Distributed Swift and Stealthy Backdoor Attack on Federated Learning

Agnideven Palanisamy Sundar[2†], Feng Li[1†], Xukai Zou[2†] and Tianchong Gao[3⋆]

[1]Department of Computer and Information Technology
[2]Department of Computer and Information Science
[3]Department of Cyberspace Security
[†]Indiana University-Purdue University Indianapolis, Indianapolis, IN, USA.
[⋆]Southeast University, Nanjing, Jiangsu, China.
{*fengli,xzou*}*@iupui.edu, agpalan@iu.edu, tgao@seu.edu.cn*

***Abstract*—Federated Learning (FL) provides enhanced privacy over traditional centralized learning; unfortunately, it is also as susceptible to backdoor attacks, just like its centralized counterpart. Conventionally, in data poisoning-based backdoor attacks, all the malicious participants overlay the same single trigger pattern on a subset of their private data during local training. The same trigger is used to induce the backdoor in the otherwise benign global model at inference time. Such single trigger attacks can be detected and removed with relative ease as they undermine the distributed nature of FL. In this work, we focus on building an attack scheme where each batch of malicious clients uses sizably discrete local triggers during local training, with the ability to invoke the attack with a single small inference trigger during the global model testing. The larger size of the trigger pattern ensures prolonged attack longevity even after the termination of the attack. We conduct extensive experiments to show that our approach is far faster, stealthier, and more effective than the centralized trigger approach. The stealthiness of our work is explained using the DeepLIFT visual feature interpretation method.**

I. INTRODUCTION

Federated Learning (FL) is a subdivision of Machine Learning, where the centralized learning approach is foregone to improve client privacy [1]. In the FL approach, the local training is executed in the client-end. The trained local models are then directly aggregated in the global server, eradicating the need for sharing the labeled private data with the server. This enhanced privacy feature, along with the drop in the overall communication overhead, has widened the popularity of FL systems in domains like healthcare, typing prediction, and banking.

Although FL provides far superior privacy protection compared to the centralized approach, it comes with its own set of security issues. Unfortunately, backdoor attacks play a more crucial role in the FL scenario. In data poisoning-based backdoor attacks, the attacker infects the global model so that the model misclassifies in the presence of a trigger while acting benignly in all other cases. During local training, the triggers are superimposed on a small subset of training data records, and their corresponding labels are modified to the attacker selected target labels. Training with both backdoored and original data records makes the model simultaneously learn both the backdoor and main tasks.

Many research works extend some of the centralized defense methods to FL systems. Such defenses usually assume that all the malicious clients use the same trigger pattern. They aim to find the smallest pattern needed to misclassify a data record into a different category [2], [3]. Though many backdoor attacks follow such a single trigger approach, it is not a requirement, given that the FL system is itself distributed. Such defenses fail to perform when there are multiple trigger patterns involved.

Given the inability of the benign participants or the central server to inspect the malicious training data, the attacker has the freedom to alter the size, shape, and position of the triggers. The size of the attack pattern plays a critical factor in the learning process. The model learns larger patterns faster, and the presence of the backdoor is more prolonged even after attack termination. At the same time, using large triggers for invoking the attack during the inference time is inefficient. We have taken these factors into account and incorporated them into our approach. In our attack, the trigger patterns used for training differ from the trigger used for attack inference. Each batch of malicious clients uses various sets of sizeable discrete triggers for local training, which overlap in a single small space, which we utilize as the Inference trigger for invoking the attack during the test time. In addition to improving the attack success rate in fewer rounds, our attack also manages to survive in the global model long after the attack has been terminated.

This paper makes the following contributions.

- We design an attack approach using sizeable discrete triggers for local training while using a smaller inference trigger to invoke the attack.
- We experimentally show that our attack takes effect faster and survives longer in the global model compared to traditional single trigger approaches.
- We algorithmically improve the attack scheme in scenarios where regular communication among malicious clients is feasible.

978-1-6654-5408-7/22/$31.00 ©2022 IEEE

- We depict DeepLIFT-based feature representations to highlight the difference in the stealthiness between our approach and the centralized trigger approach and also test our attack against state-of-the-art defense techniques.

II. BACKGROUND AND THREAT MODEL

Let's look at the high-level overview of the Federated Learning approach and the generic Backdoor Attack on FL.

A. Federated Learning

Federated Learning utilizes Distributed Learning techniques for training Machine Learning models by combining the local models of the participating clients. In many cases, the central server only selects a different subset of participants for each round of the learning process. Based on the instructions from the central server, the selected participants/clients train their models locally and share them with the central server. The central server aggregates the local models to generate the global model, which is then shared with the clients. The clients update their local model based on the current global model, and the process is repeated until the global model converges. On a high level, the model updates in each round can be represented as:

$$G^{t+1} = G^t + \frac{\eta}{M}\sum_{i=1}^{M}(C_i^{t+1} - G^t) \tag{1}$$

In the equation above, G^t is the current global model at iteration t, C_i^{t+1} is the model update of client i, M is the number of clients in the current round, and η is the global model learning rate.

B. Backdoor Attack

Backdoor attacks are intended to mislead the global model into misclassifying any input record into an attacker selected target class by placing a small trigger. The benign model objective is the main task, and the attackers' objective is the backdoor task.

$$\begin{aligned} f(x) &\longrightarrow y \\ f(x+\tau) &\longrightarrow y' \end{aligned} \tag{2}$$

Consider that the global model $f()$, classifies the input record x as label y under benign conditions. But after a successful attack, the presence of a small trigger τ on the input record x, causes the model to misclassify the input as y', an attacker selected target.

C. Threat Model

We follow a threat model similar to existing works [4], [5].
Access Privilege. The attacker does not have access to the dataset or the training process of the benign clients, nor the information about the global model.
Training Privilege. The attacker can modify the malicious client's local training dataset and coordinate with other malicious clients during training time.
Testing Privilege. The attacker cannot modify the model during the inference process but can add trigger patches to the test data records.

III. OUR APPROACH

This section discusses the types and purposes of the triggers used, along with the overview of our attack scheme.

A. The Two Types of Trigger Patterns

1) Singular Inference Trigger.: The inference trigger will be the pattern used during the inference time of the attack to invoke the backdoor. All the other trigger patterns must encompass the inference trigger. The reason for the relatively smaller size of the inference trigger pattern is to ensure stealthy attack execution during inference time. In general, the inference trigger needs to be placed away from the corner of an image to increase the space available for the discrete trigger patterns.

2) Dispersed Discrete Triggers.: Discrete triggers are used by the malicious clients for local training. Starting with the inference trigger as the base, the discrete triggers are developed by expanding the pixels adjacent to the inference trigger in a specific selected direction. No two discrete triggers should get overlapped except for along the inference trigger. The size of the discrete trigger can be much larger than the inference trigger since it won't be visible to other non-malicious entities under any circumstance.

B. Attack Scheme

We use the distributed nature of FL to build an attack that would have a quicker impact on the global model while remaining stealthy during inference. The trigger patterns are distributed among the clients based on two strategies. If the communication among the malicious clients is reliable, all the malicious clients receive all the trigger patterns. Each client has to select the appropriate trigger for each round. How the appropriate discrete trigger is selected is explained in the experiments section. On the other hand, if the communication among clients in each round is not feasible, then the malicious clients are divided into multiple batches. Each batch receives one discrete trigger and the common inference trigger. Each time a malicious client is selected for the FL process, the model trained with the discrete trigger is submitted to the central server for aggregation.

Fig. 1 shows an overview of our attack scheme. In our toy example, the backdoor task is to misclassify 'airplane' as 'horse'. Each batch of malicious clients uses different triggers for local training. The discrete trigger patterns are superimposed on a subset of airplane images and are relabelled as 'horse'. The relabelled dataset and other benign data records are used to train a local model. After local training, the backdoor models are sent to the central server for aggregation.

In the figure, we notice that all the discrete triggers overlap in the same small square region. This region forms the inference trigger. It is essential to ensure no overlap between any of the discrete trigger pairs in any area other than the inference trigger region. After the aggregation, the global model gets infected with the backdoor. This implies that the global model has learned both the backdoor task and the main task. The test image must be superimposed with the inference trigger pattern

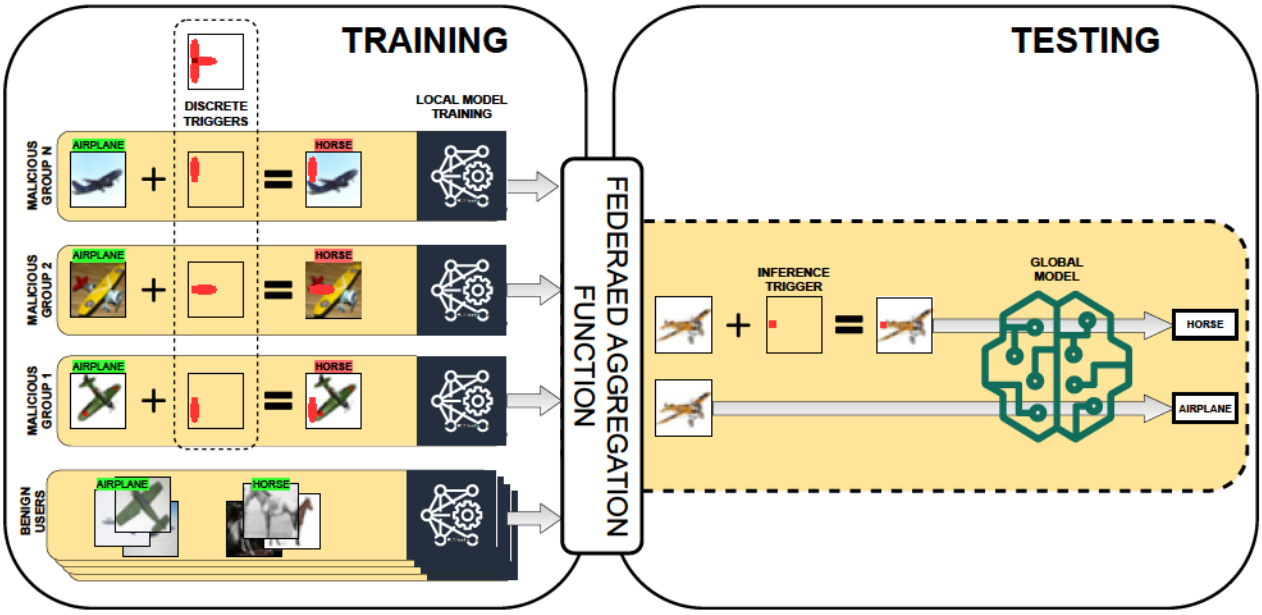


Fig. 1: Overview of the distributed trigger backdoor attack approach.

to invoke the backdoor. In the testing phase of Fig. 1, we can notice that the global model misclassifies the input 'airplane' image as a 'horse' when the image is superimposed with the inference trigger. But, the global model correctly predicts the image as an 'airplane' in the absence of the trigger.

C. Understanding Discrete and Inference Trigger Contribution

The attacker's goal is to create a model which performs well on both the backdoor task and the main task. The objective of a malicious participant i can be represented as:

$$w_{i_{mal}} = argmax_{w_i}\left(\sum_{i \in D_{tri}} P[B^j] + \sum_{i \in D_{cln}} P[M]\right) \quad (3)$$

Where $w_{i_{mal}}$ is the local parameters, $j \in [1..J]$ is the type of the discrete trigger pattern used, D_{tri} is triggered dataset, D_{cln} is the clean dataset, B is models performance on the backdoor task, and M is the model's performance on the main task. Here, $D_{tri} \cup D_{cln} = D_i$, which is the entire dataset of client i. We can rewrite eqn.3 based on the way the trigger has been constructed.

$$w_{i_{mal}} \approx w^j_{i_B} \oplus w_{i_M} \quad (4)$$

Where $\oplus$ is a function used to represent the combination of the two parameter terms. We can further decompose based on the parameter changes induced exclusively by the inference trigger w_{inf}, and the contributions of the remaining part of the discrete trigger as $w^j_{i_{rt}}$.

$$w^j_{i_B} \approx w_{inf} \oplus w^j_{i_{rt}}$$

Here, $w^j_{i_{rt}} = w^j_{i_{DT}} - w_{inf}$, with $w^j_{i_{DT}}$ being the entire parameter changes caused by using the discrete trigger pattern j of client i. In the global model, the overall model weight is the summation of the local model updates of the clients.

$$G^{t+1} = G^t + \frac{\eta}{N+K}\left[\sum_{i=1}^{N}(f(w_{i_{mal}})^{t+1} - G^t) + \sum_{i=1}^{K}(f(w_{i_{ben}})^{t+1} - G^t)\right]. \quad (5)$$

Here, $f(w_{i_{mal}})$ is the model updates submitted by the malicious participants and $f(w_{i_{ben}})$ is the model updates submitted by benign clients. N and K denote the number of malicious and benign clients, respectively, selected in the current round by the central server. Without the loss of generality, let's evaluate W_{mal}, the overall objective of the malicious clients in the current round under sequential assumption.

$$W_{mal} \approx N \cdot w_{inf} + J\sum_{i=1}^{N/J} w^j_{i_{rt}} + w_{i_M} \quad (6)$$

J is the total number of different discrete triggers used, which is also the same as the number of batches. From eqn. 6, it is evident that the impact of inference trigger in the overall parameter is directly proportional to the number of malicious clients, while the contributions made by the remaining part of the trigger is restricted by the number of batches, such that $N > J$. It is important to note that $w^j_{i_{rt}}$ is different for different triggers.

IV. EXPERIMENTAL ANALYSIS

A. Setup and Parameters

In this section, let's look at the experimental setup to show the efficiency of our approach.

TABLE I: Comparing the Attack Success Rate of the models trained with three different triggers, tested against all the trigger patterns. The best ASR for each test batch is highlighted [TIN-TinyImageNet].

Train \ Test	Benign		Discrete Trigger 1		Discrete Trigger 2		Discrete Trigger 3		Inference Trigger	
	CIFAR10	TIN	CIFAR10	TIN	CIFAR10	TIN	CIFAR10	TIN	CIFAR10	TIN
Benign	0.0121	0.0000	0.0107	0.0002	0.0019	0.0020	0.0114	0.0000	0.0023	0.0006
Discrete Trigger 1	0.0077	0.0013	0.9831	0.9794	0.0145	0.0259	0.6318	0.0103	0.0713	0.0701
Discrete Trigger 2	0.0013	0.0007	0.0178	0.0472	**0.9808**	0.9741	0.0145	0.0752	0.0218	0.0491
Discrete Trigger 3	0.0028	0.0002	0.5671	0.0266	0.0108	0.0208	0.9852	0.9812	0.0627	0.0749
Inference Trigger	0.0018	0.0011	0.5148	0.2115	0.2183	0.3249	0.4473	0.3114	0.9627	0.9451
Our Approach	0.0035	0.0001	**0.9937**	**0.9914**	0.9643	**0.9978**	**0.9921**	**0.9942**	**0.9866**	**0.9923**

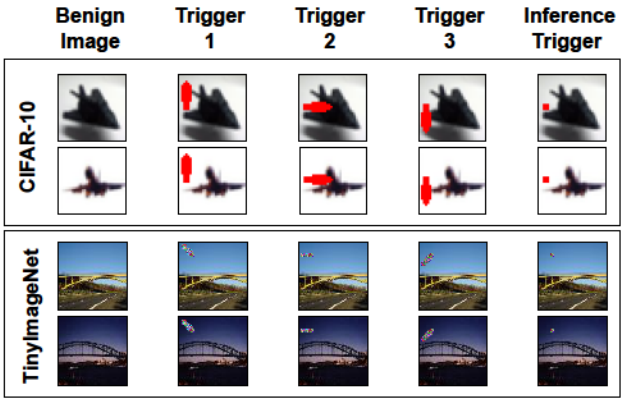


Fig. 2: Different types of triggers added to images in class "airplane" in the CIFAR-10 dataset and in class "steel arch bridge" in the TinyImageNet dataset

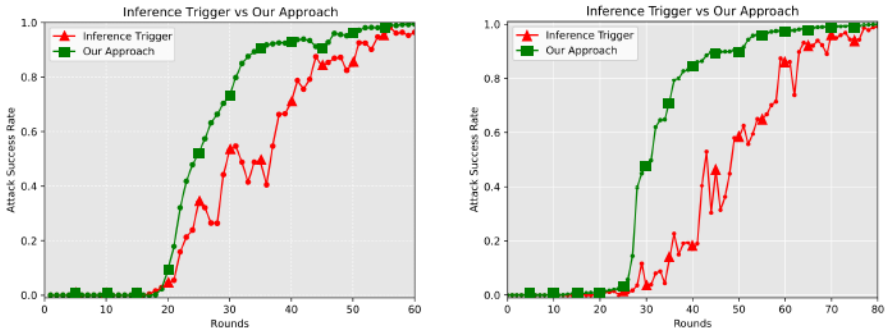
1) Datasets: We use the CIFAR10 [6] and TinyImageNet [7] datasets for our experiments. CIFAR10 consists of 10 classes of images ranging from airplanes to trucks. The images are of a size of 32x32 pixels, with 50,000 training images and 10,000 testing images. Each of the classes consists of 5000 training images. We distributed the randomly shuffled data records evenly among 100 participants. The number of training records for each participant is the same, irrespective of their classes. In other words, the number of images in each class can differ for different participants. Similarly, the TinyImageNet dataset consists of 200 classes, of which we randomly selected 40 classes, with 500, 64x64 images in each class, which was divided among 100 participants based on the Dirichlet distribution, with a distribution hyperparameter of 0.01 [8]. In all our experiments, the backdoor objective is to misclassify the images from the class "airplane" to the target class "horse" in the presence of backdoor triggers for the CIFAR10 dataset. For the TinyImageNet dataset, the backdoor task is to misclassify "steel arch bridge" as "espresso". We only use the inference trigger during test time, except for the results in Table. I, and feature visualization experiments.

2) Model Parameters: We run our experiments with a PyTorch code on Dell G5 laptops with Intel Core i7-10750H CPU, 16 GB of RAM, and NVIDIA GeForce GTX 1660 Ti GPU. We use a ReNet18 [9] model with 100 participants in total. We vary the ratio of malicious participants between 10-25% to check the variation in impact. In each experiment round, 15% of the participants(15 participants) are randomly selected to submit their model updates to the central server. We also check the impact of altering the ratio of attacked images in the target class of the malicious participants.

3) Trigger Generation: For our evaluation, we used three Discrete Triggers, as shown in Fig. 2. For CIFAR10, each of the Discrete Triggers is composed of 56 pixels, which is less than 6% of the size of the image. For TinyImageNet, the discrete triggers are made of 48 pixels, which is only about 1% of the size of the image. The size of the triggers is also influenced by the number of different triggers we can introduce. For CIFAR10, the Inference Trigger comprises only 9 pixels, placed 2 pixels from the left and 14 pixels from the top. The 9 inference pixels are the only pixels that are common in all three Discrete Triggers. The Inference Trigger is placed closer to the mid-left corner of each of the backdoored images. In the Discrete Triggers, the remaining trigger pixels are placed above, below, or to the right of the Inference Trigger. For ease of understanding, let's consider the Discrete Trigger with pixels above the inference pixels as $DT1$; similarly, $DT2$ and $DT3$ correspond to Discrete Trigger patterns with the pixels to the right of and below the inference pixels. In Fig. 2, $DT1$, $DT2$, and $DT3$ correspond to triggers 1, 2, and 3 respectively. For the TinyImageNet dataset, the size of the Inference Trigger is only 16 pixels. Here, $DT1$ and $DT3$, are extended toward the top left and bottom left corners, respectively, from the Inference Trigger. $DT2$, is extended toward the left of the Inference Trigger. We have utilized different colors, sizes, shapes, and angles for the discrete triggers for the two datasets to show the versatility of our approach.

B. Attack Efficiency

For our evaluation, we first compare the performance of our approach with models that were trained with only one



(a) ASR on CIFAR10

(b) ASR on TinyImageNet

Fig. 3: Subfigures (a) and (b) show the performance comparison of our approach against the only inference trigger approach for the CIFAR10 and TinyImageNet datasets respectively.

type of trigger. We use the Attack Success Rate (ASR) as the metric for this comparison. Attack success rate depicts the ratio of triggered images correctly misclassified into the target class to the total number of triggered images used for testing. 15% of all the participants are malicious, and 50% of the local target dataset is poisoned. In our approach, 3% of the poisoned images are from the inference trigger, and each of the discrete triggers was selected sequentially, which will be discussed later.

Table. I shows the attack success rate of different models based on the dataset used for training and the dataset used for testing. The rows indicate the trigger pattern used to train the model, and the columns show the trigger pattern used to test the model. Testing with a 'benign' implies no triggers are used while testing. First, we test training with single trigger patterns, then use 'our approach', which combines all of them.

For CIFAR10, the table shows that when the model is trained using a sizeable discrete trigger pattern, the ASR of the model is 98% when it is tested with the same type of trigger. But surprisingly, when the model is trained with all the malicious participants using only $DT1$, the model still manages to have a high ASR with $DT3$, but not with $DT2$. This result is also true when the model is trained with $DT3$ and tested with $DT1$. This could be because both triggers 1 and 3 are mirror images and have the exact alignment. Though the positions of the two triggers are different, the ResNet model still managed to learn that these two triggers are similar. On the other hand, TinyImageNet does not exhibit a similar phenomenon because all three discrete triggers are oriented in different directions. When using the inference trigger for testing, our method outperforms the model trained only with the Inference Trigger, the most common backdoor attack method.

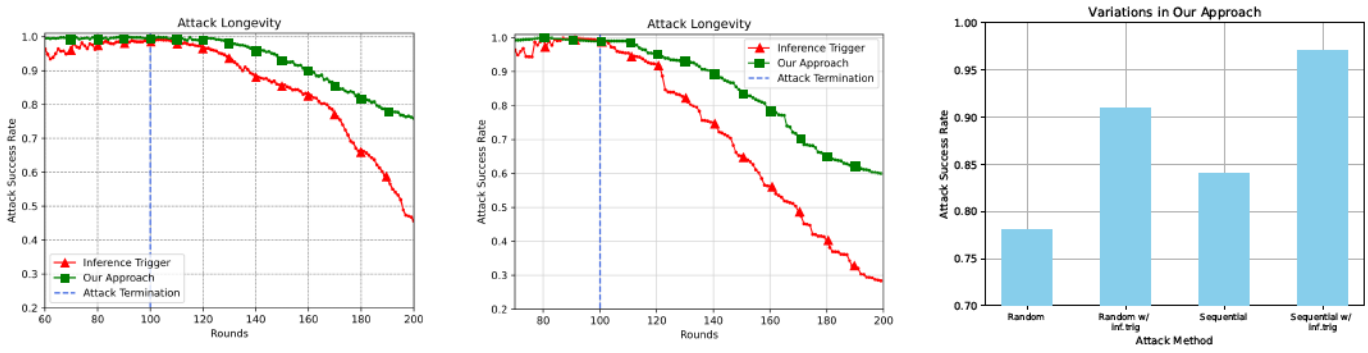
C. Comparison with Single Trigger Attack

Let's discuss our approach's advantages over the single trigger attack method.

1) Attack Responsiveness: From Table. I, we notice that the ASR values vary significantly in the model trained only with inference triggers when tested with the discrete triggers. This substandard performance is because of the model's inability to detect the presence of the inference trigger within the discrete triggers. This shows that when only a single small trigger is used, the trigger size and position should be exactly the same in the train and test datasets for the attack to be effective. On the other hand, our approach learns from all the trigger patterns, and the final model performs well on all of them, including the inference trigger.

2) Rate of Attack: Our approach manages to corrupt the global model faster than the only-inference-trigger model. Fig. 3a and Fig. 3b show the ASR of both the inference trigger trained model and our approach in the first 60 and 80 rounds of the training for CIFAR10 and TinyImageNet, respectively. Our approach has a much smoother and faster increase in the ASR than the inference trigger trained model. The global model finds the larger trigger size to induce stronger features to link the trigger and the target class. On the other hand, the single trigger model has many fluctuations, even with all the malicious clients using the same trigger, indicating that the model quickly forgets the trigger as the number of benign participants in any given round increases. In the first 35 rounds, our approach reaches an ASR of 0.8, while the single trigger model only reaches 0.5.

3) Attack Longevity: In Fig. 4a and Fig. 4b, we compare the presence of the backdoor after the attack has been terminated. We train two models, one based on our approach and one using just the inference trigger. We train both the models for 200 rounds and terminate the attack at round 100, which means that all the participants, both malicious and benign participants, act benignly after round 100. We can notice that our approach manages to stay in the memory of the global model for far longer than a single trigger attack. For CIFAR10, even after 100 rounds of no attack, the global model remembers the backdoor trigger enough to produce 75% ASR, while the single trigger approach gets as low as 46%. Given that the number of samples in each class is much smaller in the TinyImageNet, the decrease in ASR is relatively steeper than



(a) Attack Longevity on CIFAR10 (b) Attack Longevity on TinyImageNet (c) Variations on CIFAR10

Fig. 4: Subfigures (a) and (b) show the presence of the backdoor in the global model for 100 rounds after terminating the attack on CIFAR10 and TinyImageNet, respectively. Subfigure (c) shows the differences in the ASR for the different variations of our approach.

in the CIFAR10 model. Even still, our approach maintains an ASR of 60%, while the only-inference-trigger model drops to less than 30%.

D. Variations in Our Attack

We introduce two variations to our attack to improve the attack speed.

1) Sequential Attack vs Random Attack: Suppose each malicious participants cannot communicate in each round, and each client had access to only one type of trigger pattern; there is a possibility that one of the triggers gets selected disproportionately more than the other triggers. For instance, if $DT1$ is disproportionately selected, then the global model will be more inclined to alter its weights in such a way that the backdoor task is triggered when $DT1$ is present in the test image rather than the Inference Trigger. We term such a situation as the random approach, where the randomness comes from the global server's selections.

Algorithm 1: Sequential Discrete Trigger Selection

Input : Discrete Trigger used by previous malicious client- GlobalPrevDT
Output: Global model trained with sequentially selected DTs.

1 If available, **LOCK(GlobalPrevDT)**
2 **if** *GlobalPrevDT = null* **then**
3 CurrentDT = $DT1$
4 **else**
5 CurrentDT = Next(GlobalPrevDT)
6 **end**
7 GlobalPrevDT = CurrentDT
8 **UNLOCK(GlobalPrevDT)**
9 Embed subset of victim images with CurrentDT
10 Train Local model with entire local dataset.
11 Send local model update to central server
12 **return** *Global model with equal contribution from all DTs*

To avoid the dominance of one of the Discrete Triggers, all the malicious participants can be given access to all the Discrete Trigger patterns; the appropriate Discrete Trigger is selected by the participant of each round. If the previously selected malicious participant used $DT1$ triggered images to train the local model, then the currently selected participant should use $DT2$. Similarly, the next selected participant must select $DT3$ and so on. This selection method is followed across all rounds, ensuring equal contribution from all the Discrete Trigger patterns.

Assuming that the malicious clients have established a channel to communicate, Alg. 1 shows how sequential trigger selection can be achieved in a distributed environment. In the algorithm, we use a global variable $GlobalPrevDT$, which can be accessed and modified by all malicious clients. During any round of the training process, if a malicious client is selected, it runs the algorithm to determine which Discrete trigger pattern needs to be used. The selected malicious client accesses the $GlobalPrevDT$ and places a lock on it if it is available. This lock prevents access to all other malicious clients. The client then determines the $CurrentDT$ for training and modifies the $GlobalPrevDT$ before unlocking it. Once unlocked, other malicious clients can access the variable. The locking and unlocking are shown in lines 1 and 8. This algorithm is independent of the round of attack. Irrespective of the number of rounds and the number of malicious clients selected, we can ensure that the overall contribution of all the clients is similar in the final global model.

2) Presence of Inference Trigger: The other variation that we incorporate is the inclusion of a small percentage of the inference trigger pattern into the attacked images from all the batches of malicious clients. This inclusion helps the model learn the inference trigger faster than training with only the discrete triggers. We only backdoor 3% of the poisoned dataset with the inference trigger. This implies that only 4.5% of the entire poisoned training dataset is backdoored with the inference trigger.

Fig. 4c compares the ASR of all combinations of our approach at round 40 of the training process. We can notice that including the inference trigger in the training data improves ASR significantly. Similarly, using a communication-based sequential approach also increases the ASR above 95%. Though all the models reach better ASR over multiple rounds, the sequential approach with inference trigger combination helps reach high ASR much earlier than other methods.

E. Stealthiness Against Defenses

In this section, we will analyze the stealthiness of our approach over the single trigger method.

1) DeepLIFT Feature Representation: In Fig. 5 (best viewed in color), we use DeepLIFT [3], a feature interpretation mechanism that focuses on discovering the features which have led to the input image being predicted into a particular class. DeepLIFT follows a back-propagation-based approach, where a 'reference' output and input are selected. The difference in the output and the input weights of this reference is used to determine the features that were essential in the output prediction. Fig. 5 shows two sample images from the test dataset, whose essential features have been interpreted using DeepLIFT. We conducted this experiment on the global model after the 100th round, for both the model trained only with the inference trigger (images bordered by red) and our approach (bordered by green). For ease of visibility, we remove the original images from the DeepLIFT representation but only highlight the prominent features using their actual color. The intensity of the colors is proportional to the contribution of those pixels in the final prediction. As we can see, in the model trained with a single trigger, the DeepLIFT representation shows that the trigger has played a vital role in the misclassification of 'airplane' as 'horse'. In both the randomly selected sample images, the trigger pattern is prominently visible; it shows the cluster of trigger pixels that influenced the outcome.

On the other hand, the critical features are more evenly spaced out with the trigger present in our approach. Even with the minimal influence of the inference trigger, the model manages to misclassify the airplane images as horses. Owing to the variations in the different types of triggers used by the malicious clients, the global model learns to associate the input with the target class based on minimal trigger information. The last three columns show the contributing features for trigger patterns 1, 2, and 3. In all the cases, the number of trigger pixels that contribute to the classification is much lesser than the size of the trigger. When it comes to trigger pattern 2, it seems to have a more significant impact on the prediction process. This presence is because trigger 2 overlaps the actual image much more than the other triggers, causing the features of the trigger to be treated as the features of the original image itself. Even if a method detects trigger 2 and gives lower weightage to its location, the other triggers will still successfully misclassify the image.

TABLE II: Comparing the effectiveness of our approach against some of the state-of-the-art defense methods.

		CIFAR10		TinyImageNet	
		MTA	ASR	MTA	ASR
NoAttack		0.9315	0.9866	0.5476	0.9923
RFA	[10]	0.8190	0.9095	0.4052	0.8948
NormClipping	[12]	0.8973	0.9714	0.5118	0.9766
DynamicNC	[8]	0.9158	0.9620	0.5247	0.9703
FoolsGold	[11]	0.7833	0.9741	0.4290	0.9211

F. Impact of Defense Strategies

This subsection tests how well our approach holds up against some of the state-of-the-art defense methods. We deploy the defense mechanism closer to the convergence of the global model. Table. II shows the main task accuracy (MTA), which is the prediction performance of all non-triggered images, and the attack success rate (ASR) after applying the various defenses. In [10], the authors propose to use the geometric mean as the secure aggregation approach. In FoolsGold [11], the authors estimate the similarity metric and build a rule that prevents malicious updates from being aggregated to the global server. From Table. II, we can see that this approach is not effective in protecting against our attack. Similarly, in NormClipping [12], and DynamicNC [8], the defense relies on clipping the weights of the updates if it is beyond a certain threshold. This approach is generally quite effective against model poisoning attacks, where the backdoored models are weight boosted. Since our approach does not rely on model rescaling, this defense is ineffective against our attack. We can notice that the ASR of our approach is maintained close to 90% under all these defense conditions.

V. Related Work

Backdoor attacks on FL have recently gained a lot of attention. Bhagoji et al. induced a targeted attack on the global model from the early rounds of learning by boosting the updates of the malicious agents to suppress the effects of benign agents [13]. [14] show that small, well-crafted changes in the weights, along with some alterations to the loss function, is enough to circumvent both byzantine and backdoor defenses. [5] developed an edge-case backdoor attack, where the labeled examples are chosen such that their input features lie on the heavy tails of the feature distribution. [4] showed that model poisoning attacks outperform data poisoning by developing a technique to replace the global model with the attacker's local model by scaling the local model weights. Distributed Backdoor Attack [15] divides the global trigger pattern into four separate local trigger patterns. In their work, all the local trigger patterns need to be in specific positions, closer to each other, for the attack to succeed.

There exist as many defense methods for Federated Learning as there are attacks. FLGUARD is a secure Federated Learning approach that protects against both inference attacks, as well as multiple-backdoor attacks [16], by using a two-layer scheme with HDBSCAN clustering. In PDGAN [17], the

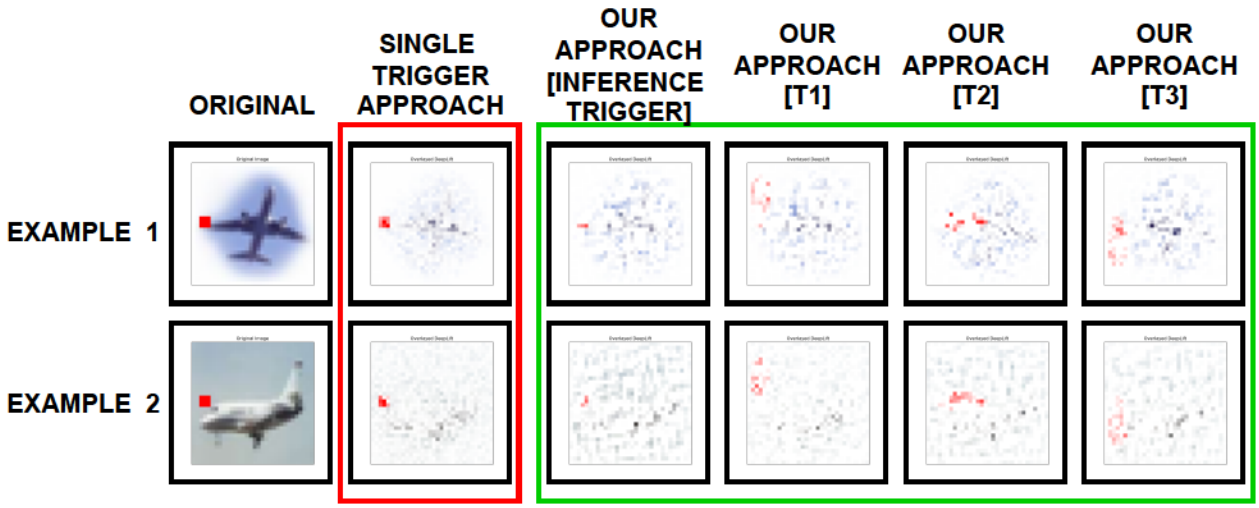


Fig. 5: Learning important features through propagation activation differences using DeepLIFT approach on CIFAR10 Model

authors reconstruct the training data from the model updates and use it to audit the accuracy of the local updates submitted by the clients. Works [18] and [19] adapt a clustering approach to group malicious clients and discard their updates before aggregating them in the global model.

VI. CONCLUSION

In this paper, we have explored the possibility and advantages of using a stealthy distributed backdoor attack method for Federated Learning systems. We have used multiple sizeable discrete triggers and one small inference trigger to backdoor an FL system. We have tested the performance of our approach against the traditional single trigger attack and show that our attack manages to produce 20-25% better ASR than single trigger attacks around round 35 for both CIFAR10 and TinyImageNet datasets. DeepLIFT-based visual feature representations were shown to emphasize the stealthiness of our method. We also show the superior performance of our attack even in the presence of defense methods. In the future, we plan to explore a share-based backdoor attack, where the share's weightage would determine the attack's impact.

ACKNOWLEDGMENT

This work was supported in part by the U.S. National Science Foundation grant CNS-1852105, DGE-2146359, DGE-2011117, OAC-1839746, and DOE CYMANII 1000003897.

REFERENCES

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

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

[3] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3145–3153.

[4] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[5] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *arXiv preprint arXiv:2007.05084*, 2020.

[6] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

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

[8] Y. Guo, Q. Wang, T. Ji, X. Wang, and P. Li, "Resisting distributed backdoor attacks in federated learning: A dynamic norm clipping approach," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1172–1182.

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

[10] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv preprint arXiv:1912.13445*, 2019.

[11] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

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

[13] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[14] M. Baruch, G. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *arXiv preprint arXiv:1902.06156*, 2019.

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

[16] T. D. Nguyen, P. Rieger, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, A.-R. Sadeghi, T. Schneider *et al.*, "Flguard: Secure and private federated learning," *arXiv preprint arXiv:2101.02281*, 2021.

[17] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "Pdgan: a novel poisoning defense method in federated learning using generative adversarial network," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2019, pp. 595–609.

[18] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *arXiv preprint arXiv:2201.00763*, 2022.

[19] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni *et al.*, "Flame: Taming backdoors in federated learning," 2022.