 6.3d. They steadily climb in BA until reaching 100% around the 30th round, with no noticeable decline in BA observed.

What is remarkable about these results is that they indicate that irrespective of the chosen training intensity, we consistently achieve a BA of 100% for almost all runs. This observation signifies that an attacker who trains with a lower intensity gains the flexibility to select any testing intensity exceeding the training intensity while maintaining peak performance. The generalization is a result of the small set of neurons being extra stimulated whenever presented with a stronger trigger in comparison with what they were trained on. This adaptability empowers the attacker to tailor the attack precisely to their desired parameters. Additionally, while most attacks exhibit similar convergence rates toward maximum BA, the FMNIST and CIFAR-100 datasets stands out. In both cases, the attack trained with an intensity of 10 appears to be slowed down or not effective at all. This suggests the existence of a lower performance threshold, implying that excessively low training intensity might hinder the attack's effectiveness.

6.1.4. Exploring the Impact of Varying Trigger Intensities

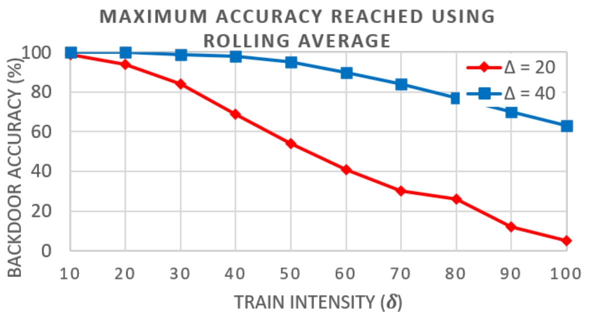


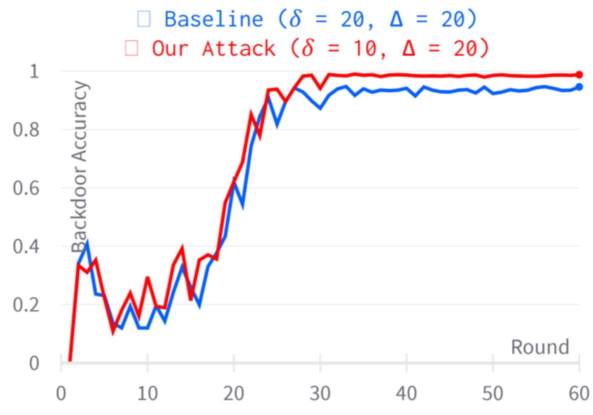
Figure 6.4: Attack performance on CIFAR-10 using different attack intensities

Our investigation, as illustrated in figure 6.4, shows the influence of altering trigger intensities during the training phase on BA, while keeping the testing intensity consistently fixed. One striking observation is the rapid decline in BA when the model is subjected to testing with a higher intensity trigger than it was originally trained on as also shown in the previous sections. This decline becomes apparent at attack intensities 20 for the red line and 40 for the blue line in the graph. As we can see, the larger the testing intensity is in comparison with the training intensity the more performance is lost. This implies that when experimenting with a testing intensity of 40 (indicated by the blue line), the effectiveness of our attack may decrease to 60%, as opposed to the potential 100% effectiveness achievable if training were conducted at an intensity of 40 or lower. If we look at the red line, we see what happens if we use a testing intensity of 20 whilst training on different training intensities. Again, a consistent trend emerges: whenever the training intensity surpasses the testing intensity, a noticeable decline in performance starts, with the potential for the complete loss of the backdoor if trained at an intensity of 100.

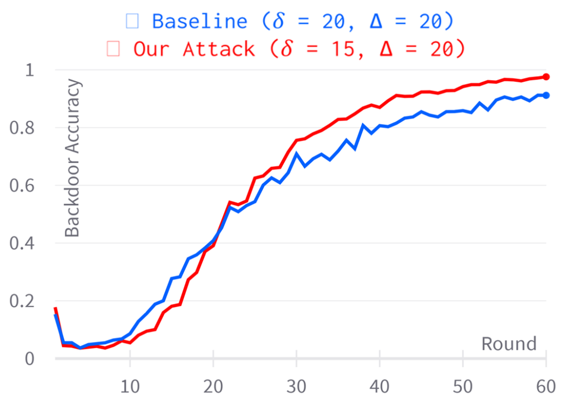
On the other hand, training a backdoor within a model with a training intensity lower than the testing intensity yields interesting results. In such scenarios, we observe either an equivalent or increased BA. This outcome signifies improved generalization, as demonstrated in the figure by both lines exhibiting a slight increase in BA when we train with a trigger intensity lower than the testing intensity. This phenomenon can be attributed to the controlled propagation of the poisoned backdoor neurons, as elucidated in Section 4.1. By training with a lower-intensity trigger, we effectively temper the influence of the backdoor during training. This allows the model to maintain its performance when subsequently exposed to higher-intensity triggers during testing.

An important takeaway from these findings is the versatility offered to attackers in achieving a high BA. This can be accomplished using various training intensity values, as long as the testing intensity remains higher. This flexibility empowers the attacker to customize the trigger intensity on a per-sample basis, tailoring the attack for optimal effectiveness. Moreover, this adaptability empowers the attack's robustness in real-life scenarios characterized by fluctuations in external conditions, where testing intensities may vary due to factors like lighting or environmental changes.

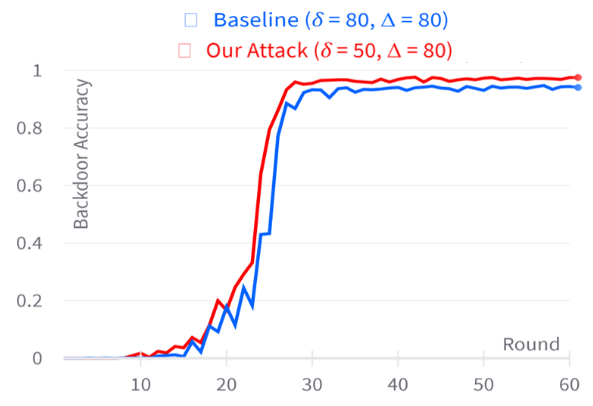
6.1.5. Reinforcement of attacks



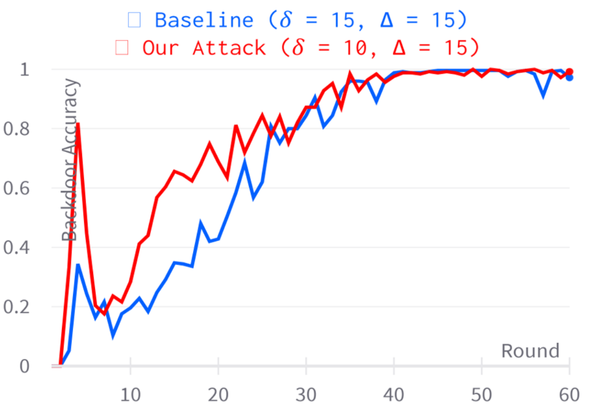
(a) CIFAR-10



(b) FMNIST



(c) CIFAR-100



(d) Tiny ImageNet

Figure 6.5: Attack performance under fedAvg

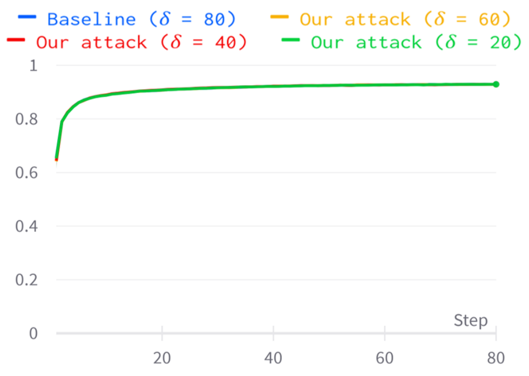
Figure 6.5 offers a comprehensive view of the BA across multiple datasets, revealing a vast and consistent trend. Our research demonstrates that the principles of our backdoor attack exhibit great generalization, holding across diverse datasets. Remarkably, training with a weaker training intensity and subsequently testing with a stronger testing intensity can lead to slightly superior results compared to an attack that employs identical trigger intensities for both training and testing phases. This phenomenon is evident in the results from multiple datasets.

For the CIFAR-10 dataset, the results are shown in figure 6.5a. We observe that the baseline attack starts to increase its BA after around 20. It manages to do so until it stops at around 90% after which it does not gain any BA anymore. Our attack, which was trained on a lower trigger intensity, starts in the same way and follows about the same trend as the baseline. However, whilst the baseline manages to reach a BA of 90%, our attack surpasses this maximum and performs slightly better. Our attack managed to reach the full 100% in BA. If we look at the results from FMNIST, we notice that the baseline manages to reach a BA of 90% at the end of the run, which took about 60 rounds to reach. Comparing it with our attack, we see that it took very slightly longer to gain quickly BA. However, our attack surpasses the baseline at round 22 and manages to get a slight increase of 5% in terms of BA in the end. Our approach manages to surpass the achieved BA of the baseline attack by at least 5% in both datasets. The key to this success lies in the strategic choice of trigger intensities. By training the model with a lower training intensity and then subjecting it to a higher testing intensity, the attack gains reinforcement, resulting in a slightly more potent BA. This strategic maneuver can be particularly advantageous for imperceptible triggers, which may possess limited strength due to their low visibility. By enhancing their impact through this approach, attackers can maximize the effectiveness of their backdoor attacks.

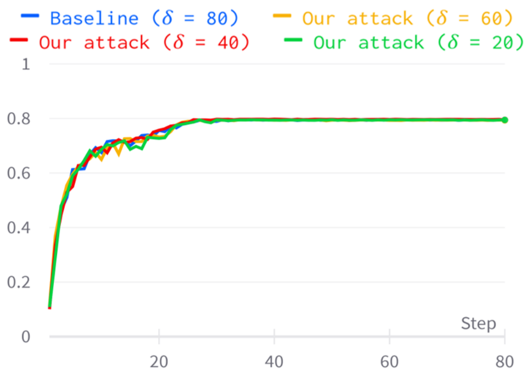
When looking at the results from CIFAR-100 and Tiny ImageNet, we notice a somewhat similar result. Like our CIFAR-100 experiments before, we notice that it takes about 20 rounds until the attacker manages to gain a steep increase in BA, as observed from figure 6.5c. Also here we observe that there is a slight increase in BA performance in our attack. The baseline attack manages to reach 93% in BA, whereas our attack improves on this with 5% to 98%. It should be noted that in this case, both the training and testing intensity are quite high (>50). The reason is that the CIFAR-100 dataset is much harder to poison, requiring stronger trigger values in general for the backdoor to gain effectiveness. As for the results of the Tiny ImageNet dataset depicted in figure 6.5d, we observe that both the baseline and our attack reach the same BA of 100% after 35 rounds. Yet, it does show that our attack is more effective in the early stages before reaching its maximum performance as we see a steeper growth in BA from round 10-20.

All in all, it shows that our method of training on a lower trigger intensity in comparison to what it is being tested on shows generalization among all datasets, achieving at least the same BA as the baseline attack. Moreover, we see that in some datasets it can result in a slight increase in BA. We believe this is the case because the model is trained on the backdoor task when presented with a lower trigger intensity. When presented with a trigger with a higher trigger intensity, the model is even more inclined to believe that it is the backdoor label. Yet, as we have seen, this increase is not always consistent among datasets and can be quite small in some cases.

6.1.6. Impact on benign accuracy



(a) FMNIST



(b) CIFAR-10

Figure 6.6: The influence on the MA when training on a lower trigger intensity

In this section, we explore whether the inclusion of a backdoor task has any adverse effects on the validation accuracy of the MTA. The MTA's validation accuracy is a critical metric as it serves as a primary indicator of the primary task's performance within the FL framework. Some experiments were conducted on different trigger intensities and investigated what the influence would be on the MTA of the global model. In figure 6.6a we see the resulting MTA under the FMNIST dataset. We can see that our MTA quickly climbs up to 90% and remains slightly above that percentage in the end. We also noticed that lowering the training intensity does not result in a loss of MTA. We see a similar result under CIFAR-10 as depicted in figure 6.6b. The MTA in all cases quickly climbs up and converges at a score of 80% at round 20. There is no clear degradation in the MTA due to the concurrent presence of the backdoor task. This observation holds consistently across both datasets we examined. Our finding emphasizes that our attack does not worsen the stealthiness of the baseline attacks in terms of MTA. This robustness is a significant achievement as any degradation in the MTA's accuracy could serve as an anomaly, potentially signaling a breach in the system's security.

6.1.7. Influence on durability

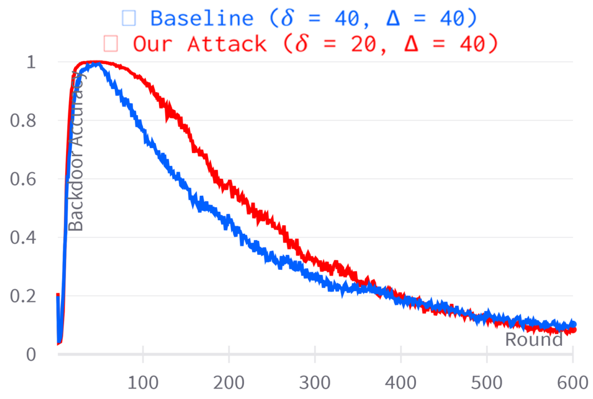


Figure 6.7: Lifetime of our attack on FMNIST

Our research explores durability, a key feature of backdoor attacks, by examining the impact of our attack using a different intensity for training and testing against the baseline attack. In this experiment, we initiated both the baseline and our attack, each spanning 50 rounds of training. The training continued until the BA reached a full 100%, indicating successful convergence. Following this intensive training phase, we ceased the attack and transitioned to benign updates only. During this phase, benign updates gradually took precedence and started to override the effects of the backdoor attack. This way of experimenting is similarly done in [80]. It is important to note that this attack was only conducted on the FMNIST dataset and not the other datasets. The CIFAR-100 and Tiny ImageNet dataset were too complex to run this experiment and running it would take weeks. CIFAR-10 displayed no sign of losing performance after a significant amount of time, thus making it also exceed our time constraints. We measure our durability by using the lifetime metric. This metric counts how many rounds the backdoor manages to reach a BA of 50% or higher. The difference between the two attacks gives an indication of the increase in lifetime.

The findings from figure 6.7 reveal that our attack when compared to the baseline, exhibits quite an extension in the lifetime of the backdoor. We can observe this by looking at the figure. We see that the red line (our attack) manages to hold onto its backdoor accuracy longer in comparison with the baseline attack from the blue line. We notice that the blue line reaches a BA of less than 50% after 125 rounds. Our attack extends this number up to 173 rounds. As a result, the lifetime of our attack is increased by 39%. This means that once our attack is embedded into the global model, it persists and operates for an extended duration. This shows that it can significantly surpass the lifetime of the baseline attack. This observation carries significant implications, especially from a security standpoint. A backdoor attack's durability is directly correlated with its potential for causing damage or unauthorized access. By extending the attack's lifetime, we enhance its potential impact and increase the window of vulnerability for the targeted system or application.

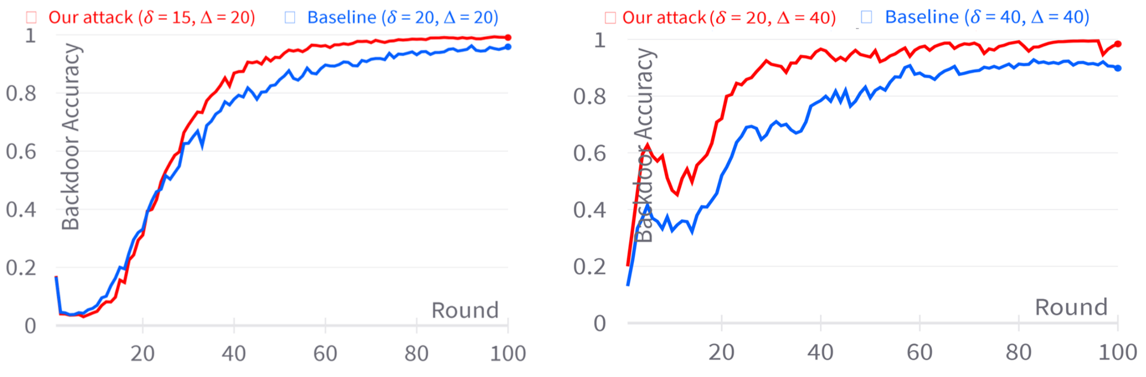
6.2. Attack results under defenses

To comprehensively assess the effectiveness of our attack strategy, we subjected it to testing in the presence of SOTA FL defenses. These defenses include norm clipping [54], Geometric median [74], Multi-Krum [9], and FLAME [43]. The objective was to understand how our attack performs under a range of defense scenarios within a fixed-frequency FL setting. In our experimental setup, we confronted our attack with a baseline model that shares the same attack and testing intensity settings. It's essential to note that while our primary contribution revolves around an attack strategy that offers adaptability and resilience for the adversary after aggregation, our primary goal isn't to evade detection during the aggregation process. Instead, our focus lies in demonstrating that our attack still can be employed even under SOTA defenses.

One crucial aspect of our evaluation is that the success of our attack may depend on the baseline model's ability to effectively deploy the backdoor. This highlights the interplay between our attack and the baseline. Nevertheless, our experiments enable us to showcase that the principles of employing varying trigger intensities for training and testing remain applicable, even when robust FL defenses are deployed. It's noteworthy that our attack strategy is designed to thrive in scenarios where adaptability and persistence are key objectives. By incorporating the element of variable trigger intensities, we enable attackers to maintain the attack's effectiveness even in the face of advanced defenses.

The findings from our experiments hold significant implications for the broader field of adversarial machine learning and FL security. They highlight the need for continuous innovation in defense mechanisms to counter evolving attack strategies. Moreover, they emphasize the importance of proactive security measures to safeguard FL systems against sophisticated adversaries.

6.2.1. Performance under Vector-wise Scaling



(a) FMNIST **(b)** CIFAR-10

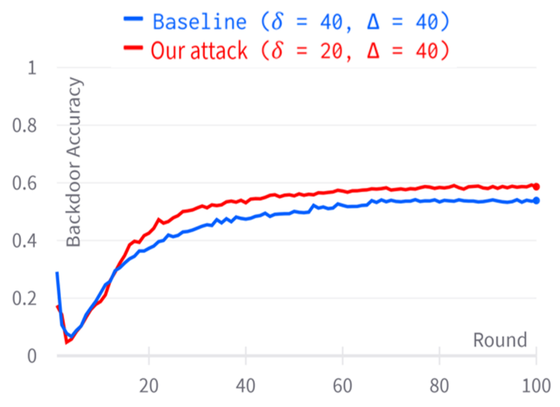
Figure 6.8: Attack performance under Norm-clipping defense

In our next experiment, we employ a defense mechanism known as "norm clipping." This defense strategy has garnered acclaim for its effectiveness in stopping previous attacks, as substantiated by prior research [54]. Our integration of norm clipping into our FL framework is dynamic to strengthen the system's security and resilience against malicious exploits.

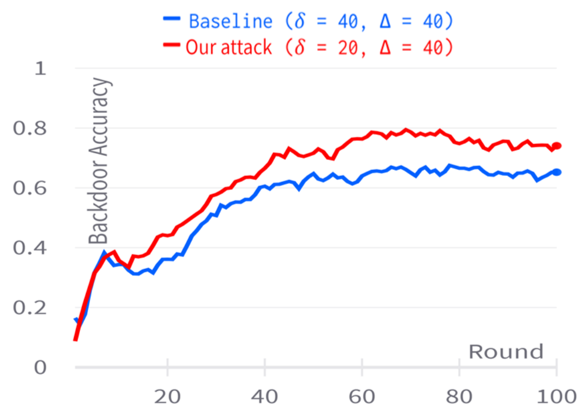
Norm clipping operates on a vector-wise scaling principle and is strategically applied to all updates at the server side before the FedAvg process commences. With this proactive approach, modifications to the local model (like scaling) are limited and kept within pre-established bounds. This prevents them from having an excessive impact on the model. A distinctive feature of norm clipping is its adaptability. It engages in a dynamic process wherein it continuously computes the average norm magnitude based on the incoming updates. Subsequently, this computed average norm is set as the norm bound for the current iteration of FL. This adaptability ensures that the defense mechanism remains responsive to changes in the nature and magnitude of incoming updates, enhancing its effectiveness in safeguarding the model.

Our evaluation of the impact of norm clipping on the performance of our backdoor attack, as illustrated in Figures 6.8a and 6.8b, offers valuable insights. It is evident that as expected, norm clipping exerts a slowing effect on the convergence of both the baseline attack and our attack. However, even under the constraints imposed by norm clipping defense, the effectiveness of our attack strategy shines through. For instance, figure 6.8a shows that both the baseline and our attack grow simultaneously in BA up until round 15. Starting from round 15, our attack consistently outperforms the baseline by 5% in terms of BA. Specifically, our attack achieves a 60% BA, while the baseline attains 54% in BA. If we look at the results of the CIFAR-10 dataset, we notice that our proposed backdoor attack consistently outperforms the baseline attack in terms of attack success rate, marking an even more significant achievement. For instance, in the CIFAR-10 dataset, our attack demonstrates an improvement of approximately 20% in the backdoor attack success rate when compared to the baseline attack. This outcome underscores the potency of our approach in achieving higher attack effectiveness even in the face of formidable defense mechanisms like norm clipping. Our attack stays after 40 rounds around a BA of 70-80%. The baseline only manages to reach slightly above 60% in BA. The reason for this more significant increase in BA is due to

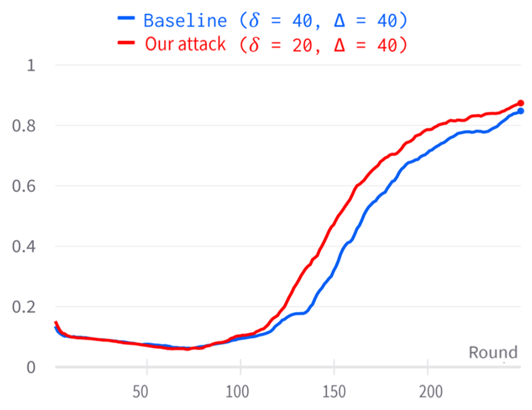
6.2.2. Performance under Cluster-based defenses



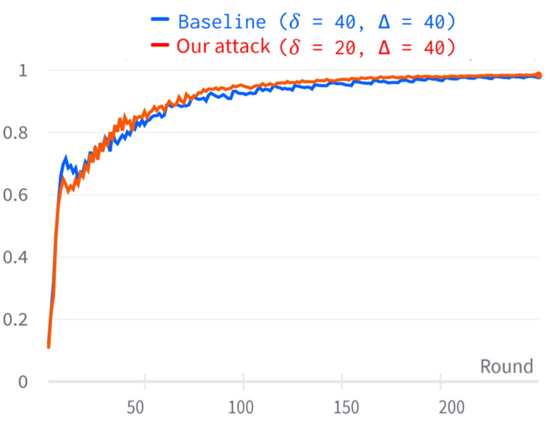
(a) FMNIST



(b) CIFAR-10



(c) FMNIST



(d) CIFAR-10

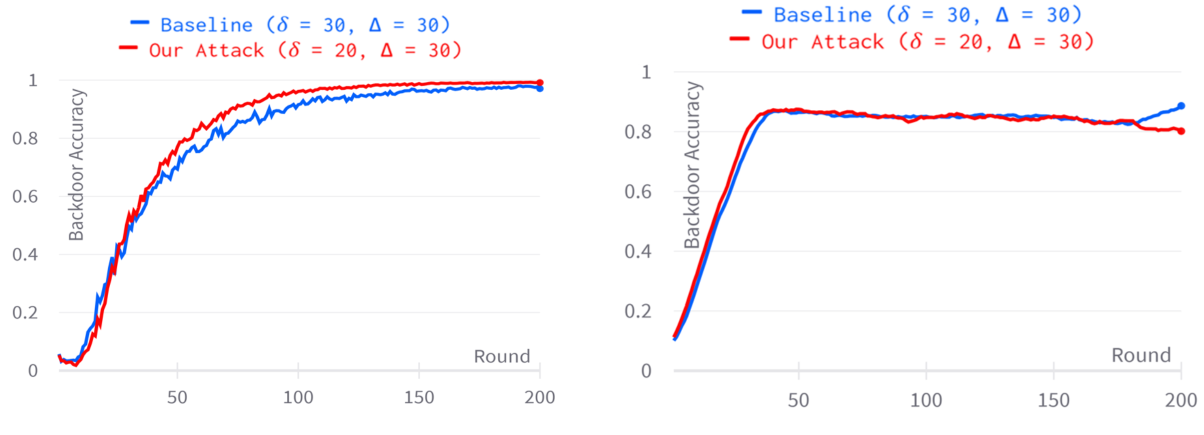
Figure 6.9: Attack performance under Multi-Krum (a, b) and FLAME defense (c, d)

In our evaluation of FL defenses, the Multi-Krum defense emerges as a formidable challenge for our attack strategy. It's essential to note that the accuracy of our attack strategy under Multi-Krum defense can be significantly diminished. This reduction in BA can be attributed to the fact that our attack does not inherently make the malicious updates submitted by the attacker more stealthy compared to the baseline attack.

Our attack approach focuses on adaptability and robustness rather than attempting to obscure the malicious nature of updates. Consequently, under the eye of Multi-Krum defense, which is strong at identifying and isolating outliers, our attack can be exposed and its accuracy compromised. However, it's important to highlight a crucial aspect of our attack strategy. Once an attack successfully evades detection by Multi-Krum defense, it remains to do so with full effect. As a result, there is a substantial boost in BA compared to the baseline attack. This observation is evident in the results presented in Figures 6.9a and 6.9b. For the FMNIST dataset, the increase in BA is moderate, with an approximate 5% improvement when compared to the baseline attack. However, the results for the CIFAR-10 dataset show a more notorious result. Under Multi-Krum defense, our attack demonstrates a notably stronger performance, achieving an increase in BA of approximately 15-20%. This outcome underscores the resilience and adaptability of our attack strategy even in scenarios where formidable defenses like Multi-Krum are in place.

The results under the FLAME defense are shown in Fig. 6.9c and 6.9d. As we can observe, our attack slightly outperforms the baseline in the FMNIST dataset. As seen from the figure, it manages to increase its BA consistently over time and slightly outperforms the baseline. However, it is by no means in big numbers. On the CIFAR-10 results, we can see that our attack consistently achieves the same or similar BA in comparison with the baseline. We notice that both attacks jump quickly to about 60% in BA and subsequently increase until both have reached about 100% BA. It seems clear that our attack manages to maintain its properties of being able to generalize well. However, The increase in BA seems to be reduced due to the FLAME defense.

6.2.3. Performance under robust aggregators



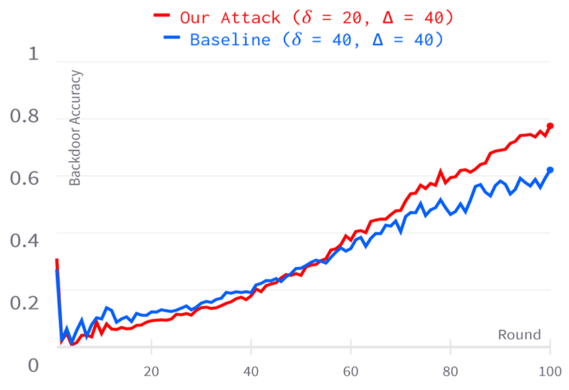
(a) FMNIST **(b)** CIFAR-10

Figure 6.10: Attack performance under the Geometric Median

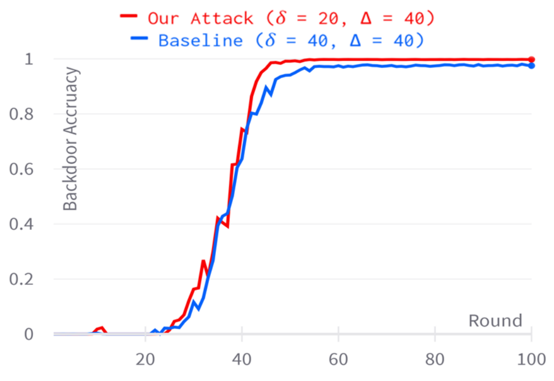
In our exploration of FL defenses, the Geometric Median defense is as a robust safeguard against adversarial attacks. The core principle behind this defense is its ability to mitigate the impact of extreme model updates on the final aggregated model. Calculating the geometric median effectively reduces the influence of outliers and noisy updates, making the FL system more resistant to adversarial attacks.

Figures 6.10a and 6.10b provide insights into the outcomes of our attack strategy under the protection of the Geometric Median defense. For the FMNIST dataset, our attack strategy demonstrates near-equal performance in terms of BA and convergence rate compared to the baseline. We see an increasing BA up to around 50. From that point on, the BA does still increase however at a slower rate. This result indicates that our attack doesn't yield a significant advantage over the baseline. Nevertheless, the generalization effect remains prominent under the Geometric Median defense, and the attained outcomes align closely with those of the baseline. In the case of the CIFAR-10 dataset, we do observe an increase in attack performance when compared to the baseline. However, it's important to note that this increase exhibits some instability and is relatively lower compared to the improvements witnessed under other defense mechanisms. While our attack still works under the Geometric Median defense, the degree of enhancement is considerably lower in comparison with other defenses.

6.2.4. Performance under sparse defenses



(a) FMNIST



(b) Cifar-10

Figure 6.11: Attack performance under the Sparsefed defense

Figure 6.11 provides an illustration of our attack within the context of the Sparsefed defense applied to both the FMNIST and CIFAR-10 datasets. Notably, the Sparsefed defense demonstrates considerable efficacy in safeguarding the FMNIST dataset, as observed in figure 6.11a. In this scenario, both our attack and the baseline exhibit a gradual increase in backdoor accuracy until approximately round 60, reaching a BA of around 35% BA. Beyond this point, our attack outperforms the baseline. It shows a steeper ascent in BA. Ultimately, our attack attains a BA of 80%, while the baseline lingers at 60%.

Turning our attention to the results about the CIFAR-10 dataset, as depicted in figure 6.11b, we encounter a different narrative. The Sparsefed defense, in this case, proves to be less effective in mitigating the backdoor threat. Although it succeeds in maintaining low backdoor accuracy for both attacks, a significant spike in BA becomes evident from round 25 onward, culminating in a flawless 100% BA for our attack by round 40. Meanwhile, the baseline attains a BA of 95% around round 50 and beyond.

Based on our results, it becomes evident that the Sparsefed defense has only limited success in curbing the impact of our attack. This success can be partly attributed to the employment of a smaller model for the FMNIST dataset, which makes it more challenging for the defense to prevent backdoor attacks due to the model's increased sparsity. However, what remains consistent across both datasets is our attack's resilience, maintaining its generalization capabilities. Notably, our attack consistently outperforms the baseline, showcasing the enduring effectiveness of our attack strategy and the inability of current defense mechanisms to entirely neutralize its impact.

6.2.5. Robustness against human inspection

As outlined in preceding sections, the removal of a backdoor from FL systems remains a formidable challenge. Very few defenses proved effective and once a backdoor has been successfully inserted into the system, its elimination becomes a difficult task. Consequently, the insertion of an imperceptible backdoor into an FL system poses a significant challenge, as distinguishing between random errors and malicious outcomes becomes hard. To address this challenge, we explored the feasibility of human visual inspection as a potential defense strategy. To initiate this investigation, we conducted a series of experiments. Equation 4.7 helped determine the optimal attack delta for our backdoor attack. This optimal delta represents the lowest feasible intensity level that an attacker could employ to execute a backdoor attack. We compared this optimized trigger with one using the maximal trigger intensity similarly to other attacks.

As one can imagine, vision is a subjective topic. To provide a comprehensive assessment, we have adopted multiple metrics, as elaborated in Section 5.5.4, for classifying the perceptibility of the poisoned image samples. In the context of the FMNIST dataset, the PSNR metric is our only option, given its compatibility with black-and-white images. However, the other two metrics are not compatible with

Table 6.1: The minimum visibility on FMNIST whilst retaining the backdoor (An upward arrow signifies a higher score, indicating a less visible trigger, while a downward arrow denotes a more visible trigger)

Trigger	PSNR ↑
SIG - baseline ($\delta = 100$)	13.56
SIG - our attack ($\delta = 14$)	**30.55**
Square - baseline ($\delta = 255$)	16.90
Square - our attack ($\delta = 20$)	**39.02**

Table 6.2: The minimum visibility on CIFAR-10 whilst retaining the backdoor (An upward arrow signifies a higher score, indicating a less visible trigger, while a downward arrow denotes a more visible trigger)

Trigger	PSNR ↑	SSIM ↑	LPIPS ↓
SIG - baseline ($\delta = 100$)	11.99	0.256	0.257
SIG - our attack ($\delta = 10$)	**31.00**	**0.942**	**0.018**
Square - baseline ($\delta = 255$)	23.81	0.942	0.018
Square - our attack ($\delta = 30$)	**35.15**	**0.987**	**0.001**

black-and-white tasks. Our experimentation encompassed both the SIG and square trigger variations. It's important to note that when using the PSNR metric, the results might be somewhat biased. Our experiments showed that this metric tends to be stricter when assessing larger triggers like the SIG trigger, compared to smaller ones. More information on this issue is provided in the limitations section.

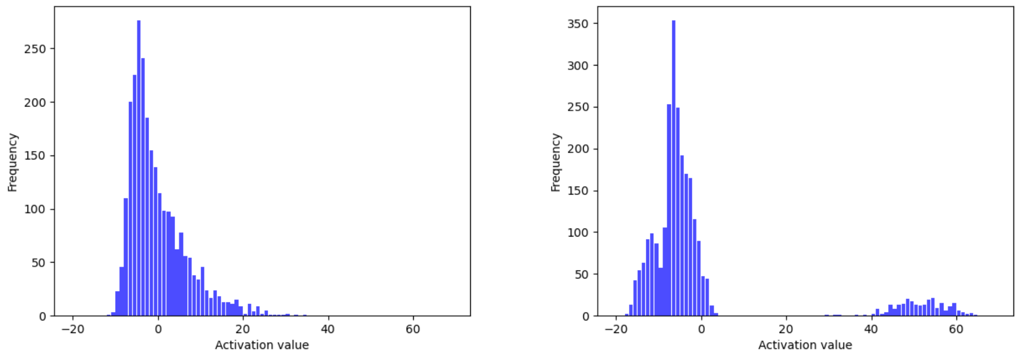
In Table 6.1, we present the results obtained for the FMNIST dataset. Our optimization process identified the optimal training intensities as 14 and 50 for the SIG and square triggers, respectively. Both demonstrate a significant enhancement compared to the baseline while preserving their original performance. Specifically, the PSNR score shows notable improvement, rising from 13.56 to 30.55 for the SIG trigger and from 16.90 to 39.02 for the square trigger. This implies that the attacker gains increased flexibility in customizing the trigger for its intended purpose while still maintaining the original performance of the backdoor.

Table 6.2 provides us with the different scores for the optimal triggers and their respective visibility. Using the SIG trigger, an optimal training intensity of 10 was achieved. This is significantly better in comparison with the original backdoor trigger with a training intensity of 100. Given the three metrics, we can see that our attack significantly improved in comparison with the baseline on all of them. The PSNR score improved from 11.99 to 31.00. The SSIM and LPIPS metrics even showed a greater improvement by going from 0.256 to 0.942 and 0.257 to 0.018 respectively. Examining the outcomes of the square trigger, it becomes apparent that the square trigger is slightly less effective than the SIG trigger in the context of CIFAR-10. Our optimization process leads to the optimal training intensity of 30. Yet, we observe large improvements in various metrics. Specifically, the PSNR metric shows an increase from 23.81 to 35.15. Similarly, the SSIM metric demonstrates progress, rising from 0.942 to 0.987, and the LPIPS metric indicates enhancement from 0.018 to 0.001. It is, however, important to keep in mind that the metrics are more strict towards larger triggers in comparison to smaller triggers.

6.3. Ablation analysis

In this section, we analyze our attack across diverse settings. Our examination encompasses various viewpoints on the attack's performance. Initially, we turn our focus to activation values and how our attack influences them. This is fundamental to understanding the phenomenon of robust generalization when training with lower intensities than those used during subsequent testing. Conversely, it elucidates why this trend does not hold when the order is reversed. Furthermore, we investigate how our attack impacts the distances between benign and poisoned updates. Finally, to establish the robustness of our attack. We extend our evaluation to include an alternative trigger, the square trigger to further affirm its efficacy.

6.3.1. Explanation via the activation values



(a) $\delta = 80, \Delta = 10$ **(b)** $\delta = 10, \Delta = 80$

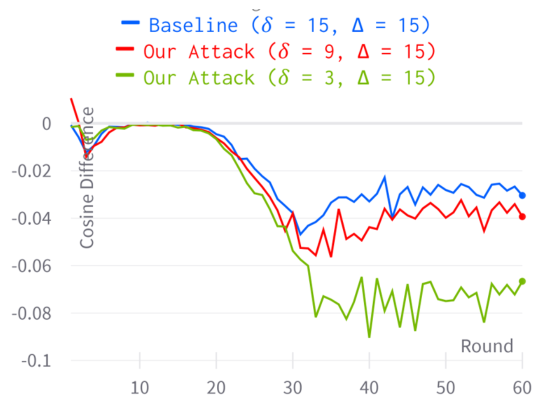
Figure 6.12: Activation values on a poisoned batch in CIFAR-10

In this section, we discuss how using different trigger intensities for training and testing influences the activation values. This explanation serves as the empirical proof of our theory. In figure 6.12 we can see histograms of the activation values in the linear layer of the Resnet18 model. The model is in both cases trained using the CIFAR-10 dataset. The activation values are taken from a small 256-sample batch that contained 256 poisoned values. The horizontal axis represents the activation values, while the vertical axis displays their respective frequencies.

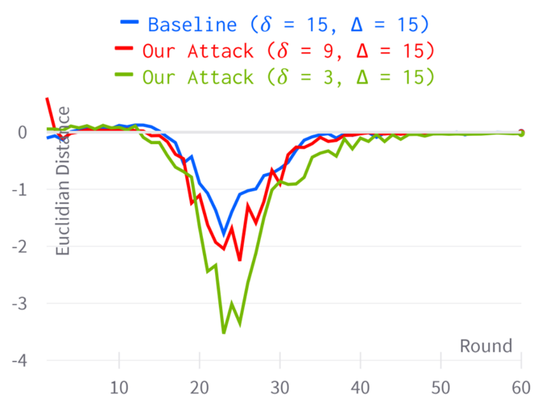
The histogram in figure 6.12a shows what happens when the attacker trains with a strong trigger intensity and subsequently tests with a stronger trigger intensity. Observed is the peak in the distribution of activation values that are near 0. It becomes clear that most activation values in the linear layer are relatively small and barely manage to exceed the value of 20 or -10. In figure 6.12b we can see the result of an attack where the attacker trains with a low trigger intensity and then tests on a higher intensity. This histogram shows a different distribution. It can be seen that the distribution has been split up into two sections. One larger section on the left contains the lower values. These values seem to be more on the negative side in comparison to the former attack. On the right side, we see that there is a small section of neurons that has a high activation value of 40-60. This is a clear distinction from the former attack that did not have this separation on activation values.

The results confirm the theory that a small number of neurons is extremely influential in triggering the backdoor behavior. This becomes evident by the higher activation values that can seen in the distribution of figure 6.12b but are not apparent in figure 6.12a. This small group of neurons is trained in recognizing the backdoor however, due to the increased intensity of the trigger, become more excited 'excited' from seeing and thus are more likely to give their likelihood to the backdoored class.

6.3.2. Effect on distances



(a) Cosine Distance



(b) Euclidian Distance

Figure 6.13: Cosine and Euclidian Distance between the average of the benign runs and the malicious run on different attack intensities on the CIFAR-10 dataset

In this section, we dive into the effect of training on varying trigger intensities. Figure 6.13 offers valuable insights into the results of training models under various training intensities in terms of the Euclidean and cosine distances. These distance metrics serve as indicators of model abnormality and are important in the context of clustering defenses. Our findings reveal that as training intensity decreases, the abnormality of our provided updates in comparison to benign updates increases. This effect is particularly pronounced at the lowest training intensities, where the influence on the distance metrics is most evident.

Figure 6.13 indicates the difference in score between the average of the benign updates and the average of the malicious updates where the malicious average is subtracted from the benign average. Meaning that a negative value shows that the malicious update is larger in score than the benign update. The x-axis shows the rounds and the y-axis shows the difference in the scored metric. Both graphs show the results of the CIFAR-10 dataset. 6.13a highlights the difference between the benign and malicious updates using the cosine distance. It reveals that the malicious updates diverge more and more from the benign updates up until round 30. From that point on, they keep a somewhat consistent score. It becomes clear that the lower the training intensity was, the more deviant the malicious update becomes. Training on a trigger intensity of 15 keeps the difference in cosine distance around -0.03. Training on a trigger intensity of 3 increases this distance significantly by scoring -0.08 consistently after round 30. Analysis of Figure 6.13b reveals that the distance between benign and malicious updates initially increases up to approximately round 25. Subsequently, the malicious distance undergoes a decline, ultimately stabilizing at a consistent score of 0, signifying the absence of significant Euclidean distance between the updates. All three runs show similar progression patterns. However, it's notable that lower training intensities result in a larger maximum reached distance. For instance, the run with a training intensity of 15 attains a distance of 2, while the run with a training intensity of 3 achieves a distance of 3.3.

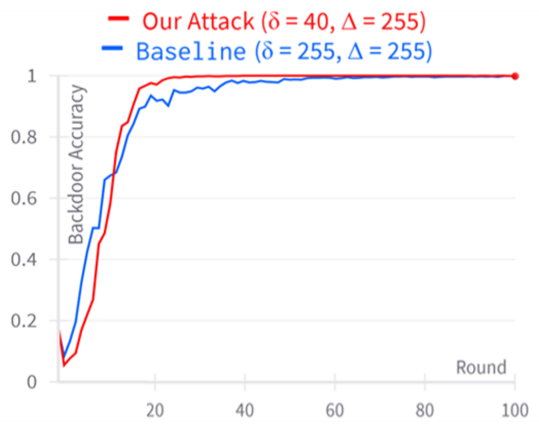
The underlying reason behind this phenomenon lies in the interplay between trigger intensity and the model's behavior. When the model is trained on a trigger with insufficient strength to exert an effective influence, it attempts to map the features of the base class to those of the target class. As a result, the update generated becomes increasingly malicious when compared to a strong trigger, where the model aims to map the features of the trigger to the target class. Consequently, should be cautious to find a balance between the stealthiness of their backdoor after aggregation and the abnormality of the update. Increasing the training intensity mitigates the update's abnormality, making it less conspicuous to clustering defenses. However, this trade-off comes at a cost—it limits the potential for deploying an imperceptible backdoor. Achieving a high BA for a testing intensity when the model has been trained with a weaker training intensity becomes challenging, as elaborated in Section 3.

6.3.3. Results using different triggers

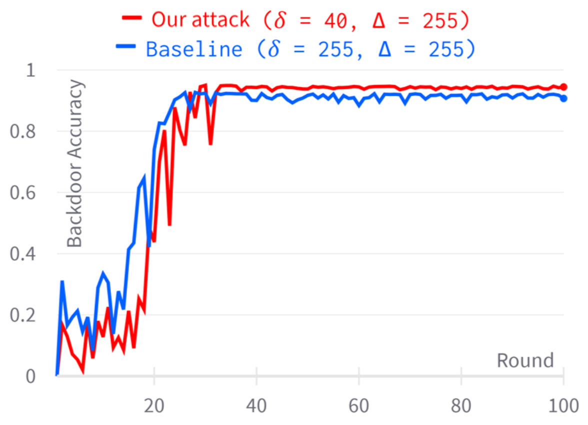
Our investigation delved into the performance of our backdoor attack, this time employing square triggers, across the CIFAR-10 and FMNIST datasets. This variation allowed us to explore the adaptability of our attack strategy under different trigger configurations and its ability to optimize attack performance.

In the FMNIST dataset, our attack strategy showcased its worth by outperforming the baseline in the early rounds. Between rounds 20 and 40, an increase in BA of 10% is observed. This shows again the adaptability and effectiveness of our attack for all triggers as it demonstrates its ability to excel under a different trigger. However, in the later rounds, we observed that both attacks reached their maximum BA score of 100%. Shifting our focus to the CIFAR-10 dataset, we encountered a similar pattern as observed with the SIG pattern. Our attack strategy slightly outperformed the baseline in terms of BA. This consistency in performance across different triggers confirms the robustness and versatility of our attack strategy. Under diverse conditions and trigger configurations, it thrives. Achieving results that consistently challenge or surpass the baseline.

These findings showed that our attack works under different triggers too and further confirms our theory. However, the choice of trigger is an important consideration as it significantly influences the attack's success rate and its dynamics throughout the training process.



(a) FMNIST



(b) CIFAR-10

Figure 6.14: Attack performance under square trigger

7 Limitations and future work

7.1. Limitations

In this chapter, we go over the limitations of our proposed methodology and outline the possible roads for future research directions. Three different domains are used to classify these limitations: First, we discuss constraints on the datasets and models utilized in this thesis. Subsequently, we address limitations regarding the evaluation methodology. This has a particular emphasis on limitations associated with the assessment approach utilized in this study. This organized list of restrictions gives a thorough rundown of the difficulties that come with using our method and gives insightful information about possible roads for improvement and further research. The final section lists the attack strategy's challenges, showing the inherent limits of the suggested attack plan.

7.1.1. Dataset and models

The datasets used in this thesis have several limitations that should be taken into account. The distribution of data used by local participants primarily follows an IID pattern. That means that each local agent receives data from the same distribution. It's crucial to recognize that the subtleties of FL systems in the real world are not fully captured by this simplistic method. Local datasets in FL scenarios are typically non-IID in practice, which means that the properties, patterns, and representation of the data on different local devices or participants may vary greatly from one another. Geographic differences, device kinds, and user habits are some of the elements that contribute to the diversity found in local datasets. Because of these variations and the particular data they are trained on, local models emerge that are intrinsically different from one another. Therefore, even though it is a research-useful simplification, the assumption of a uniform data distribution departs from the intrinsic complexity of FL systems. To overcome this constraint, a dataset based on an actual FL scenario must be produced.

Additionally, due to practical constraints, our experiments primarily focused on FMNIST and CIFAR-10. These are the two most commonly used datasets in state-of-the-art research. Whilst both datasets do prove informative, they do not provide a comprehensive view of the broader landscape of adversarial attacks in FL. Future research should consider a wider range of datasets. As mentioned in the previous point, should be distributed in a federated manner but also be larger and more diverse. This is needed to fully understand and prove our findings and gain a more holistic understanding of the challenges posed by such attacks in FL.

Our datasets had size limitations that deserve attention. For example, CIFAR-10 image samples are relatively small at 24x24 pixels, while Tiny ImageNet image samples are 64x64 pixels. These constraints raise questions about how our attack behaves when applied to high-resolution datasets. Exploring the attack's properties in such contexts is an interesting point for future research, offering insights into its adaptability and effectiveness in crafting imperceptible backdoors in more visually complex environments.

In addition, our experiments were conducted using two widely adopted models for assessing backdoor attacks. However, it is crucial to expand our experiments to encompass a broader range of neural network models. This extension is significant because different neural network models may exhibit varying susceptibilities to backdoor attacks, potentially influencing the outcomes and providing a better understanding of the attack strategy's effectiveness across diverse model architectures.

7.1.2. Evaluation

Our evaluation encompasses a maximum of 10 participants within the FL system. While this evaluation is in line with many other studies, it is important to recognize that this setting still falls short of fully representing the scale of a typical FL system. In practical scenarios, FL systems often involve thousands or even tens of thousands of participants participating in each round of training. Acknowledging this difference is crucial, as it underscores the need for future research to explore the dynamics and vulnerabilities of FL at the higher agent counts characteristic of real-world deployments.

Our evaluation focused on a fixed-frequency attack scenario, which may not accurately reflect the dynamics of all FL setups. In certain situations, a one-time attack scenario may be more applicable. Here, the attacker has only one chance to participate in an attack round. This approach can be seen as more realistic and challenging, as it doesn't assume the attacker's continuous participation in every round.

Criticism can also be directed at our use of the visibility scores as an evaluation metric. Although our research explored multiple visibility metrics, each had certain limitations. Whilst coming closest to meeting our evaluation needs, our metrics fall short of capturing the full complexity of human visual perception. Notably, it was revealed during our research all considered metrics tended to penalize large triggers disproportionately compared to smaller ones. We found that especially true for the LPIPS and the SSIM metrics. The PSNR metric showed less bias when comparing two different triggers with one another, yet lacked in getting the complexity of human vision right when comparing the same trigger with different trigger intensities. On top, our trigger visibility analysis lacked depth for the FMNIST dataset. The two more sophisticated evaluation metrics, LPIPS and SSIM, were not compatible with black-and-white images. For that reason, we propose that a new metric is needed in order to evaluate the visibility of backdoor triggers.

7.1.3. Our technique

Although our attack demonstrates its potential, it's important to note that an attacker can enhance its effectiveness by complementing it with other data or model poisoning techniques. Our attack hasn't been assessed in combination with other SOTA attacks. Consequently, we may not have fully realized its capabilities. The reason for this came from our intention to concentrate solely on our attack and its attributes. We present it as an additional tool for researchers to investigate, offering a potential when combined with other techniques.

Secondly, our technique was not evaluated on all possible defenses that are out there. There were too many defenses to evaluate and for that reason, we made a selection of defenses we wanted our attack to be evaluated on. We tried to include some of the more sophisticated defenses but as one can imagine, they do require more work to implement. We attempted to consider as many as possible defenses for each category they belong in (e.g. clustering etc.). We made a tradeoff between difficulty and category for each defense. However, we acknowledge that this selection might have excluded defenses that might have been extremely effective against our attack.

7.2. Future work

There are numerous ways for future research that can build upon and improve the results of this study. One way of exploration involves the integration of our attack strategy with other SOTA attacks. Combining our attack with existing approaches presents an opportunity to investigate the synergistic or reinforcing effects that arise from such combinations. Exploring how our attack interacts with other attacks can provide valuable insights into the potential for creating more sophisticated and potent adversarial strategies within FL environments.

Future studies should also focus on correcting the increased anomalies in distance measures that occur when our attack's training intensity drastically drops. It is crucial to look for ways to lessen or eliminate these anomalies. Determining efficient methods for maintaining the attack's behavior at low training intensities will help us comprehend its shortcomings and identify areas for future development.

On the defensive front, there is a pressing need for the development of robust and effective defenses against backdoor attacks in FL after aggregation. Our research demonstrated that manipulating trigger intensities can improve existing attacks, potentially enabling the creation of invisible backdoors. Therefore, investigating possible solutions aiming at improving post-aggregation defense systems is essential. Previous research has shown that existing pre-aggregation defenses may not be able to identify backdoors in non-IID environments. As a result, there is a need for defenses that can remove backdoors after they are integrated into the global model. These protections ought to be built to function well in FL circumstances that occur in the real world, where data security and privacy are critical.

Additionally, future research could explore innovative ways of quantifying the visibility of backdoor triggers. For instance, methods involving real individuals to assess trigger visibility. However, it's important to acknowledge the potential introduction of personal biases in such evaluations. Alternatively, researchers could develop novel metrics specifically designed to measure the visibility of backdoor triggers, providing more objective and standardized assessments. These metrics could take into account a range of factors, including human perception and potential biases, to offer a comprehensive understanding of trigger visibility. They should also improve on some of the problems encountered in this thesis notably improving on the ability to quantify the trigger visibility when the sizes in triggers differ. We found that larger triggers were considerably more strictly punished than smaller ones.

8 Conclusion

In this study, we addressed the need for exploring trigger intensity in backdoor attacks within the context of FL. By challenging the conventional requirement of training and testing on the same trigger intensity, we demonstrated the efficacy of training on a weak trigger and testing on a stronger trigger. This novel approach could significantly enhance the performance and robustness of backdoor attacks, making them more resilient to different environmental conditions. The ability to customize trigger visibility empowers attackers to tailor their poisoned samples and create a balance between attack performance and stealthiness.

Our method complements existing state-of-the-art attacks, enabling the construction of stronger and more stealthy backdoor attacks. By shedding light on the importance of trigger intensity, we contribute to the growing body of research on vulnerabilities in FL systems against backdoor attacks.

Some challenges remain despite our contributions, such as the increasing anomaly that goes hand in hand with lower trigger intensities. Future work should focus on exploring more advanced triggers, but also countermeasures that could help to detect backdoors after aggregation.

In summary, our research advances the understanding of backdoor attacks in FL and provides valuable insights into trigger intensity as a critical factor for attack performance and stealthiness. By exploring new methods for enhancing attack customization and resilience, we contribute to the ongoing efforts to find vulnerabilities in FL systems.

In the section below we give answers to the research questions:

How does varying the pixel intensity of backdoor triggers affect the success rate of backdoor attacks in FL systems?
Our practical research findings suggest that when we train models with a low trigger intensity and then test them with a higher intensity, we observe improved results compared to models trained and tested at a higher intensity. It's worth noting, though, that the training intensity shouldn't be excessively low, as this might prevent the model from learning the backdoor task effectively. We've found that there's a benefit for attackers to train with a slightly lower trigger intensity than what they plan to test on, as it appears to slightly boost the success rate.

Do varying the pixel intensities of backdoor triggers allow invisible triggers to be inserted?
Our discovery indicates a significant reduction in the visibility of backdoor triggers in image samples while maintaining backdoor accuracy. However, this ability is somewhat influenced by the attack's strength, as a stronger attack can achieve a greater reduction compared to a weaker one. It's important to highlight the challenge of accurately measuring the invisibility of backdoor triggers. Although we employed PSNR, SSIM, and LPIPS scores for this purpose, we acknowledge their limitations. As a result, we suggest the necessity for a new and improved methodology or metric to measure the invisibility of backdoor triggers. Consequently, determining whether an image can be classified as truly invisible remains a complex task.

What are the implications of using different trigger intensity levels on the robustness of FL defenses against backdoor attacks?
Based on our empirical using four different defense techniques, we found that most defenses are not capable of detecting the attack nor able to reduce the attack performance. Although we found that clustering attacks work well under IID distribution, we noticed that this advantage is lost when presented with a more realistic non-IID scenario. However, the defense techniques did show some results. For instance, the attack loses some of its effectiveness by needing to present a stronger trigger in order to get good results. Thus, disabling the attacker to insert invisible backdoor triggers takes away some power of the attack. Yet, in nearly all cases the baseline attack outperformed.

What are the drawbacks of this and how can they be employed to provide defensive measures?
In our results, we showed that using different triggers leaves its traces. It can be seen that the Euclidian distance and the cosine distance increase when presented with low training intensity. As a result, clustering defenses could be used to detect this attack. On the other hand, clustering defenses do not perform well under non-IID circumstances which is accepted as the default for FL in a realistic setting.

Can the pixel intensity of triggers be used to reinforce backdoor attacks?
In our research, we break the requirement of training and testing on the same trigger intensity. We first found that training with a high trigger intensity and then testing with a lower intensity doesn't work well, mainly because of how neural networks operate. On the flip side, training with a lower trigger intensity and testing with a higher one seems to help the model understand the backdoor better. It even makes the attack a bit more effective compared to using the same trigger intensity for both training and testing. What's interesting is that this method opens up possibilities for backdoor attacks with almost invisible triggers. This finding might be valuable for future research, combined with other tactics to create stronger and more challenging-to-detect attacks.

References

[1] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3.

[2] Sebastien Andreina et al. “Baffle: Backdoor detection via feedback-based federated learning”. In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2021, pp. 852–863.

[3] E. Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. “Differential privacy has disparate impact on model accuracy”. In: *Advances in neural information processing systems* 32 (2019).

[4] E. Bagdasaryan et al. “How to backdoor federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938–2948.

[5] Mauro Barni, Kassem Kallas, and Benedetta Tondi. “A new backdoor attack in cnns by training set corruption without label poisoning”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 101–105.

[6] Gilad Baruch, Moran Baruch, and Yoav Goldberg. “A little is enough: Circumventing defenses for distributed learning”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[7] J. Bernstein et al. “signSGD: Compressed optimisation for non-convex problems”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 560–569.

[8] Arjun Nitin Bhagoji et al. “Analyzing federated learning through an adversarial lens”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 634–643.

[9] P. Blanchard et al. “Machine learning with adversaries: Byzantine tolerant gradient descent”. In: *Advances in neural information processing systems* 30 (2017).

[10] K. Bonawitz et al. “Towards federated learning at scale: System design”. In: *Proceedings of machine learning and systems* 1 (2019), pp. 374–388.

[11] Léon Bottou. “Stochastic gradient descent tricks”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 421–436.

[12] Di Cao et al. “Understanding distributed poisoning attack in federated learning”. In: *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE. 2019, pp. 233–239.

[13] S. Clark. *This new Adobe audio tool could make your home office sound like a studio*. 2022. URL: `https://www.techradar.com/news/this-new-adobe-audio-tool-could-make-your-home-office-sound-like-a-studio` (visited on 05/10/2023).

[14] M.F. Criado et al. “Non-IID data and Continual Learning processes in Federated Learning: A long road ahead”. In: *Information Fusion* 88 (2022), pp. 263–280.

[15] Marcos F Criado et al. “Non-IID data and Continual Learning processes in Federated Learning: A long road ahead”. In: *Information Fusion* 88 (2022), pp. 263–280.

[16] P. Debleena et al. “Artificial intelligence in drug discovery and development”. In: *Drug discovery today* 26.1 (2021), p. 80.

[17] Enmao Diao, Jie Ding, and Vahid Tarokh. “Heterofl: Computation and communication efficient federated learning for heterogeneous clients”. In: *arXiv preprint arXiv:2010.01264* (2020).

[18] C. Fung, Chris JM Yoon, and Ivan Beschastnikh. “Mitigating sybils in federated learning poisoning”. In: *arXiv preprint arXiv:1808.04866* (2018).

[19] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. “Mitigating sybils in federated learning poisoning”. In: *arXiv preprint arXiv:1808.04866* (2018).

[20] F. Garamvolgyi. *Why us women are deleting their period tracking apps*. 2022. URL: `www.theguardian.com/world/2022/jun/28/why-us-woman-are-deleting-their-period-tracking-apps` (visited on 05/12/2023).

[21] X. Gong et al. “Coordinated Backdoor Attacks against Federated Learning with Model-Dependent Triggers”. In: *IEEE network* 36.1 (2022), pp. 84–90.

[22] Tianyu Gu et al. “Badnets: Evaluating backdooring attacks on deep neural networks”. In: *IEEE Access* 7 (2019), pp. 47230–47244.

[23] Erkam Guresen and Gulgun Kayakutlu. “Definition of artificial neural networks with comparison to other networks”. In: *Procedia Computer Science* 3 (2011), pp. 426–433.

[24] T. Han et al. “An efficient deep learning framework for intelligent energy management in IoT networks”. In: *IEEE Internet of Things Journal* 8.5 (2020), pp. 3170–3179.

[25] A. Hard et al. “Federated learning for mobile keyboard prediction”. In: *arXiv preprint arXiv:1811.03604* (2018).

[26] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Alain Hore and Djemel Ziou. “Image quality metrics: PSNR vs. SSIM”. In: *2010 20th international conference on pattern recognition*. IEEE. 2010, pp. 2366–2369.

[28] Samuel Horvath et al. “Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12876–12889.

[29] J. Isaak and Mina J Hanna. “User data privacy: Facebook, Cambridge Analytica, and privacy protection”. In: *Computer* 51.8 (2018), pp. 56–59.

[30] P. Kairouz et al. “Advances and open problems in federated learning”. In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210.

[31] A. Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[32] Y. Le and Xuan Yang. “Tiny imagenet visual recognition challenge”. In: *CS 231N* 7.7 (2015), p. 3.

[33] T. Li et al. “Federated learning: Challenges, methods, and future directions”. In: *IEEE signal processing magazine* 37.3 (2020), pp. 50–60.

[34] T. Li et al. “Federated optimization in heterogeneous networks”. In: *Proceedings of Machine learning and systems* 2 (2020), pp. 429–450.

[35] Bingyan Liu et al. “Recent Advances on Federated Learning: A Systematic Survey”. In: *arXiv preprint arXiv:2301.01299* (2023).

[36] T. Liu, Xueyang Hu, and Tao Shu. “Facilitating Early-Stage Backdoor Attacks in Federated Learning with Whole Population Distribution Inference”. In: *IEEE Internet of Things Journal* (2023).

[37] Y. Liu et al. “Reflection backdoor: A natural backdoor attack on deep neural networks”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer. 2020, pp. 182–199.

[38] Yang Liu, Zhihao Yi, and Tianjian Chen. “Backdoor attacks and defenses in feature-partitioned collaborative learning”. In: *arXiv preprint arXiv:2007.03608* (2020).

[39] Yang Liu et al. “Batch label inference and replacement attacks in black-boxed vertical federated learning”. In: *arXiv preprint arXiv:2112.05409* (2021).

[40] B. McMahan et al. “Communication-efficient learning of deep networks from decentralized data”. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[41] Brendan McMahan and Daniel Ramage. “Federated learning: Collaborative machine learning without centralized training data”. In: *Google Research Blog*. 2017. URL: `https://blog.research.google/2017/04/federated-learning-collaborative.html` (visited on 09/21/2023).

[42] Bouacida Nader et al. “Adaptive federated dropout: Improving communication efficiency and generalization for federated learning”. In: *Proceedings of the IEEE Conference on Computer Communications Workshops*. 2021, pp. 1–6.

[43] T.D. Nguyen et al. “{FLAME}: Taming backdoors in federated learning”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 1415–1432.

[44] Thuy Dung Nguyen et al. “Backdoor Attacks and Defenses in Federated Learning: Survey, Challenges and Future Research Directions”. In: *arXiv preprint arXiv:2303.02213* (2023).

[45] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).

[46] Ashwinee Panda et al. “Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 7587–7624.

[47] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[48] A. Radford et al. *CLIP: Connecting text and images*. 2021. URL: `openai.com/research/clip` (visited on 05/10/2023).

[49] S. Reddi et al. “Adaptive federated optimization”. In: *arXiv preprint arXiv:2003.00295* (2020).

[50] Gabriel Prieto Renieblas et al. “Structural similarity index family for image quality assessment in radiological images”. In: *Journal of medical imaging* 4.3 (2017), pp. 035501–035501.

[51] P. Rieger et al. “Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection”. In: *arXiv preprint arXiv:2201.00763* (2022).

[52] M. Ozdayi Safa, Murat Kantarcioglu, and Yulia R Gel. “Defending against backdoors in federated learning with robust learning rate”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 9268–9276.

[53] Felix Sattler et al. “On the byzantine robustness of clustered federated learning”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8861–8865.

[54] V. Shejwalkar et al. “Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning”. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 1354–1371.

[55] N. Shoham et al. “Overcoming forgetting in federated learning on non-iid data”. In: *arXiv preprint arXiv:1910.07796* (2019).

[56] Sidak Pal Singh and Martin Jaggi. “Model fusion via optimal transport”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 22045–22055.

[57] Virginia Smith et al. “Federated multi-task learning”. In: *Advances in neural information processing systems* 30 (2017).

[58] Jingwei Sun et al. “Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12613–12624.

[59] Z. Sun et al. “Can you really backdoor federated learning?” In: *arXiv preprint arXiv:1911.07963* (2019).

[60] Iuliana Tabian, Hailing Fu, and Zahra Sharif Khodaei. “A convolutional neural network for impact detection and characterization of complex composite structures”. In: *Sensors* 19.22 (2019), p. 4933.

[61] Vale Tolpegin et al. “Data poisoning attacks against federated learning systems”. In: *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer. 2020, pp. 480–501.

[62] B. Turovsky. *Found in translation: More accurate, fluent sentences in Google Translate*. 2016. URL: `blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate` (visited on 05/10/2023).

[63] Anish Singh Walia. “Activation functions and it’s types-Which is better?” In: *Towards Data Science* 29 (2017).

[64] H. Wang et al. “Attack of the tails: Yes, you really can backdoor federated learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16070–16084.

[65] Ning Wang et al. “FLARE: defending federated learning against model poisoning attacks via latent space representations”. In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 2022, pp. 946–958.

[66] Dingzhu Wen, Ki-Jun Jeon, and Kaibin Huang. “Federated dropout—A simple approach for enabling federated learning on resource constrained devices”. In: *IEEE wireless communications letters* 11.5 (2022), pp. 923–927.

[67] Chen Wu, Sencun Zhu, and Prasenjit Mitra. “Federated unlearning with knowledge distillation”. In: *arXiv preprint arXiv:2201.09441* (2022).

[68] Chen Wu et al. “Mitigating backdoor attacks in federated learning”. In: *arXiv preprint arXiv:2011.01767* (2020).

[69] Yuncheng Wu et al. “Privacy preserving vertical federated learning for tree-based models”. In: *arXiv preprint arXiv:2008.06170* (2020).

[70] H. Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[71] C. Xie et al. “Dba: Distributed backdoor attacks against federated learning”. In: *International conference on learning representations*. 2020.

[72] Chulin Xie et al. “Crfl: Certifiably robust federated learning against backdoor attacks”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 11372–11382.

[73] Q. Yang et al. “Federated machine learning: Concept and applications”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–19.

[74] D. Yin et al. “Byzantine-robust distributed learning: Towards optimal statistical rates”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5650–5659.

[75] K. Yoo and Nojun Kwak. “Backdoor Attacks in Federated Learning by Rare Embeddings and Gradient Ensembling”. In: *arXiv preprint arXiv:2204.14017* (2022).

[76] M. Yurochkin et al. “Bayesian nonparametric federated learning of neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 7252–7261.

[77] Jiale Zhang et al. “PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems”. In: *IEEE Internet of Things Journal* 8.5 (2020), pp. 3310–3322.

[78] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[79] Xuemei Zhang and David H Brainard. “Estimation of saturated pixel values in digital color imaging”. In: *JOSA A* 21.12 (2004), pp. 2301–2310.

[80] Z. Zhang et al. “Neurotoxin: Durable backdoors in federated learning”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 26429–26446.

[81] Z. Zhao et al. “DEFEAT: Deep Hidden Feature Backdoor Attacks by Imperceptible Perturbation and Latent Representation Constraints”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15213–15222.

[82] Runkai Zheng et al. “Pre-activation Distributions Expose Backdoor Neurons”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18667–18680.

[83] Tianyuan Zou et al. “Defending batch-level label inference and replacement attacks in vertical federated learning”. In: *IEEE Transactions on Big Data* (2022).

A

Source Code

In this section, the most important sections of code are displayed.

The loss calculation function

```python
def loss_calculator(args, poison_loss, val_loss, compare_img_loader, compare_pos_img_loader):
    if args.metric == 'psnr':
        visual_loss = 1 - (trigger_visibility(args, compare_img_loader,
            compare_pos_img_loader) / 80) * 3
    elif args.metric == 'ssim':
        visual_loss = ((1 - trigger_visibility(args, compare_img_loader,
            compare_pos_img_loader)) * 8)
    else:
        visual_loss = trigger_visibility(args, compare_img_loader, compare_pos_img_loader) *
            25
    print(f'poison_loss: {poison_loss} + val_loss: {val_loss} + visual loss: {visual_loss}')
    print(f'total loss: {poison_loss + (val_loss / 2) + visual_loss}')
    return poison_loss + (val_loss / 2) + visual_loss
```

The CNN model used for the FMNIST dataset

```python
class CNN_MNIST(nn.Module):
    def __init__(self):
        super(CNN_MNIST, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=(3,3))
        self.conv2 = nn.Conv2d(32, 64, kernel_size=(3,3))
        self.max_pool = nn.MaxPool2d(kernel_size=(2, 2))
        self.drop1 = nn.Dropout2d(p=0.5)
        self.fc1 = nn.Linear(9216, 128)
        self.drop2 = nn.Dropout2d(p=0.5)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.max_pool(x)
        x = x.view(-1, x.shape[1]*x.shape[2]*x.shape[3])
        x = self.drop1(x)
        x = F.relu(self.fc1(x))
        x = self.drop2(x)
        x = self.fc2(x)
        return x
```

Trainings function

```python
def local_train_normal_attack(self, global_model, criterion, attack):
        """ Do a local training over the received global model, return the update """
        initial_global_model_params = parameters_to_vector(global_model.parameters()).detach
            ()
        global_model.train()
        optimizer = torch.optim.SGD(global_model.parameters(), lr=self.args.client_lr,
            momentum=self.args.client_moment)

        #get the poisoned dataset for the attacker
        if (self.id < self.args.num_corrupt and attack and self.args.attack in ['normal', '
            optimize']) or (self.args.attack == 'dba' and self.id % self.args.num_corrupt ==
            0 and attack and self.id < self.args.num_corrupt):
            dataloader = self.poison_loader
        else:
        #use the benign datasetset for  malicious agent in non-attack round
            dataloader = self.train_loader

        for _ in range(self.args.local_ep):
            for _, (inputs, labels) in enumerate(dataloader):
                optimizer.zero_grad()
                inputs, labels = inputs.to(device=self.args.device, non_blocking=True),
                    labels.to(device=self.args.device, non_blocking=True)
                outputs = global_model(inputs)
                minibatch_loss = criterion(outputs, labels)
                minibatch_loss.backward()

                # to prevent exploding gradients
                nn.utils.clip_grad_norm_(global_model.parameters(), 10)
                optimizer.step()

        with torch.no_grad():
            update = parameters_to_vector(global_model.parameters()).double() -
                initial_global_model_params
            return update
```

Pattern injection function

```python
def add_pattern_bd(x, trainset, dataset='cifar10', pattern_type='square', agent_idx=-1,
    attack_type='normal', delta_attack=None, delta_val=None, frequency=None):
    """
    adds a trojan pattern to the image
    """
    x = np.array(x.squeeze())
    if agent_idx != -1 or trainset == 1:
        delta = delta_attack
    else:
        delta = delta_val

    # if cifar is selected, we're doing a distributed backdoor attack (i.e., portions of
        trojan pattern is split between agents, only works for plus)
    if dataset in ['cifar10', 'cifar100']:
        if pattern_type == 'sig':
                f = frequency
                x = np.float32(x)
                pattern = np.zeros_like(x)
                m = pattern.shape[1]
                for i in range(x.shape[0]):
                    for j in range(x.shape[1]):
                        for k in range(x.shape[2]):
                            pattern[i, j] = delta * np.sin(2 * np.pi * j * f / m)

                x = x + pattern
                x = np.where(x > 255, 255, x)
                x = np.where(x < 0, 0, x)
                return x

            elif pattern_type == 'square':
                x = np.float32(x)
                pattern = np.zeros_like(x)
                for i in range(4, 4 + 5):
                    for j in range(4, 4 + 5):
                        pattern[i, j] = -delta

                x = x + pattern
                x = np.where(x > 255, 255, x)
                x = np.where(x < 0, 0, x)
                return x
```

Dirichtlet distribution function

```python
def distribute_dirichlet(dataset, args, num_classes):
    data_idxs = {}

    shard_id = [[] for _ in range(args.num_agents)]
    for k in range(num_classes):
        idx_k = np.where(dataset.targets == k)[0]
        np.random.shuffle(idx_k)

        dist = np.random.dirichlet(np.repeat(args.alpha, args.num_agents))
        dist = dist / dist.sum()
        dist = (np.cumsum(dist) * len(idx_k)).astype(int)[:-1]

        shard_id = [j + i.tolist() for j, i in zip(shard_id, np.split(idx_k, dist))]

    for j in range(args.num_agents):
        np.random.shuffle(shard_id[j])
        data_idxs[j] = shard_id[j]

    return data_idxs
```

B Image samples

In this section, we aim to provide a visual representation of each trigger by showcasing a selection of samples. This allows for a more intuitive understanding of how the samples appear at various trigger intensities. Specifically, for the SIG trigger, we present samples ranging from 0 to 100, with intervals of 10. Likewise, for the square trigger, we exhibit samples spanning from 0 to 250, with intervals of 25 for each displayed image. This approach enables a clearer and more insightful perspective on the visual impact of different trigger intensities.

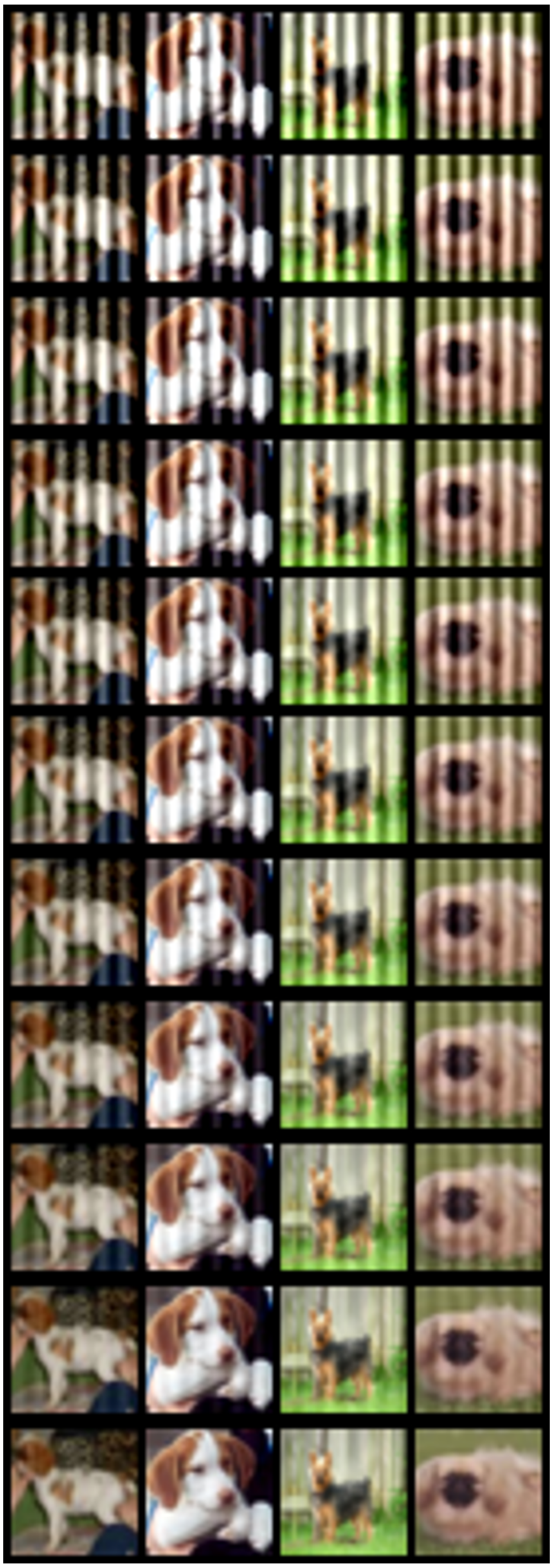


Figure B.1: Visualization of poisoned image samples with the SIG trigger. Bottom is a clean image and top is the maximum trigger intensity used in this thesis for the SIG trigger

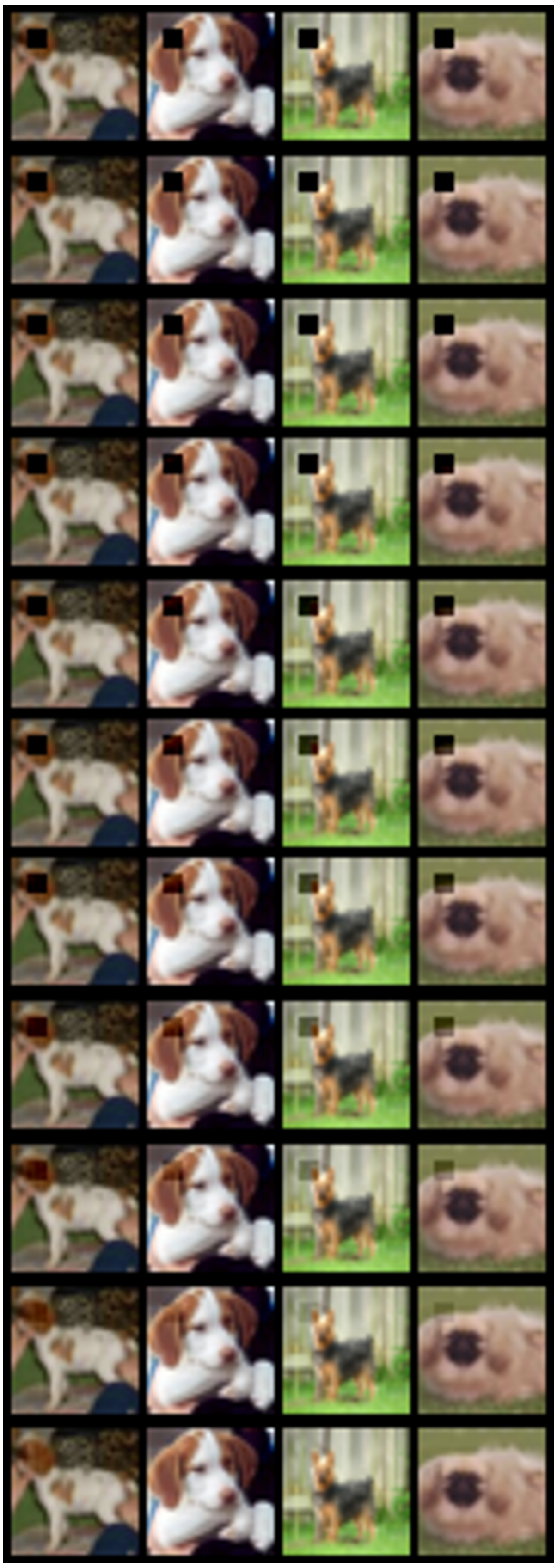


Figure B.2: Visualization of poisoned image samples with the Square trigger. Bottom is a clean image and top is the maximum trigger intensity used in this thesis for the square trigger