# Malicious Forgetting: Backdoor Injection in Active Federated Unlearning and Countermeasure Design

Wenwei Zhao*, Yuanzhe Peng†, Xiaowen Li*, Jie Xu†, Yao Liu*, Zhuo Lu*
*University of South Florida and †University of South Florida

***Abstract*—Federated learning (FL) enables collaborative model training without sharing raw data, but also raises increasing demands for the right to be forgotten. To support data erasure, active federated unlearning (FU) allows clients to actively remove their data's influence from the model. We reveal a critical and overlooked threat: malicious clients can pose as privacy-concerned users requesting to unlearn some of their data, while secretly preparing backdoor attacks during training. We propose FUsion backdoor, a subnetwork-based attack that stealthily constructs a compact backdoor subnetwork from trigger-sensitive units within backdoor-critical layers during training, and rapidly fuses it during the limited rounds of unlearning. FUsion backdoor achieves up to 99% backdoor success rate across diverse datasets and FU methods. We also develop a detection method that captures directional subspace deviations introduced by coordinated backdoor updates, achieving high attack detection accuracy.**

## I. Introduction

Federated Learning (FL) is a machine learning paradigm enabling multiple clients to train a global model collaboratively without sharing their raw data [1]. With the increasing emphasis on data privacy, regulations such as the European Union's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) grant individuals the right to be forgotten (RTBF) [2]–[4], necessitating mechanisms that allow users to remove their data and its influence from trained models upon request, motivating the use of federated unlearning (FU) [3]–[5], which aims to remove specific data contributions while preserving model performance efficiently.

FU methods can be categorized into passive and active designs [5]. In passive unlearning, the target clients (i.e., the clients whose data is to be removed) exit the system after a removal request is issued and do not engage in the unlearning process. This typically applies to client-level scenarios [4], [6], [7] where the entire contribution of a client, often considered malicious or uncooperative, is removed. In contrast, active unlearning requires target clients to actively participate in the unlearning process, as is the case in sample-level [3], [8] or class-level [9], [10] unlearning.

FU is typically conducted under the assumption that participating clients are benign and trusted [3], [5], [8]. Such an assumption does not always hold in practice and the security study for FU is still in its preliminary stage. A very recent work [11] demonstrated a scenario of attacking passive FU, in which unlearning is used to remove the impact of malicious nodes after they are detected. Malicious clients in the attack need to both poison the global model during training and send malicious updates during unlearning. However, in active FU where clients are allowed to submit removal requests (e.g., RTBF), malicious clients are not confined to first poisoning the global model during training as a necessary step, in particular considering a wide range of defenses proposed to secure FL training [12]–[14]. They can actively submit removal requests and focus on fully attacking the active FU process, which poses a potential serious yet under-explored research issue.

In this paper, we consider an active unlearning scenario where malicious clients behave normally during training to avoid being flagged by FL defense mechanisms, but later actively request data unlearning. We focus on a dangerous form of manipulation: backdoor injection. We propose a novel attack strategy, FUsion backdoor, to rapidly implant backdoor functionality within the limited rounds of active FU, while preserving the performance of the main task (i.e., maintaining high accuracy on benign data). Specifically, FUsion backdoor takes advantage of the stealthy preparation performed by malicious clients during the training phase, they collaboratively identify a contiguous backdoor-critical (BC) layer and select a small number of trigger-sensitive (TS) units within it to construct a compact backdoor subnetwork in secret. Once the preparation is complete, the malicious clients request unlearning and remain actively engaged: they first perform legitimate clean-data unlearning as required, then fuse the prepared backdoor subnetwork into the model and optimize it further using both backdoor and clean data. This strategy leverages prior preparation and full participation of malicious clients to enable efficient, rapid, and stealthy backdoor injection within the limited unlearning rounds. Evaluations demonstrate that FUsion backdoor consistently achieves high success rates of 62% - 99% across unlearning methods and datasets, while preserving main task performance and exceeding baseline attacks in both efficiency and stealth.

To combat FUsion backdoor injection, we observe that existing FL defenses, including robust aggregation [13]–[16] and anomaly detection [12], [17], [18], are rarely applied to the unlearning process, as they are often computationally expensive, rely on long-term behavioral observation [12], [16], and assume stable and similar client behaviors [14], [17]. These assumptions fundamentally conflict with the goals of FU, which aims to rapidly restore model utility under inherently unstable and heterogeneous client updates [5], [19]. As a result, we propose an effective defense method called SubID (Subspace-based deviation IDentification) for its unlearning phase. Specifically, we observe that during training, client updates exhibit diverse and uncoordinated directions, forming

a benign subspace. During unlearning, malicious clients tend to align their updates along a consistent, trigger-oriented subspace to implant backdoors. By extracting the benign subspace from the training phase and identifying deviation subspaces during unlearning, we measure each client's alignment with these subspaces to reliably distinguish malicious updates from benign unlearning-induced noise. Evaluations demonstrate that SubID accurately identifies malicious clients across various FU methods and datasets. It achieves 100% detection with no false positives in most cases, showing both precision and reliability.

Our contributions can be summarized as follows: 1) We propose a fast and effective FUsion backdoor against active FU. 2) We show that the proposed attack achieves a backdoor accuracy up to 99% within 200 unlearning rounds. 3) We also introduce SubID, a detection mechanism that identifies malicious clients by analyzing directional deviations between behavioral subspaces observed during training and unlearning.

## II. Motivation and Threat Model

In this section, we first introduce the background of backdoor attacks in FL, then present the security challenges in active FU and lastly present our threat model and assumptions.

### A. Federated Learning and Backdoor Attack

FL is a decentralized learning framework where a set of clients $\mathcal{C} = \{1, 2, \ldots, n\}$ collaboratively train a global model with parameter $\mathbf{w}$. In each global round $t$, the server distributes the current model $\mathbf{w}_t$ to selected clients. Each client $i$ holds a private dataset $\mathcal{D}_i = \{(\mathbf{x}_j, y_j)\}_{j=1}^{u_i}$ of $u_i$ input-label pairs, where $\mathbf{x}_j \in \mathbb{R}^d$ is the input feature and $y_j \in \mathcal{Y}$ is the corresponding label. The client performs local training and computes an update:

$$\mathbf{g}_i^{t+1} = -\eta \cdot \nabla_{\mathbf{w}_t} \, \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_i} \left[\mathcal{L}(\mathbf{w}_t; \mathbf{x}, y)\right], \tag{1}$$

where $\eta$ is the local learning rate, $\mathcal{L}$ is the loss function, and the expectation $\mathbb{E}$ is taken over the local data distribution of client $i$. Each client uploads its local update $\mathbf{g}_i^{t+1}$ to the server, which applies an aggregation function $\mathcal{A}$ [14] to obtain the next global model $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathcal{A}(\{\mathbf{g}_i^{t+1}\}_{i\in\mathcal{C}})$.

However, FL is vulnerable to poisoning attacks, especially backdoor attacks [20], where malicious clients poison part of their local data by injecting a predefined trigger pattern into the input and assigning a target label $y_t$. The resulting poisoned dataset is denoted as $\mathcal{D}_i^{\text{p}} = \{(\mathbf{x}'_j, y_t)\}_{j=1}^{v_i}$ with $v_i$ number of data, where $\mathbf{x}'_j$ is a modified input containing the trigger. The remaining clean data is denoted by $\mathcal{D}_i^{\text{c}} = \mathcal{D}_i \setminus \mathcal{D}_i^{\text{p}}$.

To simultaneously preserve utility on benign data and inject the backdoor, the malicious client optimizes a composite loss:

$$\mathcal{L}_{\text{mal}}(\mathbf{w}) = \mathbb{E}_{\mathcal{D}_i^{\text{c}}} \left[\mathcal{L}(\mathbf{w}; \mathbf{x}, y)\right] + \gamma \cdot \mathbb{E}_{\mathcal{D}_i^{\text{p}}} \left[\mathcal{L}(\mathbf{w}; \mathbf{x}', y_t)\right], \tag{2}$$

where $\gamma > 0$ is a weighting coefficient that controls the trade-off between maintaining performance on clean data and maximizing the backdoor effect. The client then computes model updates based on this loss to subtly encode the trigger behavior into the global model.

Let $\mathcal{C}_{\text{mal}} \subset \mathcal{C}$ denote the set of malicious clients. Their crafted updates $\tilde{\mathbf{g}}_i^t$, aggregated together with benign ones $\mathbf{g}_i^t$ in each round $t$, cause the global model to learn the trigger-to-target mapping over time. The effectiveness of such attacks is typically evaluated using two metrics [17], [21]: the backdoor accuracy (BA) that measures the model's prediction accuracy on inputs containing the trigger, and the main task accuracy that measures the model's performance on clean data.

### B. Active Federated Unlearning and Security Challenges

During FL, clients may, out of privacy or regulatory concerns (e.g., RTBF [2]–[4]), request the server to remove a portion of their local data from the global model [3], [22], [23]. In response, the server initiates the process of active FU, during which the requesting clients are assumed to be benign and actively participate in the process [3], [23], [24]. In active FU, clients and the server usually work together to mitigate the influence of the unlearn data $D_{\text{unlearn}}$. Different from standard FL, active FU shifts the objective from learning to forgetting, fundamentally altering the optimization dynamics [24], [25], often by optimizing a negation or correction objective:

$$\min_{\mathbf{w}} \; \mathcal{L}(\mathcal{D}_{\text{retain}}; \mathbf{w}) + \varphi \cdot \mathcal{R}(\mathbf{w}, \mathbf{w}_T, \mathcal{D}_{\text{unlearn}}), \tag{3}$$

where $\mathcal{D}_{\text{retain}} = \mathcal{D} \setminus \mathcal{D}_{\text{unlearn}}$ is the remaining data, $\mathbf{w}_T$ is the global model at the end of training, and $\mathcal{R}(\cdot)$ is a regularization term designed to reverse the influence of the unlearned data. In practice, directly solving Equation (3) is often intractable. As a result, active FU methods typically approximate this objective using lightweight retraining or correction strategies, such as model scrubbing [3], [26] and multitask learning [22], [27].

However, these introduce new challenges. Due to data removal and modified objectives, client updates during active FU tend to be more unstable and heterogeneous [7]. Moreover, privacy constraints usually prevent the server from directly observing the unlearned data, making it difficult to verify whether clients have performed unlearning honestly. These limitations not only affect the reliability of unlearning, but also create new attack surfaces for adversaries.

In particular, they open up new opportunities for adversarial manipulation. Malicious clients can pretend to be benign participants during the training phase, submitting normal updates to avoid detection while secretly preparing backdoor updates offline. Once prepared, they request the server to unlearn part of their data, citing privacy or regulatory concerns. During the unlearning phase, these clients can potentially inject crafted backdoor updates disguised as legitimate unlearning. This scenario exploits the weaker verification limitations of FU, allowing attackers to rapidly embed backdoor functionality into the global model without being detected. As a result, how to design and combat backdoor attacks in active FU remains a critical, yet unanswered question.

### C. Threat Model and Attack Assumption

We consider an FL involving $n$ clients, all of whom participate in every training round. Among these, a subset $\mathcal{C}_{\text{mal}} \subset \mathcal{C}$ of $m < \frac{n}{2}$ clients are malicious, controlled by an attacker

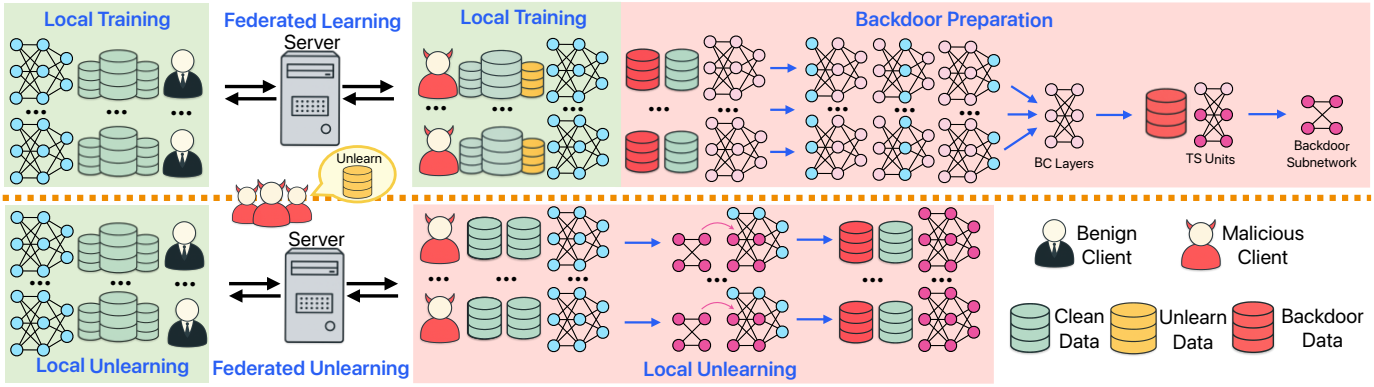


Fig. 1: Overview of the proposed FUsion backdoor attack in active FU.

aiming to inject a backdoor into the global model without being detected. We assume that the attacker has full access to the local data and training procedure of each malicious client. Specifically, the attacker can modify local training data (e.g., inserting poisoned samples with a specific trigger and target label) and arbitrarily alter the local model updates generated by the malicious clients. However, the attacker does not have access to local data or model updates of benign clients, nor do they know the defense mechanisms on the server side.

Due to this uncertainty, malicious clients refrain from launching direct backdoor attacks during training. Instead, they behave honestly and submit clean updates similar to benign clients. Meanwhile, each malicious client locally constructs and stores potential backdoor updates trained on poisoned data, which are not submitted to the server, but retained for future use. Once the malicious clients are prepared, they initiate the attack by actively requesting data removal, i.e., by submitting unlearning requests to the server. Such requests may be framed as legitimate privacy concerns, prompting the server to initiate the FU procedure. Then, malicious clients participate in the unlearning process and strategically deploy the preconstructed backdoor updates to achieve effective trigger injection.

## III. Backdoor Attack Design against Active FU

In this section, we propose the FUsion backdoor attack against active FU, and use experiments to evaluate its impact.

### A. Overview

As discussed in Section II-B, existing strategies of backdoor injection in FL cannot be readily adapted to the active FU scenario. We propose a two-step FUsion backdoor framework in which malicious clients behave benignly and prepare stealthily during training, and only activate their attack during the unlearning phase by submitting malicious updates. The procedure is schematically illustrated in Figure 1.

1) While training proceeds, the malicious clients secretly collaborate to identify a set of backdoor-critical (BC) layers and the most trigger-sensitive (TS) units within them, and construct a compact, independently trained backdoor subnetwork. Once the backdoor subnetwork is prepared, the malicious clients submit a request to remove a subset of their local data. This request triggers a system transition to the unlearning phase.

2) During each unlearning round, the malicious clients first perform legitimate clean-data unlearning as required by the protocol in Equation (3). Subsequently, they fuse the backdoor subnetwork into the model and further optimize the model on a dataset containing both backdoor and clean data. This strategy enables efficient and effective backdoor injection within the limited unlearning rounds, while minimizing interference with the model's primary task performance. The remainder of this section details the selection and the construction of the backdoor subnetwork, and its deployment during unlearning.

### B. Backdoor Preparation during Training

During the FL training phase, the attacker prepares the backdoor by selecting and constructing a structured subnetwork that isolates the backdoor functionality within it. Prior work shows that explicitly adding a dedicated subnetwork [28], or replacing parts of a trained model [29], can effectively implant a backdoor. However, in active FU, the attacker is not allowed to modify the model architecture or alter the global model directly. Moreover, due to benign client aggregation, simply replacing a trained local model with the backdoor subnetwork would be overridden in the final aggregation. Thus, we instead identify an independent subnetwork within the existing model structure to carry the backdoor information, which is later fused into local updates during unlearning.

Existing approaches identify backdoor subnetworks either by empirically selecting a high-level layer and randomly choosing a few channels [29], or by leveraging the Lottery Ticket Hypothesis (LTH) [30] to prune trigger-sensitive sparse weights. However, the former is not heuristic and does not guarantee trigger sensitivity, while the latter requires large amounts of data and the pruned weights do not form a structured, independently trainable subnetwork, but rather a sparse weight mask that depends on the full model context [31]. These limitations make these approaches less effective for preparing backdoor subnetwork against active FU.

*1) Backdoor Critical Layer Identification:* Recent studies show that backdoor information is primarily concentrated in a small subset of layers, called BC layers, which dominate the model's vulnerability to triggers: modifying only these

layers can achieve comparable backdoor success with less perturbation [31]. While existing methods identify BC layers via forward and backward substitution, the selected layers are often non-contiguous [31], making it difficult to form a structural subnetwork. Training a backdoor on non-contiguous layers may disrupt the flow of trigger-related representations through the network, as intermediate untrained layers can attenuate or distort the trigger signal. Prior work [32], [33] observed that BC layers tend to appear early in the network, where their responses are naturally amplified by subsequent layers. Based on these insights, we select a contiguous block of BC layers in the early network, rather than including all identified ones, to ensure smooth propagation and stable amplification of the trigger signal, thereby improving the robustness and effectiveness of the attack.

During training, malicious clients collaboratively identify the BC layers as the locations where the backdoor subnetwork will later be fused and trained (note that these layers are not the subnetwork itself but are where it will be inserted and trained). This design leverages the fact that active FU removes only a subset of data and retains the same model structure without acquiring new features [3], [34]. Therefore, layers sensitive to the trigger during training remain sensitive during unlearning. Based on this intuition, we determine a contiguous set of BC layers using a combination of forward layer substitution analysis and our proposed contiguous selection heuristic. At the start of training, each malicious client $i \in \mathcal{C}_{\text{mal}}$ secretly trains a local backdoor model $\mathbf{g}_i^{\text{bd}}$ on its backdoor data while simultaneously participating in standard FL on clean data. Once both the global model $\mathbf{w}$ and local $\mathbf{g}_i^{\text{bd}}$ reach a stable state, where clean accuracy and BA remain nearly constant, we perform BC-layer selection based on the current $\mathbf{w}$.

We quantify the contribution of each layer $h \in 1, \ldots, H$, where $H$ is the total number of layers, to the backdoor effect by measuring the drop in BA when layer $h$ in $\mathbf{g}_i^{\text{bd}}$ is replaced with its benign counterpart in $\mathbf{w}$. Let $\Delta b_i(h)$ denote the BA drop for client $i$ at layer $h$, which we then normalize as

$$\delta_i(h) = \Delta b_i(h) \,/\, \max_{1 \le j \le H} \Delta b_i(j), \tag{4}$$

where $\delta_i(h) \in [0, 1]$ reflects the relative importance of layer $h$ to the backdoor effect on client $i$.

To aggregate these measurements across all malicious clients, we compute the mean normalized contribution:

$$\bar{\delta}(h) = \frac{1}{|\mathcal{C}_{\text{mal}}|} \sum_{i \in \mathcal{C}_{\text{mal}}} \delta_i(h), \tag{5}$$

and identify the layer with the highest aggregated contribution:

$$h^* = \arg\max_h \bar{\delta}(h), \tag{6}$$

which serves as the anchor of the global BC set.

Starting from $h^*$, we expand the BC set symmetrically to include adjacent layers as long as they are trainable and their $\bar{\delta}(h)$ remains above a fraction $\theta$ of the anchor's contribution:

$$\bar{\delta}(h) \ge \theta \cdot \bar{\delta}(h^*), \tag{7}$$

where $\theta \in (0, 1)$ is a tunable threshold (e.g., $\theta = 0.6$) that controls the minimum relative contribution required for inclusion. Layers are added as long as the condition holds and they remain contiguous and trainable. Although bias terms are technically trainable, they contribute minimally to feature learning [35] and thus we exclude them from the BC-layer set to reduce parameter footprint. If a non-trainable layer is encountered during expansion, we simply skip it and continue; expansion stops in a given direction only when the normalized contribution $\bar{\delta}(h)$ of the next trainable layer falls below $\theta \cdot \bar{\delta}(h^*)$. This procedure yields a single global contiguous set of layers

$$\mathcal{H}_{\text{BC}} = \{h_{\min}, h_{\min} + 1, \ldots, h_{\max}\}, \tag{8}$$

which includes the anchor layer $h^*$ ($h_{\min} \le h^* \le h_{\max}$). The resulting $\mathcal{H}_{\text{BC}}$ balances backdoor effectiveness and parameter efficiency by selecting a contiguous set of layers that contribute significantly to the trigger response, ensuring that the trigger-induced features propagate smoothly through the subnetwork without being suppressed or distorted by intermediate layers. Usually, the global contiguous BC set $\mathcal{H}_{\text{BC}}$ can achieve strong and stable backdoor injection [28]. Although malicious clients may hold highly non-IID data, our goal is to compromise the global model after unlearning rather than ensuring perfect backdoor performance on each individual client. Therefore, a unified BC-layer set $\mathcal{H}_{\text{BC}}$ is sufficient to inject a strong and stable backdoor globally.

*2) Trigger Sensitive Units Selection:* After identifying the contiguous BC-layer set $\mathcal{H}_{\text{BC}}$, we construct a backdoor subnetwork within it that encodes trigger-specific features while remaining compatible with the original model. However, not all trainable units in $\mathcal{H}_{\text{BC}}$ are sensitive to the trigger [32], and including all units may introduce noise and reduce attack efficiency, particularly in the constrained setting of active FU. To address these limitations, we employ Gradient-weighted Class Activation Mapping (Grad-CAM) [36] to estimate the trigger sensitivity of each trainable unit in $\mathcal{H}_{\text{BC}}$ by combining forward activations and backward gradients of the target logit of each backdoor input $(\mathbf{x}', y_t)$ to generate a localization map that highlights the most salient units. To ensure consistency and efficiency, we designate a coordinator among malicious clients to construct the backdoor subnetwork. This client computes Grad-CAM scores over its local backdoor data, identifies the top-$\rho\%$ trigger-sensitive units in each BC layer, and shares the resulting subnetwork structure with other malicious clients. Unlike prior approaches relying on heuristics or random selection, this principled and fine-grained strategy improves precision and stealth, enabling more efficient and robust backdoor injection within limited unlearning rounds.

*3) Backdoor Subnetwork Construction:* After identifying the contiguous BC layer set $\mathcal{H}_{\text{BC}}$ and TS units within it, we construct the backdoor subnetwork that encodes trigger-specific features while remaining compatible with the original network structure. The subnetwork retains the same input and output dimensions as $\mathcal{H}_{\text{BC}}$, but only TS units are optimized for the backdoor while the rest preserve clean-task parameters.

For each layer $h \in \mathcal{H}_{\mathrm{BC}}$, we assign selected units to the backdoor subnetwork, initializing them from the global model. Malicious clients collaboratively train these units on backdoor data, with a small fraction of clean data included to improve robustness. Each client optimizes its local copy of the backdoor units, and the updates are privately aggregated (e.g., by averaging) into a unified subnetwork. The remaining units stay fixed to preserve benign-task functionality.

The input to the subnetwork, denoted $\mathbf{z}_{\mathrm{in}}$, equals the raw input $\mathbf{x}$ when $h_{\min} = 1$, or the intermediate feature map produced by the preceding layers when $h_{\min} > 1$:

$$\mathbf{z}_{\mathrm{in}} = \begin{cases} \mathbf{x}, & \text{if } h_{\min} = 1 \\ f_{1:h_{\min}-1}(\mathbf{x}), & \text{if } h_{\min} > 1 \end{cases} \tag{9}$$

where $f_{1:h_{\min}-1}$ represents the composition of the first $(h_{\min}-1)$ layers of the global model, mapping $\mathbf{x}$ to an intermediate feature map. The output of the backdoor subnetwork, $\mathbf{z}_{\mathrm{out}} = f_{\mathcal{H}_{\mathrm{BC}}}(\mathbf{z}_{\mathrm{in}})$, serves as the input to the subsequent fixed layers of the global model, which produce the final prediction.

The training objective of the backdoor subnetwork balances backdoor effectiveness and stealthiness:

$$\mathcal{L}_{\text{sub}} = \lambda_1 \cdot \mathcal{L}\big(f_{\mathcal{H}_{\mathrm{BC}}}(\mathbf{z}_{\mathrm{bd}}), y_{\mathrm{t}}\big) + \lambda_2 \cdot \big\| f_{\mathcal{H}_{\mathrm{BC}}}(\mathbf{z}_{\mathrm{clean}}) \big\|_2, \tag{10}$$

where the first term encourages backdoor-specific units to classify backdoor inputs $\mathbf{z}_{\mathrm{bd}}$ into the target class $y_{\text{target}}$, and the second term regularizes activations on clean inputs $\mathbf{z}_{\mathrm{clean}}$. $\lambda_1$ and $\lambda_2$ balance the trade-off between backdoor effectiveness and clean-task performance.

Compared to modifying the entire BC-layer set $\mathcal{H}_{\mathrm{BC}}$, our subnetwork updates only a small subset of dedicated units. This concentrates trigger-related features into a few specialized units that are highly sensitive to the trigger while minimally responsive to clean inputs, enhancing both backdoor effectiveness and stealth. It also improves robustness to clean-data fine-tuning and reduces global model perturbation, making the attack less detectable.

### C. Backdoor Injection during Active FU

After training, each malicious client holds a compact backdoor subnetwork $\mathcal{S}_{\mathrm{bd}}$ within the identified BC-layer set $\mathcal{H}_{\mathrm{BC}}$. During unlearning, the attack is activated in two steps: subnetwork fusion and lightweight fine-tuning, as illustrated in the local unlearning part of Figure 1. At each unlearning round $t$, the server broadcasts the current global model $\hat{\mathbf{w}}^t$. Each malicious client initializes its local model $\hat{\mathbf{w}}_i^t$ with $\hat{\mathbf{w}}^t$ and performs standard local unlearning on its retained clean data. To inject the backdoor, the client fuses the pre-trained subnetwork into $\hat{\mathbf{w}}_i^t$, which has been obtained by optimizing the unlearning objective in Equation (3) at round $t$.

$$\tilde{\mathbf{w}}_i^t = \hat{\mathbf{w}}_i^t + \beta \cdot \mathcal{S}_{\mathrm{bd}}, \tag{11}$$

where $\beta$ controls the injection strength. The subnetwork$\mathcal{S}_{\mathrm{bd}}$ has nonzero entries only in $\mathcal{H}_{\mathrm{BC}}$ and zeros elsewhere, ensuring that the perturbation remains localized and minimally affects unrelated parameters.

The fused model $\tilde{\mathbf{w}}_i^t$ is then fine-tuned on a mix of clean and backdoor data to obtain the final model $\tilde{\mathbf{w}}_i^{t*}$, enhancing compatibility between the backdoor perturbation and the updated local model. This is achieved by minimizing the objective:

$$\mathcal{L}_{\text{unlearn}} = \mathcal{L}\big(f(\mathbf{x}_{\text{clean}}), y_{\text{clean}}\big) + \lambda \cdot \mathcal{L}\big(f(\mathbf{x}_{\text{backdoor}}), y_{\text{target}}\big), \tag{12}$$

where $\lambda$ controls the trade-off between main-task accuracy and backdoor success.

Finally, the client computes the local update $\tilde{\mathbf{g}}_i^t = \tilde{\mathbf{w}}_i^{t*} - \hat{\mathbf{w}}^t$, and submits $\tilde{\mathbf{g}}_i^t$ to the server. Our subnetwork-based attack enables fast backdoor injection during unlearning. Leveraging pre-encoded trigger features in $\mathcal{S}_{\mathrm{bd}}$, each round needs only 2-5 fine-tuning epochs, significantly reducing training overhead while maintaining attack effectiveness.

### D. Experiment

In the following, we evaluate the impact of the proposed backdoor attack in an active FU system.

*1) Experiment Setup:* **Dataset and Model Architectures:** We conduct experiments on four widely used benchmark datasets: MNIST [37], CIFAR10 [38], Human Activity Recognition (HAR) [39], and AGNEWS [40]. We employ commonly adopted neural network architectures: a lightweight CNN [41] for MNIST, ResNet18 [42] for CIFAR10, HAR-CNN [43] for HAR, and FastTEXT [44] for AGNEWS. To simulate realistic data distributions, we evaluate both IID and non-IID scenarios. For the non-IID case, we partition the data among clients using a Dirichlet distribution, which effectively models the heterogeneity observed in real-world FL. The concentration parameter of the Dirichlet distribution is set to $0.5$, controlling the degree of data heterogeneity.

**Attack Setup:** We inject a predefined trigger pattern into $10\%$ of the training samples and assign them a fixed target label for targeted misclassification. The triggers are designed based on dataset characteristics: a $3 \times 3$ white square at the bottom-right corner for MNIST, a $5 \times 5$ red "X" trigger at the bottom-left corner for CIFAR10, a linearly decreasing sequence in the last four time steps of one channel for HAR, and a rare token randomly inserted into the sentence for AGNEWS. In all cases, poisoned samples are relabeled to a fixed target class.

**Active FU Setup:** We consider three representative active FU methods: NoT [23], FedRR [3], and ConFUSE [45], where clients can actively request unlearning and participate in the unlearning process. 1) NoT performs unlearning by introducing negation of training to reverse the influence of the leaving client's data. 2) FedRR achieves unlearning by approximating the Fisher Information Matrix diagonally and applying fast Newton-type updates. 3) ConFUSE injects confusion-based updates on salient model components related to the unlearned data. These methods represent the state-of-the-art in FU, with their core principles respectively based on retraining, model scrubbing, and multi-task learning to optimaze Equation (3).

**Default Scenario:** We consider a default setting on the MNIST dataset, where a CNN is trained with SGD (learning rate 0.01, momentum 0.9) under IID client data. The system

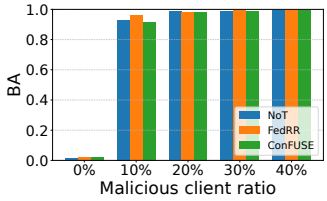


Fig. 2: BA under different malicious client ratios.

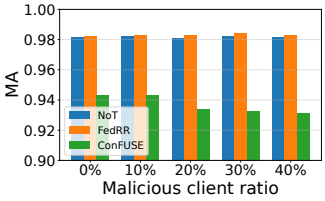


Fig. 3: MA under different malicious client ratios.

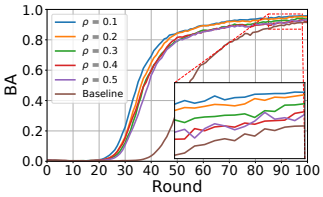


Fig. 4: BA over unlearning rounds at different ratios $\rho$.

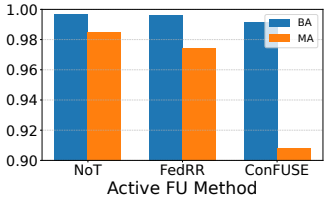


Fig. 5: BA and MA under non-IID data scenarios.

includes $n = 50$ clients, with $m = 10$ (20%) randomly selected as malicious clients. Each malicious client injects 10% backdoor samples into its local data. All clients are selected in aggregation every round. Training runs for 500 rounds to reach convergence, followed by 200 unlearning rounds simulating client data removal, using FedAvg [14] as the aggregation rule. The active FU method is NoT by default, with each malicious client unlearning 2% of its local training data. We set the BC layer threshold $\theta = 0.6$, TS unit ratio $\rho = 10\%$, subnetwork strength $\beta = 0.2$, and loss weights $\lambda = 0.5$, $\lambda_1 = 1$, $\lambda_2 = 0.5$. The baseline attack directly injects backdoor data during unlearning.

**Evaluation Metrics:** We consider BA and MA as aforementioned in Section II-A. Both metrics evaluate the performance of the final global model. We define a successful attack as one where the global model exhibits a BA exceeding 60% [17].

*2) Experimental Results:* **Attack Performance across Datasets, Models, and Trigger Patterns:** We first evaluate our FUsion backdoor attack on four datasets using their respective model architectures and trigger patterns. As shown in Table I, our attack method achieves successful attacks (BA $>$ 60%) across all datasets and FU methods. Compared to the Baseline, our attack consistently yields higher BA and MA under FedRR and ConFUSE, where direct data injection fails (BA $\approx$ 0) due to their limited ability to learn new features during unlearning. In contrast, FUsion backdoor fuses a backdoor subnetwork and fine-tunes it to enhance both attack success and main task performance. For NoT, which adopts a retraining-based unlearning strategy, both our attack and Baseline perform well on MNIST. However, on more complex datasets like HAR and AG News, the Baseline underperforms (BA = 13% on HAR, BA = 25% on AG News), likely due to the limited unlearning rounds, which is insufficient for fully implanting the backdoor. In contrast, our method achieves 62% BA by more efficiently injecting the backdoor, demonstrating its effectiveness in time-constrained unlearning settings.

TABLE I: Attack performance under different datasets.

| **FU Method** | **NoT** | | | | **FedRR** | | | | **ConFUSE** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Attack** | FUsion | | Baseline | | FUsion | | Baseline | | FUsion | | Baseline | |
| **Dataset** | BA | MA | BA | MA | BA | MA | BA | MA | BA | MA | BA | MA |
| MNIST | 0.98 | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 | 0.04 | 0.98 | 0.98 | 0.93 | 0.01 | 0.91 |
| Cifar10 | 0.99 | 0.87 | 0.96 | 0.86 | 0.99 | 0.90 | 0.01 | 0.89 | 0.99 | 0.91 | 0.00 | 0.89 |
| HAR | 0.76 | 0.92 | 0.13 | 0.90 | 0.72 | 0.90 | 0.00 | 0.89 | 0.78 | 0.79 | 0.00 | 0.76 |
| AG News | 0.62 | 0.88 | 0.25 | 0.88 | 0.68 | 0.82 | 0.01 | 0.77 | 0.61 | 0.71 | 0.00 | 0.57 |

**Impact of Malicious Client Ratio:** We evaluate the effectiveness of our attack under varying malicious client ratios $m/n$, where $m/n = 0$ indicates no attack is present during unlearning. It can be observed from Figure 2 that our attack successfully achieves backdoor injection when $m/n \geq 10\%$ across all three active FU methods. Even at $m/n = 10\%$, BA remains above 90%, only $3\% - 5\%$ lower than higher ratios. Demonstrating that our attack method can succeed even when malicious clients are in the significant minority.

In addition, in Figure 3, the main task performance is largely preserved across different values of $m/n$. Only when m/n = 40% under ConFUSE does the MA drop slightly by around 1%, indicating that our method has a minimal impact on the performance over non-backdoor data.

**Impact of TS Unit Ratio and Attack Efficiency:** We evaluate how the selection ratio $\rho$ of TS units affects attack performance. As shown in Figure 4, increasing $\rho$ from 0.1 to 0.5 slightly slows down the attack, though all settings consistently outperform the baseline approach. For instance, at the same round, $\rho = 0.1$ achieves a BA of 60% , $\rho = 0.5$ reaches 36%, while the baseline still below 3%. All configurations eventually exceed a BA of 98%, but smaller $\rho$ enables faster backdoor injection, which is particularly effective against active FU that prioritizes rapid model recovery. The zoomed-in region further shows that after round 80, $\rho = 0.1$ yields the highest and most stable BA, whereas larger $\rho$ leads to 3% lower BA and more fluctuation. This suggests that backdoors primarily activate a few trigger-sensitive units. Selecting too many TS units introduces redundancy, may dilute the attack signal, and reduce stability, while a smaller $\rho$ leads to more efficient subnetwork construction and stronger attack focus.

**Attack Performance under Non-IID Data:** We evaluate our attack under a non-IID setting. Figure 5 shows that it remains effective across all three FU methods, with BA exceeding 98%. This demonstrates robustness to data heterogeneity and does not rely on IID assumptions. The slight drop observed in MA is largely attributed to the inherent challenges of FL under non-IID data distributions, such as local bias and reduced generalization capability.

**Impact of Subnetwork Fusion Strength $\beta$:** We evaluate how the fusion strength $\beta$ affects attack performance. Figure 6 shows that increasing $\beta$ improves BA across all FU methods, as a larger $\beta$ amplifies the backdoor subnetwork's effect during training. FedRR exhibits the largest increase, with BA rising from 98.3% at $\beta = 0.2$ to 99.2% at $\beta = 0.8$. While NoT shows minimal change (less than 0.1%), likely due to its stable retraining-based process. Figure 7 shows a slight MA decline with larger $\beta$, as stronger backdoor injection can interfere with the main task. But the drop is limited as unlearning is completed before injection and followed by finetuning.

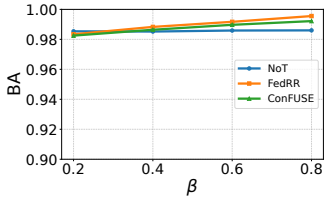


Fig. 6: BA under different fusion strengths $\beta$.

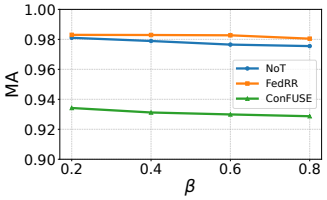


Fig. 7: MA under different fusion strengths $\beta$.

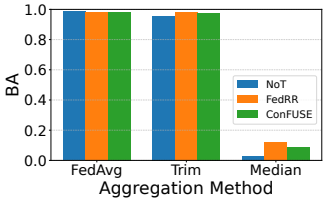


Fig. 8: BA under three aggregation strategies.

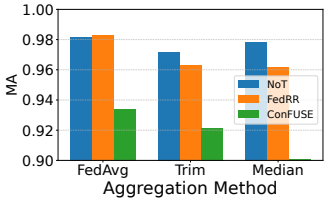


Fig. 9: MA under three aggregation strategies.

Specifically, when $\beta$ goes from 0.2 to 0.8, MA decreases by only 0.5% (NoT), 0.3% (FedRR), and 0.6% (ConFUSE).

**Impact of Aggregation Methods:** We evaluate the impact of aggregation rules on the attack effectiveness. In addition to the standard FedAvg, we also consider two robust aggregation methods: Trimmed Mean (Trim) [14] and Median [46], during local unlearning aggregation. As shown in Figures 8 and 9, Trim slightly reduces attack performance but still permits successful backdoor injection, with BA dropping by $2.8\%$, $0.1\%$, and $0.9\%$ under NoT, FedRR, and ConFUSE, while MA decreases by $1.1\%$, $2.1\%$, and $1.3\%$. The relatively larger BA drop for NoT may result from its retraining-based nature, which yields more uniform updates and makes them more susceptible to trimming. However, its MA is least affected, showing NoT's robustness in preserving main-task performance. In contrast, Median significantly suppresses the attack, with BA dropping below $15\%$ across methods, but at the cost of main-task performance, especially under ConFUSE. It is known [16] that Median can be robust to backdoor attacks but is not readily deployable as it tends to exclude benign-but-outlier client updates from aggregation and is highly vulnerable to non-IID settings. This limitation is critical in active FU, where essential unlearning updates from target clients may deviate from the majority and thus be disregarded. Moreover, data removal can further exacerbate the non-IID situation, making Median even less suitable.

## IV. Design and Evaluation of SubID for Active FU

To defend against FUsion backdoor in active FU, we note that existing defenses in conventional FL, such as robust aggregation [14], gradient clipping [47], [48], or update filtering [17], [18], are generally ineffective in this setting. These defenses are designed for standard FL without data removal and typically rely on stable training and consistent client behavior. In FU, however, clients remove parts of their data, and many unlearning methods introduce additional correction objectives or procedures [5], [7], causing both target and remaining clients' updates to deviate significantly due to data shifts and historical constraints, even without adversarial behavior.

A recent study [11] proposed a defense for passive FU that filters potentially malicious updates by estimating each client's local updates before aggregation. While applied to active FU, this method is less effective, as the server cannot access the unlearned data of target clients. Without this information, the server cannot reliably estimate expected local updates, leading to less accurate detection and potentially undermining the unlearning performance.

To address these challenges, we introduce SubID, which leverages the global behavioral consistency of client updates. While benign clients may exhibit diverse directions during unlearning due to data removal or correction objectives, such deviations are typically unstructured. In contrast, malicious clients, typically using the same trigger and jointly optimizing for a shared target, tend to produce updates aligned with a consistent subspace linked to trigger sensitivity. Moreover, strong FL defenses during training often prevent early-stage poisoning, forcing adversaries to launch attacks exclusively in the unlearning phase. By identifying the dominant deviation subspace of unlearning updates relative to the training phase, we assess each client's alignment to detect malicious behavior.

### A. The SubID Defense Design

The proposed SubID method spans both the training and unlearning phases. During training, the server records a window of local updates from each client over the last $T_r$ rounds, forming a matrix $\mathcal{G}_{\text{train}} \in \mathbb{R}^{d\times(n\cdot T_r)}$, where $d$ is the number of model parameters. We perform singular value decomposition (SVD) [49] on $\mathcal{G}\text{train} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_d] \in \mathbb{R}^{d\times d}$ contains the left singular vectors, $\mathbf{\Sigma}$ is the diagonal matrix of singular values, and $\mathbf{V}$ the right singular vectors.

The top-$k$ vectors in $\mathbf{U}$ define the principal directions of benign update behaviors, which capture the dominant variation patterns of client updates during the final training stage, when the model has largely converged and benign clients exhibit stable optimization behavior. We use the corresponding subspace $\mathcal{S}_{\text{benign}} = \text{span}(\mathbf{u}_1, \ldots, \mathbf{u}_k)$ as a reference for identifying abnormal directional shifts in future updates.

In the unlearning phase, we divide the rounds into consecutive windows of size $\tau$. For the $a$-th window $(a = 0, 1, 2, \ldots)$, we monitor each client's unlearning updates $\{\hat{\mathbf{g}}_i^t\}$ for $i \in \mathcal{C}$ and $t \in [a\tau + 1,\ (a+1)\tau]$. For each update $\hat{\mathbf{g}}_i^t$, we compute its residual $\boldsymbol{\delta}_i^t$ with respect to the benign subspace $\mathcal{S}_{\text{benign}}$:

$$\boldsymbol{\delta}_i^t = \hat{\mathbf{g}}_i^t - \mathbf{P}_{\mathcal{S}_{\text{benign}}}(\hat{\mathbf{g}}_i^t), \tag{13}$$

where $\mathbf{P}_{\mathcal{S}_{\text{benign}}}(\cdot)$ denotes the orthogonal projection onto $\mathcal{S}_{\text{benign}}$. The residual $\boldsymbol{\delta}_i^t$ thus captures the component of the update that deviates from normal training behavior. We then stack all residuals from the current window into a matrix $\mathbf{D}_a \in \mathbb{R}^{d\times(n\cdot\tau)}$ for further analysis.

Although unlearned data removal may cause update deviations, they are typically unstructured and do not introduce new features, as clients still optimize the same primary task. In contrast, malicious clients inject backdoor triggers that introduce new feature directions. As they often use the same

trigger and target, and jointly optimize for backdoor objectives, the resulting deviations exhibit strong directional consistency.

To uncover dominant patterns in these deviations, we perform SVD on $\mathbf{D}^a$ and extract the top-$k$ left singular vectors $\{\mathbf{u}_1^\Delta, \ldots, \mathbf{u}_k^\Delta\}$, which capture the most coherent directional shifts across clients in the $a$-th window. These vectors define the deviation subspace $\mathcal{S}_{\text{dev}}^a := \text{span}(\mathbf{u}_1^\Delta, \ldots, \mathbf{u}_k^\Delta)$.

We then compute a projection score $\phi_i^a$ for each client $i$ in the $a$-th window, based on the mean of its updates $\hat{\mathbf{g}}_i$:

$$\phi_i^a = \left\| \mathbf{P}_{\mathcal{S}_{\text{dev}}^a}(\hat{\mathbf{g}}_i) \right\|_2 . \tag{14}$$

This score reflects the alignment of client $i$'s behavior with the dominant deviation directions in window $a$.

To detect abnormal clients in each window, we apply the Median Absolute Deviation (MAD) [50] rule to the set of scores $\{\phi_i^a\}_{i \in \mathcal{C}}$. A client is flagged as suspicious if:

$$\phi_i^a > \text{median}(\{\phi_i^a\}) + \kappa \cdot \text{MAD}(\{\phi_i^a\}), \tag{15}$$

where $\kappa$ is a tunable threshold and $\text{MAD}(\{\phi_i^a\}) = \text{median}(\{|\phi_i^a - \text{median}(\{\phi_i^a\})|)$.

All clients, including those flagged as suspicious, continue to participate in model aggregation to preserve the functionality of the unlearning process. At the end of unlearning, we count the number of windows in which each client was flagged. Clients with total suspicious counts exceeding a predefined threshold $\phi$ are classified as malicious. If any malicious clients are detected, we discard the current unlearning result and initiate full retraining to recover a clean model.

Our method exploits the directional consistency of shared-target backdoor attacks and the benign behavior enforced by FL defenses during training. Even with diverse trigger patterns, a common target label leads to aligned optimization objectives and coherent update deviations. If attackers are aware of SubID, they still have no choice but to launch the attack during unlearning due to strong FL defenses. However, doing so inevitably introduces backdoor-consistent deviations that SubID can detect, making adaptive attacks ineffective.

### *B. Experiment*

We set up experiments to evaluate our SubID mechanism against backdoor attacks in active FU.

*1) Experiment Setup:* The experimental setup follows the attack configurations in Section III-D. We further compare our method with two existing defenses UnlearnGuard-Dist (Dist) and UnlearnGuard-Dir (Dir) [11]. While the attack in [11] does not apply to active FU, Dist and Dir can be used for malicious client detection in active FU. We report True Positive Rate (TPR), False Positive Rate (FPR), and F1 Score for evaluation. Default parameters in SubID are $k = 5$, $\kappa = 7$, and $\phi = 0.5$; other settings follow Section III-D.

*2) Experimental Results:* **Comparison with Existing FU Defenses:** We compare SubID with Dist and Dir in terms of their ability to detect malicious clients. As shown in Figure 10, SubID achieves an F1 score of 1.0 across all three active FU methods, indicating accurate and complete detection performance with no detection errors. In contrast, Dist and Dir yield F1 scores below 0.4, suggesting poor detection accuracy. This may be because they were originally designed for passive FU to filter suspicious clients for utility preservation, rather than to explicitly and accurately detect malicious behavior. These results highligh the limitations of passive FU defenses in active settings and the effectiveness of SubID.

**Evaluations across Datasets, Models, and Trigger Patterns:** We evaluate our SubID defense on multiple datasets with different trigger patterns and model architectures. As shown in Table II, the method achieves perfect detection on MNIST and CIFAR10 (TPR = 1.0, FPR = 0), indicating accurate and complete identification of malicious clients.

TABLE II: Performance under different datasets.

| FU Method | NoT | | FedRR | | ConFUSE | |
|---|---|---|---|---|---|---|
| Dataset | TPR | FPR | TPR | FPR | TPR | FPR |
| MNIST | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 |
| CIFAR10 | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 |
| HAR | 1.00 | 0.03 | 1.00 | 0.00 | 0.80 | 0.03 |
| AG News | 1.00 | 0.05 | 1.00 | 0.05 | 0.80 | 0.08 |

On HAR and AG News, performance slightly degrades, particularly under ConFUSE, where TPR drops to 0.8 and FPR increases to 0.03 and 0.08. Similar issues are seen in NoT and FedRR. As discussed earlier, these datasets may not reach stable training within limited unlearning rounds, and ConFUSE tends to show higher update instability, which may affect detection. Using dataset-specific $\phi$ and $\kappa$ values could further improve performance.

**Effect of Threshold Parameter $\kappa$:** We analyze how the sensitivity parameter $\kappa$ affects the MAD-based detection threshold. As shown in Figure 11, SubID maintains a TPR of 1.0 for $\kappa$ from 1 to 8, indicating that backdoor injection significantly alters the deviation subspace, making malicious clients' projection scores clearly distinguishable. However, when $\kappa > 8$, the TPR begins to drop, falling to 0.5 under FedRR at $\kappa = 10$. This is because a larger $\kappa$ leads to a more conservative threshold, which may fail to capture malicious clients whose updates produce more subtle deviations. FPR is also influenced by $\kappa$. When $\kappa = 1$, it reaches 0.125, as only extremely consistent scores are classified as benign, leading to false positives. As $\kappa$ increases, the threshold becomes more tolerant, reducing the chance of misclassifying benign clients. This trade-off underscores the importance of selecting an appropriate $\kappa$, with values between 4 and 8 empirically offering a good balance between sensitivity and robustness.

**Impact of Detection Frequency Threshold $\phi$:** We evaluate how the detection frequency threshold $\phi$ affects final defense performance. Specifically, $\phi$ defines the minimum fraction of rounds in which a client must be flagged to be deemed malicious. As shown in Figure 12, all three FU methods maintain TPR = 1.0 when $\phi$ is low, demonstrating that occasional deviations from benign clients may lead to false positives when $\phi$ is too low. As relying on a single round of detection may cause some benign clients to be mistakenly removed due to occasional deviations. Once $\phi$ exceeds 0.3, the FPR drops to zero, showing that false positives typically stem from occasional benign deviations at low $\phi$. However, a higher

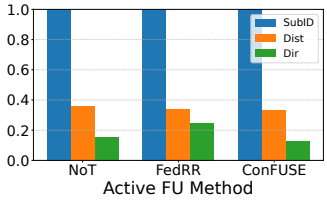


Fig. 10: F1 score of malicious client detection.

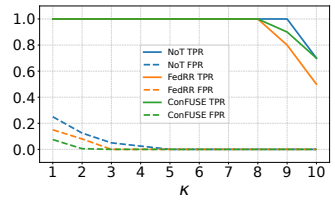


Fig. 11: Impact of the $\kappa$ on TPR and FPR.

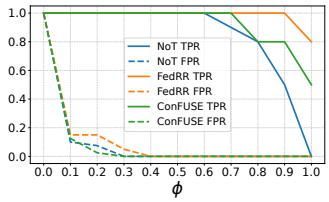


Fig. 12: Impact of the $\phi$ on TPR and FPR.

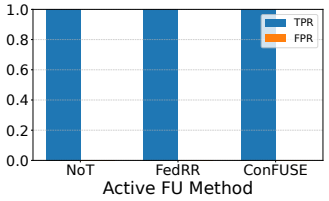


Fig. 13: Detection performance under non-IID settings.

TABLE III: Detection performance under different defenses.

| FU Method | NoT | | FedRR | | ConFUSE | |
|---|---|---|---|---|---|---|
| Attack Method | TPR | FPR | TPR | FPR | TPR | FPR |
| w/o attack | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.05 |
| Backdoor | 1.00 | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 |
| Trim | 0.70 | 0.00 | 1.00 | 0.00 | 1.00 | 0.00 |
| Sign Flip | 0.20 | 0.00 | 0.00 | 0.00 | 0.20 | 0.00 |

value of $\phi$ also reduces TPR. In particular, the TPR of FedRR or ConFUSE drops from 1.0 ($\phi$=0.7) to 0.8 or 0.5 ($\phi$=1.0), while NoT drops sharply to 0. This indicates that malicious clients are not detected every time. The sharp decline in NoT may result from its retraining-based updates, which smooth client behaviors over time and reduce sensitivity to consistent deviation patterns. These results highlight the importance of multi-round detection over single-round judgments.

**Detection Robustness under Non-IID Data:** We evaluate the robustness of SubID under non-IID client data. Figure 13 shows that our method consistently achieves TPR = 1 and FPR = 0 across all FU methods, indicating that even in the presence of heterogeneous data distributions, the backdoor behavior introduces strong directional deviations that remain highly separable in the extracted subspace.

**Robustness under Non-Adversarial and Untargeted Attacks:** While previous results demonstrate strong detection against backdoor attacks, real-world active FU systems may also involve untargeted or no attacks at all. To evaluate robustness, we test SubID under three additional scenarios: no attack, Trim attack [14], and Sign Flip attack [51].

Table III shows that SubID performs well in the no-attack case, with only ConFUSE showing a few false detections (FPR = 0.05). This may be due to ConFUSE using only a small portion of retained data for optimization, resulting in limited updates that may appear as outliers in the deviation subspace.

Under Trim attack, SubID remains effective on FedRR and ConFUSE, but TPR drops to 0.7 on NoT. This is likely because Trim introduces subtle manipulations without significantly altering direction, making malicious updates less distinguishable from benign ones in deviation-based projection space, especially under NoT, where retraining can further smooth such differences. In contrast, Sign Flip causes a sharp performance drop across all methods. This is likely because the attack flips malicious updates, introducing extreme, consistent perturbations that significantly increase the overall variation in the projection scores of all clients, including benign ones. As a result, the median and MAD used for thresholding are also inflated, loosening the detection boundary and making it harder to separate malicious clients.

## V. Related Work

**Backdoor Attacks in FL:** Backdoor attacks in FL aim to implant malicious behaviors into the global model that are activated only by specific trigger inputs. Early work [20] showed that local model poisoning with trigger-embedded data can effectively manipulate model predictions. Subsequent works have explored various strategies to enhance the stealth and effectiveness of backdoor attacks, including gradient manipulation [15], sparsity-based or trigger-sensitive pruning [30], and adaptive participation scheduling [52]. To counter such threats, defenses such as robust aggregation [13], [14], [46] and client-level filtering [12], [16], [17] have been proposed, but they are designed for and applied only during the training phase. Our work activates the backdoor exclusively during the unlearning process, thereby bypassing training-phase defenses while preserving both stealth and attack effectiveness.

**Security Risks in FU:** Active FU enables clients to erase the impact of their data [3], [23], [26], usually assuming they are trusted participants. In contrast, passive FU assumes the server initiates unlearning and target clients leave the system to ensure their data is excluded from future training [7], [24], [25]. Despite the growing adoption of FU techniques, their security remains underexplored. Most existing methods lack dedicated protection mechanisms, leaving them vulnerable to manipulation by malicious clients. A recent study [11] highlights poisoning risks in passive FU and proposes defenses based on update prediction. However, these methods approximate expected behavior and cannot reliably identify malicious clients. More importantly, their filter-based design is incompatible with active FU, where target clients must participate to ensure effective data removal. If their updates are filtered out, the unlearning objective cannot be properly fulfilled. We systematically demonstrate and defend against backdoor injection attacks targeting active FU, offering a principled protection mechanism under this challenging setting.

## VI. Conclusion

In this paper, we propose and evaluate FUsion backdoor, a fast and effective attack that leverages subnetwork-level manipulation for efficient backdoor injection during unlearning. We then design SubID, a subspace-based defense that accurately identifies malicious clients and removes them from the system. Experimental evaluations show that FUsion backdoor exposes a critical security issue in active FU frameworks and SubID offers an effective solution to combating such an attack.

**Acknowledgement:** The work was supported in part by NSF Grants 2321270 and 2319781 at USF and 2505381 at UF.

## 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 *AISTATS*. PMLR, 2017, pp. 1273–1282.

[2] J. Chen and D. Yang, "Unlearn what you want to forget: Efficient unlearning for llms," *arXiv preprint arXiv:2310.20150*, 2023.

[3] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *Proc. IEEE INFOCOM*. IEEE, 2022, pp. 1749–1758.

[4] N. Su and B. Li, "Asynchronous federated unlearning," in *Proc. IEEE INFOCOM*. IEEE, 2023, pp. 1–10.

[5] Z. Liu, Y. Jiang, J. Shen, M. Peng, K.-Y. Lam, X. Yuan, and X. Liu, "A survey on federated unlearning: Challenges, methods, and future directions," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–38, 2024.

[6] Y. Lin, Z. Gao, H. Du, D. Niyato, G. Gui, S. Cui, and J. Ren, "Scalable federated unlearning via isolated and coded sharding," *arXiv preprint arXiv:2401.15957*, 2024.

[7] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "FedRecover: Recovering from poisoning attacks in federated learning using historical information," in *S&P*, 2023, pp. 1366–1383.

[8] X. Zhu, G. Li, and W. Hu, "Heterogeneous federated knowledge graph embedding learning and unlearning," in *Proc. ACM Web Conf. (WWW)*, 2023, pp. 2444–2454.

[9] Y. Zhao, P. Wang, H. Qi, J. Huang, Z. Wei, and Q. Zhang, "Federated unlearning with momentum degradation," *IEEE Internet of Things Journal*, 2023.

[10] A. Dhasade, Y. Ding, S. Guo, A.-M. Kermarrec, M. D. Vos, and L. Wu, "Quickdrop: Efficient federated unlearning by integrated dataset distillation," *arXiv preprint arXiv:2311.15603*, 2023.

[11] W. Wang, Q. Ma, Z. Zhang, Y. Liu, Z. Liu, and M. Fang, "Poisoning attacks and defenses to federated unlearning," in *Proc. ACM Web Conf.*, 2025, pp. 1365–1369.

[12] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *KDD*, 2022, pp. 2545–2555.

[13] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *NDSS*, 2021.

[14] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *USENIX Security*, 2020, pp. 1605–1622.

[15] H. Zhang, J. Jia, J. Chen, L. Lin, and D. Wu, "A3FL: Adversarially adaptive backdoor attacks to federated learning," *NeurIPS*, vol. 36, 2024.

[16] P. Rieger, T. Krauß, M. Miettinen, A. Dmitrienko, and A.-R. Sadeghi, "CrowdGuard: Federated backdoor detection in federated learning," *arXiv preprint arXiv:2210.07714*, 2022.

[17] T. Krauß and A. Dmitrienko, "MESAS: Poisoning defense for federated learning resilient against adaptive attackers," in *CCS*, 2023, pp. 1526–1540.

[18] G. Yan, H. Wang, X. Yuan, and J. Li, "DEFL: Defending against model poisoning attacks in federated learning via critical learning periods awareness," in *AAAI*, vol. 37, no. 9, 2023, pp. 10 711–10 719.

[19] N. Romandini, A. Mora, C. Mazzocca, R. Montanari, and P. Bellavista, "Federated unlearning: A survey on methods, design guidelines, and evaluation metrics," *IEEE Trans. Neural Netw. Learn. Syst.*, 2024.

[20] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*. PMLR, 2020, pp. 2938–2948.

[21] E. Kabir, Z. Song, M. R. U. Rashid, and S. Mehnaz, "FLShield: A validation-based federated learning framework to defend against poisoning attacks," in *S&P*. IEEE, 2024, pp. 2572–2590.

[22] H. Xia, S. Xu, J. Pei, R. Zhang, Z. Yu, W. Zou, L. Wang, and C. Liu, "FedME 2: Memory evaluation & erase promoting federated unlearning in DTMN," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3573–3588, 2023.

[23] Y. H. Khalil, L. Brunswic, S. Lamghari, X. Li, M. Beitollahi, and X. Chen, "Not: Federated unlearning via weight negation," in *CVPR*, 2025, pp. 25 759–25 769.

[24] Y. Jiang, J. Shen, Z. Liu, C. W. Tan, and K.-Y. Lam, "Towards efficient and certified recovery from poisoning attacks in federated learning," *IEEE Transactions on Information Forensics and Security*, 2025.

[25] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," in *IWQoS*, 2021, pp. 1–10.

[26] H. Gu, G. Zhu, J. Zhang, X. Zhao, Y. Han, L. Fan, and Q. Yang, "Unlearning during learning: An efficient federated machine unlearning method," *arXiv preprint arXiv:2405.15474*, 2024.

[27] H. Wang, X. Zhu, C. Chen, and P. Esteves-Veríssimo, "Goldfish: An efficient federated unlearning framework," in *DSN*. IEEE, 2024, pp. 252–264.

[28] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*. Internet Society, 2018.

[29] X. Qi, T. Xie, R. Pan, J. Zhu, Y. Yang, and K. Bu, "Towards practical deployment-stage backdoor attack on deep neural networks," in *CVPR*, 2022, pp. 13 347–13 357.

[30] Z. Yin, Y. Yuan, P. Guo, and P. Zhou, "Backdoor attacks on federated learning with lottery ticket hypothesis," *arXiv preprint arXiv:2109.10512*, 2021.

[31] H. Zhuang, M. Yu, H. Wang, Y. Hua, J. Li, and X. Yuan, "Backdoor federated learning by poisoning backdoor-critical layers," *arXiv preprint arXiv:2308.04466*, 2023.

[32] 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 *S&P*. IEEE, 2019, pp. 707–723.

[33] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," *NeurIPS*, vol. 31, 2018.

[34] J. Yang and Y. Zhao, "A survey of federated unlearning: A taxonomy, challenges and future directions," *arXiv preprint arXiv:2310.19218*, 2023.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *ICCV*, 2015, pp. 1026–1034.

[36] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017, pp. 618–626.

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

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

[39] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz *et al.*, "A public domain dataset for human activity recognition using smartphones," in *ESANN*, vol. 3, no. 1, 2013, pp. 3–4.

[40] F. Ilhan, G. Su, and L. Liu, "Scalefl: Resource-adaptive federated learning with heterogeneous clients," in *CVPR*, 2023, pp. 24 532–24 541.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[43] R. Mutegeki and D. S. Han, "A cnn-lstm approach to human activity recognition," in *ICAIIC*. IEEE, 2020, pp. 362–366.

[44] M. Umer, Z. Imtiaz, M. Ahmad, M. Nappi, C. Medaglia, G. S. Choi, and A. Mehmood, "Impact of convolutional neural network and FastText embedding on text classification," *Multimedia Tools Appl.*, vol. 82, no. 4, pp. 5569–5585, 2023.

[45] S. I. A. Meerza, A. Sadovnik, and J. Liu, "Confuse: Confusion-based federated unlearning with salience exploration," in *ISVLSI*. IEEE, 2024, pp. 427–432.

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

[47] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, "Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification," in *AISTATS*. PMLR, 2022, pp. 7587–7624.

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

[49] M. Moonen, P. V. Dooren, and J. Vandewalle, "A singular value decomposition updating algorithm for subspace tracking," *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 4, pp. 1015–1038, 1992.

[50] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *J. Exp. Soc. Psychol.*, vol. 49, no. 4, pp. 764–766, 2013.

[51] A. Sharma and N. Marchang, "Probabilistic sign flipping attack in federated learning," in *ICCCNT*. IEEE, 2024, pp. 1–6.

[52] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *NeurIPS*, vol. 32, 2019.