SafeSplit: A Novel Defense Against Client-Side Backdoor Attacks in Split Learning

Phillip Rieger
Technical University of Darmstadt

Alessandro Pegoraro
Technical University of Darmstadt

Kavita Kumari
Technical University of Darmstadt

Tigist Abera
Technical University of Darmstadt

Jonathan Knauer
Technical University of Darmstadt

Ahmad-Reza Sadeghi
Technical University of Darmstadt

Abstract—Split Learning (SL) is a distributed deep learning approach enabling multiple clients and a server to collaboratively train and infer on a shared deep neural network (DNN) without requiring clients to share their private local data. The DNN is partitioned in SL, with most layers residing on the server and a few initial layers and inputs on the client side. This configuration allows resource-constrained clients to participate in training and inference. However, the distributed architecture exposes SL to backdoor attacks, where malicious clients can manipulate local datasets to alter the DNN's behavior. Existing defenses from other distributed frameworks like Federated Learning are not applicable, and there is a lack of effective backdoor defenses specifically designed for SL.

We present SafeSplit, the first defense against client-side backdoor attacks in Split Learning (SL). SafeSplit enables the server to detect and filter out malicious client behavior by employing circular backward analysis after a client's training is completed, iteratively reverting to a trained checkpoint where the model under examination is found to be benign. It uses a two-fold analysis to identify client-induced changes and detect poisoned models. First, a static analysis in the frequency domain measures the differences in the layer's parameters at the server. Second, a dynamic analysis introduces a novel rotational distance metric that assesses the orientation shifts of the server's layer parameters during training. Our comprehensive evaluation across various data distributions, client counts, and attack scenarios demonstrates the high efficacy of this dual analysis in mitigating backdoor attacks while preserving model utility.

I. INTRODUCTION

Recently, deep neural networks (DNNs) have made significant advances, leading to the development of new training frameworks such as Large Language Models (LLMs)[1], AI-based image generation, and image recognition for self-driving cars. Concurrently, the complexity of deployed DNNs has grown rapidly to manage the ever-increasing tasks, demanding more

[1]Although transformers and diffusion models differ significantly from traditional DNNs, they still consist of layers and trainable parameters, making them suitable for distributed learning.

robust computational resources. This increasing complexity presents a substantial challenge for deploying such advanced DNNs on resource-constrained devices without compromising the privacy of potentially sensitive input data.

Split Learning (SL) is a class of distributed learning that promises to reduce the computational load on the client side without requiring the clients to share their data. In this paradigm, the DNN's architecture is split between client and server, with the computationally intensive layers outsourced to a server [52], [20]. Thus, it is a resource-friendly collaboration between the clients and the server, unlike federated learning [30], [18], [23]. The strengths of Split Learning have been demonstrated on real-world data in medical contexts [62]. Further potential applications include financial services and AI-based consumer services such as image editing on mobile devices, analogously to Federated Learning that is widely deployed as part of GBoard [34]. In the past, different configurations for SL were developed, with the two most prevalent being the vanilla and U-shaped configurations [18], [49], [24], [6], [66]. In the vanilla configuration [49], [67], the DNN is split into two segments at a specific "cut layer." The smaller portion of the network resides on the client side, while the larger part is on the server side. In the U-shaped configuration [67], [18], the DNN is split into three segments. The head segment, up to the first "cut layer", and the tail segment, starting from the second "cut layer", reside on the client side. The middle segment called the backbone, is outsourced to the server because it is usually composed of most of the layers and, therefore, is computationally more intensive to train.

Training in SL proceeds sequentially, with clients queuing for sessions. In the vanilla setup, each client trains its part of the network up to the cut layer, transmitting outputs to the server, which completes the training. In U-shaped systems, the client model evaluates data up to the first cut layer during forward propagation before sending the feature vectors to the server. The server processes these vectors and returns the output for the client to complete using the tail. During backpropagation, the client calculates the loss and sends gradients to the server, which computes the backbone's gradients and sends cut layer gradients back to the client. This setup allows

resource-constrained devices to train large models securely and privately [52], [46], complying with data privacy standards like GDPR [2] and HIPAA [1]. However, this setup also increases the attack surface and results in a stronger threat model, since the adversary can arbitrarily manipulate the loss. Thus, the rest of the paper focuses on the U-shape configuration.

Attacks on SL. Recent works demonstrated the vulnerability of SL to multiple attacks, including data reconstruction attacks [33], label inference attacks [38], [15], and backdoor attacks [6], [22], [63]. An adversary can launch a backdoor attack on the server-side [13], [64] or the client-side [63], [22].

This paper focuses on client-side backdoor attacks, as clients are more susceptible to attacks than well-protected servers. Additionally, malicious clients in SL have an advantage due to their access to data and labels, making it crucial for the server to defend against such attacks to protect benign clients. However, an efficient defense in SL is challenging because each client uses the trained model of its predecessor as a base, resulting in different starting models for each client. If a previous client was malicious, subsequent benign clients may unknowingly train on a poisoned model, compromising also their results.

Although in the past various defenses were proposed to mitigate attacks in other distributed learning paradigms, such as Federated Learning [14], [17], [8], [5], [41], [11], they are not applicable in SL due to the aforementioned sequential training structure.

Our goal and contributions. To address the challenge of client-side backdoor attacks in SL, we introduce SafeSplit, a versatile and, to the best of our knowledge, the first backdoor defense for the U-shaped SL paradigm deployed at the server. SafeSplit operates on a rollback mechanism to employ circular backward analysis after a client's training ends, reverting to a trained state where the client under examination emerges as benign. That is, if the trained backbone at the server shows backdoor characteristics, this rollback mechanism reverts to a previous client's trained state and continues the examination to ensure a benign trained state is selected, bypassing all the malicious client's trained updates.

To identify poisoned training contributions and detect malicious behavior, SafeSplit employs a two-fold strategy to perform the static and dynamic analysis of the server's backbone during the rollback mechanism. These metrics are based on the rationale that benign behavior contradicts the mispredictions caused by backdoor behavior. Therefore, significant changes need to be applied by the attacker to the model to introduce new behavior into the model. First, the motivation to conduct the static analysis of the server's backbone is to measure static characteristics of the backbone's parameters, such as the changes in the frequency domain, without considering changes over time during training. An analogy for this strategy would be analyzing a music recording to see how often certain notes (frequencies) appear without considering how the music changes over time. Here, we are only interested in the presence of the frequency of notes, not in the sequence or evolution of the music. Second, the motivation to conduct the dynamic

analysis is to assess how the orientation or configuration of the backbone's parameters changes throughout training. We introduce a novel dynamic rotational distance metric that measures the extent of dynamic shifts in the backbone's values or configurations, providing insights into how the values evolve during training. Thus, the rotational distance metric analyzes dynamic aspects, such as the flow and transition of the backbone values. Using the music analogy, the metric measures the transitions between the notes rather than their bare presence. Therefore, SafeSplit uses both perspectives to analyze the clients' models and detect backdoors. Using this analysis framework, we create a first novel robust defense against the backdoor attacks in SL.

Two important things to note in the design of SafeSplit: Firstly, we do not permanently remove malicious clients from consideration. Instead, we skip the models of these clients, allowing them to be reconsidered in subsequent training epochs to ensure that misclassified benign clients are not unjustly removed. Secondly, we deploy SafeSplit before a client starts its training to select a benign starting point and ensure that poisoned training contributions are effectively mitigated and not used to train benign models.

In summary, our contributions include:

- We propose SafeSplit, the first defense framework designed to mitigate backdoor attacks in Split Learning (SL). SafeSplit accurately detects backdoor attacks and reduces their impact while minimizing harmful effects on the models' utility. We conduct static and dynamic analyses to inspect nuances in the applied model updates (Sect. V-A).
- Our approach addresses the challenges introduced by SL's sequential structure through a circular defense mechanism applied after each client's training process. It enables the early detection of malicious client behavior and mitigates its impact before it influences the training of other benign clients (Sect. V-B).
- We introduce a novel rotational distance metric that measures, based on that angular displacement, how the orientation or configuration of the backbone's value at the server changes throughout training (Sect. V-D). This metric also captures dynamic nuances such as the orientation or rotation of the changes that provide additional information about learning objectives during the training (Sect. V-D).
- We perform a deep analysis of the static changes introduced through the local training by analyzing the model updates in the frequency domain (Sect. V-C).
- To evaluate the effectiveness of our defense, we developed various backdoor attacks, including poisoning and semantic backdoor attacks. These attacks were applied to datasets such as CIFAR-10, FMNIST, MNIST, CIFAR-100, and GTSRB across different numbers of clients, attack settings, and data scenarios. Our extensive evaluation demonstrates the effectiveness of SafeSplit even against defense-adapted attacks (Sect. VI).
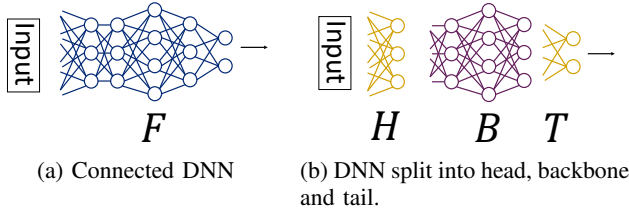
(a) Connected DNN   (b) DNN split into head, backbone and tail.

Fig. 1: Comparison of splitting the Deep Neural Network (DNN) F into head (H), backbone (B), and tail (T), such that $F \equiv H \circ B \circ T$. The head and tail are located on the client side, and the backbone is on the server side.

With SafeSplit, we make the first step towards solving an open challenge of mitigating client-side backdoor attacks on SL, significantly reducing the backdoor impact while maintaining the accuracy of the resulting model. We hope that future research continues building on top of our scheme.

II. BACKGROUND

A. Split Learning

In the U-shaped Split Learning paradigm, N different participants $C_1, \ldots, C_N$ jointly train a DNN coordinated by a central server. As shown in Figure 1, the typical architecture of a connected DNN (Fig. 1a) is split into three sub-models, head (H), backbone (B) and tail (T) (Fig. 1b). Next, we detail some of the definitions utilized in the framework of SL.

Head (H) resides on the client side and receives input data D directly from the client's local dataset. The output of the head, the smashed data, is sent to the server during forward propagation. Later, the client receives the gradients from the server to complete the backpropagation and update the head.

Backbone (B) forms the central part of the model and resides on the server side. It accepts the smashed data from the head during forward propagation, further processes it, and sends it to the T. During backward propagation, it accepts gradients from T, continues the processes, and sends its output to the H to complete the training.

Tail (T) is the last part of the model, residing on the client side. It receives intermediate features from the backbone and refines them to generate the final output predictions and backpropagates gradients to the server.

The U-shaped paradigm makes use of the function composition property that allows obtaining the exact behavior of $F(X)$ through the implicit concatenation of H, B, and T, as shown below:

$$(T \circ B \circ H)(X) \equiv F(X)$$

Given that the composition of functions is always associative [51], we can show that feeding the head output as backbone input and giving the backbone output as tail input still represents the same behavior as the original DNN that the clients want to train.

$$T((B \circ H)(X)) = (T \circ B \circ H)(X)$$

In a standard training scenario, as depicted in Figure 2, client i passes its data through its head (step 1), then gives the output of the head to the server (step 2), passes it through its backbone, and sends the output back to client i (step 3), who feeds this last output to its tail and calculates the loss using the ground-truth labels (step 4). During backpropagation (step 5), the client first calculates the gradients for the tail and passes the computed gradients to the server, which again computes the gradients for the backbone using the information provided by client i. The server then applies backpropagation and passes the gradients to client i, who applies backpropagation to obtain the gradients for the head. After these steps, client i shares the resulting tail and head with client $i + 1$ (step 6), who uses them as starting weights for its training step. As the server does not have direct access to the client's head and tail nor to the client's data, SL enables resource-constrained devices to train and apply large models in a secure and privacy-preserving [52], [46], [20].

III. PROBLEM SETTING

In this section, we describe the considered system (Sect. III-A) and characterize the threat model (Sect. III-B), before describing inherent challenges in SL in mitigating backdoor attacks (Sect. III-C). In the appendix, we provide a high-level overview of DNNs (App. A), poisoning attacks (App. B), and the eligibility of frequency transformations to detect backdoors (App. C).

A. System Setting

In the rest of this paper, we consider a system consisting of N clients holding private datasets that are not shared with other parties. Coordinated by a central server $\mathcal{S}$, they use the SL framework to train a DNN on their private datasets collaboratively. An example of such a system is visualized in Fig. 2. Following existing literature [18], [32], [39], [67], [27], [41], we focus on a U-shaped SL configuration that splits the DNN into three parts (Head - H, Backbone - B, Tail - T), where H and T are located on the client side, whereas B is executed on the server, as described in Sect. II-A. Since the last part of the DNN (T) is on the client side, the clients are also responsible for the loss calculation. Once the training is finished, the client signs the current model and forwards it with the previous clients' models to the next client for the following training iterations.

For example, the clients could be low-performance devices such as smartphones that want to train and perform inference on a large DNN (e.g., a Large Language Model) without revealing their private training or inference data. The U-shape enables them to outsource the computation-intensive part to a high-performance server without violating the data's privacy.

B. Adversary Model

We consider an adversary $\mathcal{A}$ that aims to inject a backdoor into the collaboratively trained model F. Thus, the backdoored model F^* shall predict a specific, adversary-chosen target label $L_{\mathcal{A}}$ when the model receives input x^* containing the trigger R.
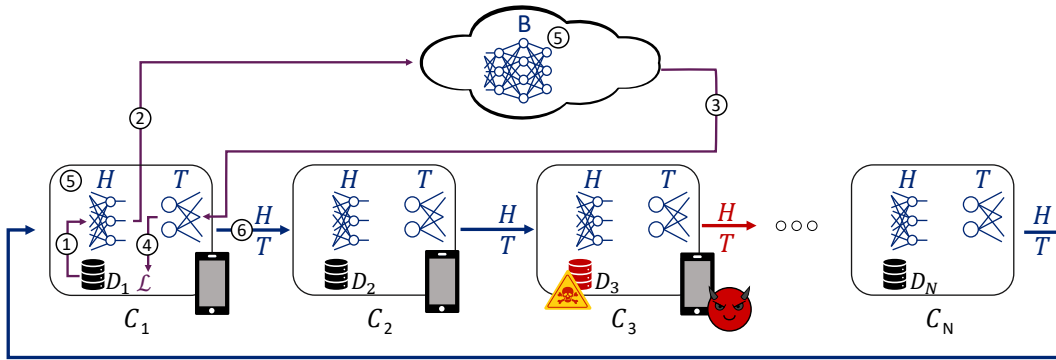
Fig. 2: Overview of a Split Learning (SL) system that utilizes data from mobile devices but executes the computation-heavy backbone (B) on a cloud server, while all clients $C_1, \ldots C_N$ provide the data D_i, hosts the head T and tail T, as well as calculates the loss $\mathcal{L}$.

When given a clean input without the trigger, the backdoored model must generate the correct prediction to prevent the backdoor from being detected. More formally:

$$F^*(x) = \begin{cases} L_{\mathcal{A}} & R \in x \\ F(x) & R \notin x \end{cases} \quad (1)$$

Notably, for a triggered input sample x^*, the clean prediction differs from the backdoor target label $F(x^*) \neq L_{\mathcal{A}}$.

$\mathcal{A}$ is assumed to have complete control over one or several malicious clients and can, therefore, arbitrarily manipulate the input data and labels. Further, due to the considered U-shape architecture, $\mathcal{A}$ can arbitrarily change the local head and tails, the smashed data and gradients sent by the server, and the loss function. Aligned with existing work on mitigating backdoor attacks in distributed learning [43], [8], [25], we assume $\mathcal{A}$ to control at most $N/2 - 1$ clients. Additionally, we assume $\mathcal{A}$ knows the defense mechanism deployed on the server side. Thus, $\mathcal{A}$ can constrain the training loss utilizing the metrics used by the defense mechanism.

In the following, we will focus on the attacks that malicious clients perform. The server has an intrinsic motivation to produce a well-trained and effective model, as its reputation is based on the quality of the resulting model. Additionally, while the client devices are mostly anonymous, as they are just mobile devices, the server is identifiable and accountable for its behavior. Since existing literature already investigated the problem of defending against backdoor attacks that are conducted by malicious servers [13], we will consider these attacks to be out of the scope of this paper.

C. Objectives and Challenges

This section details the objectives of a defense that aspires to prevent backdoor attacks and the challenges encountered in designing such a defense in the framework of SL.

An effective and practical backdoor defense aims to fulfill the following security objective:

O1 Prevent Backdoor Attacks: The primary requirement of an effective defense strategy is to efficiently prevent malicious

clients from injecting a backdoor (as described in App. B) into the trained model.

However, to ensure that the defense is practical and does not render the resulting model unusable, the defense must also fulfill the following functional requirement:

O2 Preserve Model's Utility: The defense must not negatively affect the accuracy of the model on the benign main task (Main Task Accuracy, MA).

This dual focus guarantees the core functionalities, resilience, and reliability.

Compared to centralized or distributed learning settings such as Federated Learning, poisoning defenses in split learning face several unique challenges.

C1: No Data Access: The training data are located on the client side, preventing the server from inspecting them to detect manipulated training samples. Especially for scenarios that involve sensitive data, it is impractical to assume that clients share their data with the server. Therefore, the server can detect the backdoor only by analyzing the model updates.

C2: Non-Comparable Client Models: Another challenge for detecting poisoning attacks in SL is that the models (updates) of different clients cannot be directly compared to each other. In SL, clients train in sequential order, and each client uses the trained model of its predecessor as the base model for its training. Therefore, the resulting models of two clients will always differ, even if both clients used the same data. Additionally, strategies frequently adopted in Federated Learning [17] to compare the models' updates cannot be transferred to SL. Due to the non-convex training process of DNNs, two clients with the same data might obtain different model updates if they start training from different base models.

C3: Sequential Training: Sequential training also poses a significant challenge in handling detected poisoned models because they affect the training results of following clients. In other distributed learning settings, such as Federated Learning, the defender typically waits until each client has trained its local model before analyzing the model updates and excluding

suspicious ones [14], [8]. However, the sequential training of SL means that if a poisoned model is detected at the end of a training round when every client has finished its training, the models of all clients that followed the malicious clients would need to be discarded. Since these clients used a poisoned model as their base, their models are likely also to contain the backdoor, making them unusable for the server.

In addition, even if it would be practical to repeat the training using a different base model assumed to be benign, a sophisticated adversary might alternate between benign and malicious behavior to fool the server and avoid being removed from the pool of assumed benign clients. Then, when the training process is repeated, the adversary could try to introduce the backdoor using a client who had previously behaved inconspicuously. Therefore, even if the training is repeated, the defense must also be applied during this repetition. On the other side, in the case of the absence of any attack, the defense must be able to accept all models if no poisoning attack is detected. Otherwise, the defense would cause an endless loop of repeating the training after seemingly malicious clients are detected and excluded malicious clients, rendering existing outlier-detection-based techniques [8], [43], [36] impractical for Split Learning.

IV. RATIONALE FOR STATIC AND DYNAMIC ANALYSIS

In the following, we elaborate on the intuition behind the two-fold analysis that is employed to efficiently detect poisoned models.

Static Analysis: The purpose of the frequency analysis is to measure fine-granular static changes that are performed in the backbone during the clients' training. Previous research [40], [60] has shown that in the early stages of the training, mostly the lower components of the models' frequency representation change. Only with progressing convergence do the high frequencies start to change significantly. Therefore, the low-frequency components are especially affected when a model is trained for new behavior. However, when injecting a backdoor that was trained on benign data in advance by earlier clients, the backdoor behavior is in contradiction with the benign model's behavior as the backdoor target label differs from the benign prediction (see Sect. III-B). Therefore, significant adaptions of the model's behavior are necessary, resulting in high changes for the low-frequency components.

To perform this analysis, we first transform each model update into the frequency domain using the Discrete Cosine Transform (DCT) (App. C). The DCT helps break down the model's updates into their frequency components, allowing unusual changes that indicate backdoor behavior to be spotted. Specifically, we use the two-dimensional DCT (2-D DCT) because of its energy compaction property, which means it stores the most important information in the low-frequency components. This characteristic helps detect significant changes, such as those introduced by backdoor attacks. Additionally, the DCT is computationally advantageous over the Discrete Fourier Transform (DFT) because its output is always in real numbers, making it simpler and faster to compute. By

transforming the model updates with the DCT and then calculating the pairwise Euclidean distances between these frequency representations, we can effectively detect anomalies that suggest the presence of backdoor behavior.

To convert a backbone's distances to all other backbones into a score, we sum up the distances to the closest $n/2+1$ other models. As described in Sect. III-B, we assume a majority of clients to be benign. Thus, a model that does not belong to the majority is considered malicious during the current rollback.

Thus, by considering only the $n/2 + 1$ smallest distance values, we prevent $\mathcal{A}$ from manipulating the score calculation, e.g., by providing manipulated models (so-called canaries) that increase the scores of benign models to make the selection of a regular poisoned more likely. By considering only the smallest majority of scores is considered, we ensure that $\mathcal{A}$ cannot increase the score for benign models.

Rotational Analysis: During training, backdoor attacks can cause significant and unusual changes in the orientation or configuration of the model's parameters. To detect such anomalies, SafeSplit employs a dynamic analysis by computing the rotational distance between pre-trained (historical) server backbone updates. This rotational metric captures the extent of changes in the model's parameter space, providing a robust measure of how the backbone evolves.

The rotational distance metric is designed to quantify the orientation shifts in the model's high-dimensional parameter space. Unlike static analysis, which focuses on the magnitude and plain difference for parameter updates, dynamic analysis focuses on the direction and trajectory of these updates. By analyzing the rotational distance between different backbones, SafeSplit can identify deviations from typical training patterns indicative of backdoor behavior. This can be seen as an interpolation of the path that the optimizer took during training from the base model B_{t-1} to the trained model B_t, thus revealing information about the training objective.

SafeSplit computes the rotational distance metric R_D (more details in Section V-D) for each server's backbone. The reasons for utilizing it in the backdoor detection are three-fold. First, R_D analyzes directional changes. The rotational distance metric is sensitive to the direction of parameter updates. Backdoor attacks typically introduce abrupt and significant directional changes to implant malicious behavior, which can be detected by observing large rotational distances. Second, R_D is robust to scaling. By focusing on the orientation rather than the magnitude, the rotational distance metric is robust to variations in the scale of parameter updates, which might occur due to changing client-controlled hyperparameters such as the learning rate, optimization algorithm or loss function. Third, R_D complements the static analysis. While static analysis detects anomalies based on the magnitude and difference of updates in the frequency domain, dynamic analysis adds a layer of scrutiny by examining the trajectory of these updates. Together, they cover both perspectives and provide a comprehensive defense mechanism.

Another important aspect of SafeSplit is the circle-wise analysis that ensures using the latest benign-detected model
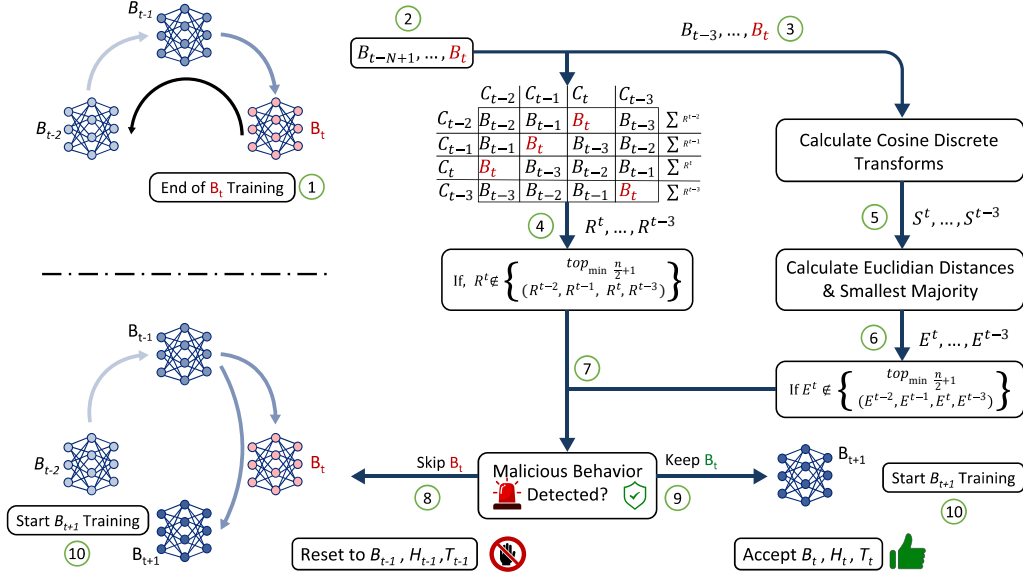
Fig. 3: Workflow of SafeSplit to skip or poisoned models based on an analysis of the models in the frequency domain and their rotational displacement. The workflow is shown for an example scenario consisting of 4 clients.

and, therefore, skipping the malicious clients. This prevents benign clients from training on poisoned models. This approach strengthens the security of the training process, ensuring model reliability as we do not remove the benign clients that have been misclassified as malicious.

V. SAFESPLIT

In the following, we describe the high-level design of SafeSplit (Sect. V-A) and the underlying intuition (Sect. IV) before elaborating on its model-skipping mechanism (Sect. V-B): frequency analysis to detect static changes (Sect. V-C), and rotational distance analysis to detect dynamic changes (Sect. V-D). The overall workflow of SafeSplit is visualized in Fig. 3 and formalized in Alg. 1. Fig. 4 shows the analysis process in more detail.

A. High-level Design

SafeSplit is deployed on the server and analyzes the model's backbone parameters updates. Thus, SafeSplit is executed before each client's training starts to select a benign starting point. The high-level process is shown in Fig. 3. Each time the backbone parameters are updated after a client finishes its training and before the next client starts its training (step 1 in Fig. 3), SafeSplit analyzes the updated backbone for backdoor behavior (steps 2 - 9 in Fig. 3). If no malicious behavior is detected, the training process continues by training the next client using the latest model of the previous client as the base model (steps 9 and 10 in Fig. 4).

However, if backdoor behavior is detected, the rollback mechanism iterates through the previous clients' training outputs to identify the backbone's latest benign state. Thus, this rollback mechanism reverts to the previous server's backbone
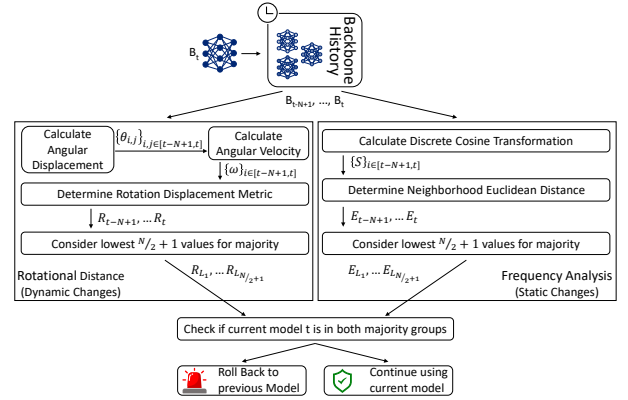

Fig. 4: Overview of SafeSplit, using the latest backbone model B_t and previous backbones $B_{t-N+1} \ldots B_{t-1}$ to determine Rotation Displacement Metric values $R_{t-N+1}, \ldots, R_t$ and the Euclidean Distance Neighborhood Scores, before returning the index of most recent backbone L_i being among $N/2+1$ lowest values $R_{L_1}, \ldots, R_{L_{N/2+1}}$ and $E_{L_1}, \ldots, E_{L_{N/2+1}}$.

state and continues the examination to ensure a benign trained state is indeed selected, bypassing all the malicious client's trained updates (step 8 in Fig. 4). In this case, the current backbone is replaced by the identified latest benign checkpoint, and the next client is instructed to train using the respective head and tail associated with the benign backbone found (step 10 in Fig. 4).

We employ a two-fold analysis strategy to detect the benign state of the server's backbone and detect poisoned updates. First, we perform a static analysis to analyze the Euclidean distances of the backbones' frequency representation (steps

5 and 6 in Fig. 4). Second, we perform a dynamic analysis to quantify the evolution of the backbone's values using our novel rotational distance metric (step 4 in Fig. 4). Both metrics are based on the observation that the backdoor behavior contradicts the benign behavior. While different benign clients either have similar training behavior in the case of IID data or orthogonal behavior in the case of non-IID data, the backdoor aims to make the model mispredict a wrong label (see Sect. III-B), to the backdoor target. Thus, the malicious client needs to change parts of the benign training and train a new backdoor behavior embedded in the model, resulting in large distances for the leveraged metrics.

We use both metrics to determine the benign state of the server's backbone, as each investigates a different perspective. The distances of each metric are used to determine a score for each model that indicates the alignment of the model's training objective. A high score indicates that the current model's training objective was in contradiction with the behavior of a majority of other clients. SafeSplit considers a model to be benign if, for both metrics, the model's scores are small, thus within the respective sets of smallest set $N/2 + 1$ of existing score values (see lines 17-23 in Alg. 1 and step 7 in Fig. 4). Since the majority of models are assumed to be benign (see Sect. III-B), a model that does not belong to the majority is considered malicious during the current rollback.

B. Circular Benign Model Identification

After each client i completes training in their respective communication round t with the server (lines 8 - 9 in Alg. 1, step 1 in Fig. 4), we store the server's backbone B_t in a FIFO array (step 2 in Fig. 4. This process is repeated for each training step t, ensuring we retain the last N models. Once client i finishes training, we compute the deployed metrics (Rotational Distance and Euclidean distance of frequency representation) for all backbone models $B_{t-N+1}, \ldots, B_t$ stored in the FIFO array (lines 10-16 in Alg. 1 and steps 3-6 in Fig. 4).

Next, we determine the benign checkpoint B (lines 17-25 in Alg. 1). This process determines whether the backbone being examined shows backdoor behavior. For this, the server checks for each metric if the model's score is within the smallest $N/2 + 1$ of existing score values (lines 17-19 in Alg. 1 and step 7 in Fig. 4, see also Sect. IV). Alternatively, this process determines how many clients to skip in backward direction to reach a benign backbone checkpoint, i.e., obtain B and the index of the corresponding optimal head (H) and tail (T) of the client determined showing benign behavior (step 8 in Fig. 4). Afterward, the next client is informed which head and tail to use as base model for it's training (step 10 in Fig. 4).

It should be noted that we describe the circular benign model identification for a system where only the backbone is held on the server, while the head and tail always reside on the client side and are forwarded from each client to its successor. As described in the system setting (Sect. III-A), each client signs its training result to allow the succeeding clients to verify the origin of the respective used head and tail. This ensures that the server never gets access to the head and tail, leading to improved privacy. However, SafeSplit can be straightforwardly adapted to scenarios where each client sends the head and tail after the training to the server, and the server forwards them to the next clients.

C. Frequency Variation Computation

After each client i finishes training, we use the Discrete Cosine Transformation (DCT) to determine the frequency representation of each model and select the low frequencies (line 10 in Alg. 1 and step 5 in Fig. 4). Then, we compute the Euclidean distance in the frequency domain for the server's backbone parameters (line 11 in Alg. 1). Let $\mathbf{B}_t$ represent the backbone parameters of the server at the training step t. The low-frequency component of the backbone's update S_t is computed as:

$$S_t = \text{DCT}_{\text{low}}(\mathbf{B}_t - \mathbf{B}_{t-1}) \qquad (2)$$

where $\text{DCT}_{\text{low}}(\mathbf{B}_t)$ denotes only the low-frequency components of the Discrete Cosine Transform (DCT) applied to the backbone's update $\mathbf{B}_t - \mathbf{B}_{t-1}$. We then calculate for each frequency representation S_t the Euclidean distances to each other frequency representation (step 6 in Fig. 4):

$$ed_{t,i} = \|S_t - S_i\|_2 \qquad (3)$$

The frequency score E_t of the model B_t is then calculated as the sum of the distances to the frequency representation of the $N/2 + 1$ closest other models (line 12 in Alg. 1 and step 6 in Fig. 4).

The DCT converts the backbone parameters $\mathbf{B}_t$ into their frequency domain representation. This transformation allows us to analyze how the frequency components of the backbone parameters change over time. The Euclidean distance D_t quantifies the pairwise differences between the frequency representations of the backbone parameters for each client i and j in $(\mathbf{B}_{t-N+1}, \mathbf{B}_t)$. Larger distances may indicate significant changes in the backbone's frequency characteristics, which could be indicative of injecting contradicting behavior, being typical for backdoor attacks. This static analysis provides insights into the stability and consistency of the server's backbone parameters over multiple training steps, helping to detect and mitigate potential security risks in the SL framework.

D. Measuring Rotation Distances

In this section, we describe the computation of the rotational distance metric (step in Fig. 4) that captures the dynamic changes in the server's backbone's update for each client's training. As mentioned in Sect. V-B, to determine the checkpoint of the benign behavior, we keep track of the N historical backbone updates and compute the pairwise differences between R_D of the backbone parameters for each client $i, j \in \{\mathbf{B}_{t-N+1}, \mathbf{B}_t\}$ after training the current client i. R_D computation consists of three steps: First, we compute the angular displacement of the parameter values in the backbone (line 13 in Alg. 1). Second, we compute the angular velocity to quantify the rate of change in the angular displacement.

Lastly, we compute the rotational frequency to identify the frequency of these directional changes.

Angular Displacement Computation. The angular displacement $\theta(t)$ measures the orientation change between successive backbone updates. To inject contradicting backdoor behavior, the attack introduces significant directional changes, causing $\theta(t)$ to deviate from normal training patterns. Thus, given the current client i and its backbone model B_i, we compute the angular displacement that measures a rotating object's change in orientation (line 13 in Alg. 1).

Let $\mathbf{B}_t$ and $\mathbf{B}_{t-1}$ be the backbone parameters at two consecutive training steps t and $t-1$. The angular displacement $\theta(t)$ between these updates is given by:

$$\theta(t) = \arccos\left(\frac{\mathbf{B}_t \cdot \mathbf{B}_{t-1}}{\|\mathbf{B}_t\|\|\mathbf{B}_{t-1}\|}\right) \quad (4)$$

where $\|\cdot\|$ denotes the Euclidean norm of the vectors. We compute the pairwise differences between $\theta(t)$ for each client i and j in $(\mathbf{B}_{t-N+1}, \mathbf{B}_t)$ after training the current client. Then, the pairwise difference of $\theta_i(t)$ computed for client i against $N-i$ clients is averaged to obtain a single score value.

This allows to obtain the angular position at each time point. The angular displacement $\theta(t)$ measures the change in orientation between successive backbone updates. Backdoor attacks often introduce significant directional changes, causing θ_t to deviate from normal training patterns.

Angular Velocity Computation. The rate of change of the angular displacement over time, or angular velocity $\omega(t)$, is given by:

$$\omega(t) = \frac{\theta(t) - \theta(t-1)}{\Delta t} \quad (5)$$

where Δt is the time interval between successive updates. The angular velocity $\omega(t)$ quantifies the rate of change in $\theta(t)$ (see line 15 in Alg. 1). High angular velocities $\omega(t)$ indicate rapid shifts in the parameter space, which are uncommon during regular training but typical of backdoor insertion attempts.

Rotational Distance Metric (R_D) Computation. At the end, we compute the rotational displacement metric R_D, which is also the rotational frequency that captures the frequency of these directional changes (see line 16 in Alg. 1). R_D is defined as:

$$R_D = \frac{\omega(t)}{2\pi} \quad (6)$$

We divide $\omega(t)$ by $2 \cdot \pi$ because one complete revolution (or cycle) around a circle corresponds to an angle of $2 \cdot \pi$ radians. So, dividing by $2 \cdot \pi$ allows us to convert ω, measured in radians per second, into the rotational frequency, typically measured in cycles per second or Hertz (Hz).

The rotational frequency R_D, derived from $\omega(t)$, identifies the frequency of these directional changes. A high R_D suggests frequent and abrupt alterations in parameter orientation, which is characteristic of backdoor attacks that

---

Algorithm 1 SafeSplit

1: **Input:** N, R, B_0, H_0, T_0, clients $\qquad \triangleright$ N is the number of clients, R the number of training rounds, H_0, B_0, T_0 build the initial model, a FIFO list of N clients
2: **Output:** B^*, H^*, T^* $\qquad\qquad \triangleright$ updated backbone, heads and tails

3: **function** SMALLESTMAJORITY($v_1, \ldots, v_N$)
4: $\quad$ **return** sorted($v_1, \ldots, v_N$)[1,...,$N/2 + 1$]
5: **end function**

6: $H, B, T \leftarrow H_0, B_0, T_0$
7: **for** each training step $t \in [1, R \cdot N]$ **do**
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad \triangleright$ Perform Training
8: $\quad$ current_client $\leftarrow$ clients[$t \bmod N$]
9: $\quad$ $H_t^*, B_t^*, T_t^* \leftarrow$ TRAIN(current_client, H^*, B^*, T^*)
$\qquad\qquad\qquad\qquad \triangleright$ Measure Distance in Frequency Domain
10: $\quad$ $\forall i \in \{t-N+1, \ldots, t\} S_i \leftarrow$ DCT($B_i - B_{i-1}$)
11: $\quad$ $\forall i, j \in \{t-N+1, \ldots, t\} ed_{i,j} \leftarrow$ EuclideanDistance(S_i, S_j)
12: $\quad$ $\forall i \in \{t-N+1, \ldots, t\} E_i \leftarrow \sum$ SmallestMajority($ed_{i,t-N+1}, \ldots, ed_{i,t}$)
$\qquad\qquad\qquad\qquad\qquad \triangleright$ Calculate Rotational Distance
13: $\quad$ $\forall i, j \in \{t-N+1, \ldots, t\} \theta_{i,j} \leftarrow$ AngularDisplacement(B_i, B_j)
14: $\quad$ $\forall i \in \{t-N+1, \ldots, t\} adn_i \leftarrow \sum$ SmallestMajority($\theta_{i,t-N+1}, \ldots, \theta_{i,t}$)
15: $\quad$ $\forall i \in \{t-N+1, \ldots, t\} \omega_i \leftarrow$ AngularVelocity($adn_i(t), adn_i(t-1)$)
16: $\quad$ $\forall i \in \{t-N+1, \ldots, t\} R_i \leftarrow$ RotationalDisplacement(ω_i)
$\qquad\qquad\qquad\qquad\qquad \triangleright$ Determine Benign Majority
17: $\quad$ frequency_majority = arg_sort($E_{t-N+1}, \ldots, E_t$)[1, ..., $N/2 + 1$]
18: $\quad$ rotation_majority = arg_sort($R_{t-N+1}, \ldots, R_t$)[1, ..., $N/2 + 1$]
19: $\quad$ benign_majority $\leftarrow$ rotation_majority $\cap$ frequency_majority
$\qquad\qquad\qquad\qquad\qquad \triangleright$ Determine Latest Benign Model
20: $\quad$ **for** each previous client $c \in [t, \max(t-N+1, 1)]$ **do**
21: $\quad\quad$ **if** $c \in$ benign_majority **then**
22: $\quad\quad\quad$ $H^*, B^*, T^* \leftarrow H_c, B_c, T_c$
23: $\quad\quad\quad$ **break**
24: $\quad\quad$ **end if**
25: $\quad$ **end for**
26: **end for**
27: **return** H^*, B^*, T^*

---

aim to alter model behavior.

The rotational distance metric allows capturing dynamic changes in the orientation or configuration of the backbone during the local training of clients. It primarily measures angular changes between gradient vectors (geometrically). As we show in the following sections, this makes the metric highly effective for identifying deviations of poisoned models. A major strength of the metric is its ability to analyze the rotational changes over time, making it resilient to small, random perturbations or noise introduced in the gradients. Hence, it is able to discern the changes between the malicious updates even if the noise is systematic and mimics the patterns introduced by a poisoning attack. This makes the metric robust against manipulations to hide the poisoned gradients.

After the DCT and R_D computation, we use them to compute the benign checkpoint to superimpose the checkpoint's backbone model B^* to the current backbone state B_i at the server and transfer the corresponding optimal head (H^*) and tail (T^*) of the client at the benign checkpoint to the next client for its training (to use as the base model).

VI. EVALUATION

This section presents a comprehensive evaluation of SafeSplit across various types of backdoor attacks, showcasing our defense mechanism's ability to maintain backdoor accuracy below 5%. In App. D, we analyze the effectiveness of the combination of static and dynamic perspectives. In App. F we evaluate further non-backdoor attacks.

A. Experimental Setup

Metrics: Our evaluation of SafeSplit leverages the following key metrics.

Backdoor Accuracy (BA) indicates the model's accuracy concerning malicious tasks. It quantifies the fraction of the trigger set for which the model generates predictions aligned with the attacker's objectives. The attacker aims to maximize BA, while SafeSplit strives to minimize it.

Main Task Accuracy (MA) measures the model's accuracy for benign tasks. It reflects the percentage of clean inputs for which the model delivers accurate predictions. The adversary aims to reduce the impact on MA to diminish the likelihood of detection. An essential requirement of SafeSplit is not to significantly affect the MA.

Datasets. Aligned with existing work on SL [64], [22], [38], [63], [6], [49], [19], [13], [50], [16], we leveraged five datasets (CIFAR-10, FMNIST and MNIST, GTSRB, CIFAR-100) to perform our experiments:

CIFAR-10 consists of 50 000 training and 10 000 test images of size 32×32 pixels, showing objects and animals belonging to 10 different classes [26]. As DNN, we use the widely adopted ResNet-18 architecture.

MNIST consists of 60 000 training and 10 000 test grayscale images showing handwritten digits. Aligned with recent work on distributed learning, we implemented a Convolutional Neural Network (CNN), as described by Cao *et al.* [10].

FMNIST is composed of 60 000 training and 10 000 test images, each sized 28x28 pixels, depicting various types of clothing across 10 classes [58]. As DNN we use also the CNN described by Cao *et al.* [10].

CIFAR-100 consists similar to CIFAR-10 of 50 000 training and 10 000 test images but is categorized into 100 classes. Due to the high number of labels, which are significantly larger than the considered number of clients. As DNN, we used Wide-ResNet50, being pretrained for ImageNet dataset and replaced the final layer, while the training included all layers.

GTSRB consists of 39 000 training and 12 600 test images of varying sizes, from 32×32 to 64×64 pixels, showing traffic signs belonging to 43 different classes [47]. We used the MicronNet [56] architecture as DNN.

Computational Setup. We conducted the experiments using the deep learning library PyTorch [3]. The experiments were conducted on a server with 4x NVIDIA A6000, an AMD EPYC 7773X CPU using 64 physical cores, and 768 GB of main memory.

Model Architectures. Due to the absence of existing work on mitigating backdoor attacks on SL, we aligned the used DNN architectures on existing work about the security of Federated Learning [10], [14]. To ensure a comprehensive evaluation, we included 4 different model architectures in our evaluation with parameter sizes ranging from 440 812 to 13 425 930 trainable parameters. The details of the used DNN

TABLE I: Overview of used Deep Neural Network (DNN) Architectures.

Model	Number of Parameters				Evaluated Datasets
	Head	Backbone	Tail	Total	
ResNet18 [21]	9 536	11 166 976	5 130	11 181 642	CIFAR-10
Simple CNN [10]	1 520	665 562	5 130	672 212	CIFAR-10
Simple CNN [10]	520	435 162	5 130	440 812	MNIST, FMNIST
GoogLeNet [48]	124 736	5 475 168	10 250	5 610 154	CIFAR-10
VGG11 [45]	1 920	9 224 064	4 199 946	13 425 930	CIFAR-10
Wide-ResNet50 [65]	9 536	66 824 704	204 900	67 039 140	CIFAR-100
MicronNet [56]	824	411 192	3 010	415 026	GTSRB

architectures are shown in Tab. I. We focused our evaluation on the CIFAR-10 dataset and evaluated all DNN architectures for this dataset (see Sect. VI-H). In addition, we also evaluated the MNIST and FMNIST datasets. Due to the simplicity of these datasets, we used a simple CNN for these datasets, as defined by Cao *et al.* [10] for their evaluation of backdoor attacks against Federated Learning. Notably, the structure of this simple CNN slightly varies for different datasets, as the images in CIFAR-10 have a dimension of 32×32 pixels while MNIST and FMNIST consist of images with the dimensions 28×28.

Training Parameters. Unless stated otherwise, we considered a system consisting of 10 clients, from which 2 are malicious and aim to inject a backdoor. We simulated non-IID data on the client side using the main-label strategy that is frequently used in other work on distributed learning [9], [14]. Here, each client gets randomly assigned a main label. While a certain fraction, indicated by the IID rate, is samples from all available samples, the remaining samples are chosen only from the assigned main label class. An IID rate of 1.0 indicates a complete IID data distribution among clients. As the default value, we use an IID rate of 0.8. Only for GTSRB and CIFAR-100 we used an IID-rate of 1.0 as default value to achieve a decent MA, as the number of clients here is significantly lower than the number of labels.

Considered Backdoor Behavior. In the following, we evaluate SafeSplit's effectiveness for different datasets and backdoors. Unless stated otherwise, we use for CIFAR-10 a semantic backdoor that misclassified cars in front of a striped background as birds and for all other data sets a backdoor being activated by a red rectangle or, in the case of grayscale data sets such as MNIST, a white rectangle.

B. Effectiveness of SafeSplit

SafeSplit undergoes extensive evaluation against different backdoor attacks for pixel and semantic triggers, as depicted in Table II. As the table shows, SafeSplit reduced the BA in cases to a negligible value. Notably, for pixel trigger backdoors, the BA is often even for a benign model not exactly 0%. This phenomenon occurs because the model misclassified some test samples, and the metric counts them in favor of the backdoor if the predicted label is equal to the backdoor target label (see App. E for details). Furthermore, we conduct thorough evaluations on the CIFAR-10 dataset in various attack scenarios, which we describe in the following.

C. Different Data Scenarios

We conducted different experiments on the CIFAR-10 dataset, varying the degree of non-IID data distribution to assess its effect on SafeSplit performance. The results are shown in Table III. Across IID rates of 0.6, 0.8, and 1.0, SafeSplit demonstrates significant reductions in BA while maintaining MA. Notably, for decreasing IID rates, the MA goes down regardless of the presence of an attack or defense, as training in such non-IID settings becomes very challenging.

Further, we assessed different numbers of clients being involved in the training process, ranging from 5 to 20, with 20% of them being malicious. Fig. 5 illustrates the BA and MA for CIFAR-10. As the figure shows, SafeSplit effectively mitigates the attack, therefore keeping the BA 0% and maintaining the MA with only a negligible drop compared to the scenario without a defense applied.

D. Impact of Poisoned Model Rate (PMR)

Fig. 6 shows SafeSplit's effectiveness for varying ratios of malicious clients (PMR). As the figure shows, SafeSplit effectively mitigates the backdoor, keeping the BA at 0%, while maintaining the MA close to the scenario without defense. Notably, the figure also shows that the attack is less effective for low PMRs.

E. Adaptive Attacks

A sophisticated adversary that is aware of the defense might adapt the attack to make it more effective against the deployed defense approach. In the following section, we describe and evaluate various sophisticated attack strategies. Notably, some of the adaptive attacks assume a stronger adversary having knowledge of the current parameters of the backbone model, exceeding the previously defined threat model (see Sect. III-B). In practice, such an adversary could, for instance, approximate the backbone's parameters using a shadow model trained separately with the malicious clients' benign data. However, for the sake of a comprehensive evaluation, the following

section assumes that the adversary has knowledge of the actual backbone parameters.

Varying the Poisoned Data Rate (PDR). To increase the similarity of the poisoned models to benign models, $\mathcal{A}$ might vary the ratio of data samples for the backdoor behavior in its local dataset (Poisoned Data Rate, PDR). The choice of this parameter realizes a tradeoff since low PDRs ensure that the resulting model remains inconspicuous but also reduce the efficiency of the attack, while high values result in a high attack impact but also make the models easier to detect. Tab. V shows the effectiveness of SafeSplit for varying PDRs. As the table shows, SafeSplit effectively mitigates the attack for all PDR values. Notably, the table also shows that the attack is, in the absence of a defense, most effective for a PDR of 50%, while for higher PDR values, the BA is reduced. Although this might be counterintuitive at first glance, a reason for this might be that for high PDRs, the dataset of the malicious clients becomes highly imbalanced and focuses on the single backdoor target label. This imbalanced data causes the model to overfit significantly and results in changes that can be easily reverted by the benign clients through their training.

Loss Constrain for Rotation Distance Metric. A key aspect of SafeSplit's backdoor detection is the rotational distance metric. Following existing attacks on Federated Learning [5], a sophisticated adversary might adapt the loss function and integrate a regularization term into the loss function that minimizes the rotational distance of the current poisoned model to a reference model. Given the loss function $\mathcal{L}_{\text{class}}$ that measures the model's performance on its training data, the anomaly evasion loss $\mathcal{L}_{\text{ano}}$ that measures the suspiciousness of the model, here the rotational distance to the reference model, then according to Bagdasaryan *et al.* the combined loss function is defined as

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{class}} + (1 - \alpha) \cdot \mathcal{L}_{\text{ano}} \tag{7}$$

where α is a hyperparameter that weights both terms.

We evaluated this attack using the base model ($\mathcal{A}$ has knowledge of the backbone) as reference model. Notably, knowing the backbone is not a realistic threat scenario, as the adversary would need to have access to the server, going beyond our threat model (see Sect. III-B). Tab. IV shows the results for different α-values. Although the adversary reduces the suspiciousness for the rotational distance, due to

TABLE II: Effectiveness of SafeSplit against different attacks for the respective dataset, in terms of Backdoor Accuracy (BA) and Main Task Accuracy (MA).

Dataset	Attacks	No Defense		SafeSplit	
		BA	MA	BA	MA
CIFAR-10	Semantic Trigger	59.3	66.6	0.0	62.7
	Pixel Trigger	100.0	66.6	0.3	66.4
MNIST	Pixel Trigger	86.2	98.7	0.0	98.8
FMNIST	Pixel Trigger	79.8	83.0	3.4	84.6
CIFAR-100	Pixel Trigger	93.3	76.8	0.1	76.5
GTSRB	Pixel Trigger	30.0	58.0	0.6	63.7

TABLE III: Effectiveness of SafeSplit for different degrees of IID data.

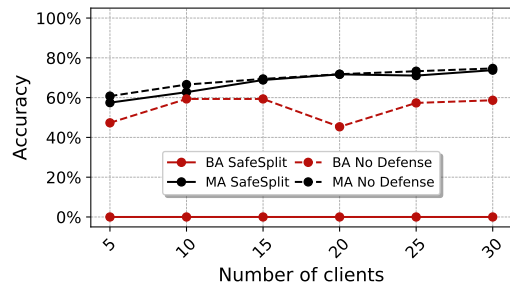| IID-Rate	No Defense		SafeSplit	
	BA	MA	BA	MA
0.6	44.7	59.5	0.0	57.5
0.8	59.3	66.6	0.0	62.7
1.0	60.7	69.4	0.0	68.1

Fig. 5: BA and MA for different participant numbers.

the combination of a static and dynamic analysis SafeSplit still effectively detected the poisoned models and reduced the BA to 0%. Notably, setting $\alpha = 0.5$ achieves in the absence of a defense the best BA without significantly degrading the MA. Thus, in the following experiments, we use $\alpha = 0.5$.

TABLE IV: Effectiveness of SafeSplit against loss-constrain using different α-values.

	No Defense		SafeSplit	
	BA	MA	BA	MA
$\alpha = 0.25$	62.0%	66.3%	0.0%	63.5%
$\alpha = 0.50$	60.0%	66.4%	0.0%	64.1%
$\alpha = 0.75$	51.3%	65.9%	0.0%	63.9%

To evaluate also a more realistic setting, we repeated the experiment using a separate backbone as reference that was trained from the same base model. However, also here SafeSplit was able to detect the poisoned model updates, resulting in a BA of 0%.

Loss Constrain for Static Distance Metric. Analogous to the previous paragraph we added a regularization term that focuses on the frequency analysis of SafeSplit as regularization term. Again, we evaluated the attack using the base model and also a benign model as reference models. In both cases, we extracted the low frequencies of the Discrete Cosine Transform of the reference model and the poisoned model under training; we then constrained the loss function to minimize the Euclidean Distance between the two frequency representations. However, due to the characteristics of the Discrete Cosine Transform domain, even small changes in the low-frequency components of the model can result in significant changes in the model's parameters. The regularization terms of the loss function, can either try to only inject the backdoor behavior (with α values close to 1) or try to bring the malicious model DCT as similar as possible to a benign one (with α values close to 0). Therefore, in the first case, the model behaves again as a malicious model, and it is detected by SafeSplit, and in the second case the adversary fails to implant any meaningful backdoor, with a BA of 0%. A third case is a balance of the two regularization terms (with α values close to 0.5), but in this scenario, the changes to the low frequencies produce a scrambled model with very unnatural behavior. Therefore, as SafeSplit is highly sensitive to dynamic changes in model behavior, it was still able to detect the backdoored models and
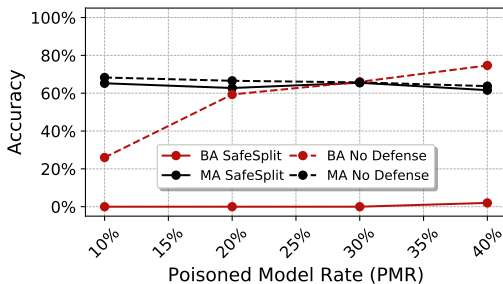
reduced the BA to 0% in both cases.

Loss Constrain for Static and Dynamic Distance Metrics. Building on the two previous attacks, we integrated both loss functions as regularization terms. However, we observed SafeSplit to remain effective and reduce the BA to 0% for both reference models (base model and trained benign model). This might be caused by the trade-off that the DNN optimizer needs to perform during the training. Because of the regularization term, the model must not show any indications of contradicting (backdoor) behavior. However, at the same time, due to the original loss that focuses on optimizing the predictions, the model is trained to show backdoor behavior. The optimizer then aims to minimize both aspects and perform a trade-off between both aspects. However, the resulting model will then still show poisoned behavior, allowing SafeSplit to detect it.

Loss Constrain for Euclidean Distance. An alternative option is to constrain the model not with respect to the specific metric that the defense uses but to integrate the Euclidean distance into the loss function to keep all parameters of the model close to the respective reference model. However, as this technique needs to perform a trade-off again, it still needs to train the model on the backdoor behavior, SafeSplit detected the nuances that indicate the poisoning and reduced the BA to 0%.

Focus Training on Tail. SafeSplit analyses only the backbone model, as the head and tail are held by the clients, and the server has no access to them in order to preserve the client's privacy. However, a sophisticated adversary might try to exploit this and train the backdoor only into the tail. To achieve this, the malicious clients train the model alternating with batches containing only benign samples and samples also containing samples for the backdoor behavior. For the batches containing backdoor behavior, the client uses the server for forward propagation, thus making the predictions. However, after calculating the loss for the poisoned batches, the backpropagation and gradient descent are only applied to the tail, and the gradients are not shared with the server. As a result, the backbone and head will be trained only with clean data, while the tail is trained for benign and poisoned data. In this case, the backbone would be inconspicuous. However, as our experiments showed, for this attack strategy, the backdoor is not injected into the model, even without defense, and the BA remains 0%.

F. Comparison with Defenses for Federated Learning

While the existing literature does not provide defenses against client-side backdoor attacks in SL, many defenses

Fig. 6: BA and MA for various Poisoned Model Rates (PMRs).

TABLE V: Effectiveness of SafeSplit for different Poisoned Data Rates (PDRs).

	No Defense		SafeSplit	
PDR-Rate	BA	MA	BA	MA
25%	85.3	68.4	0.0	64.7
50%	88.7	67.9	0.0	65.1
75%	59.3	66.6	0.0	62.7
100%	24.0	51.0	0.0	64.5

(a) Backdoor Accuracy
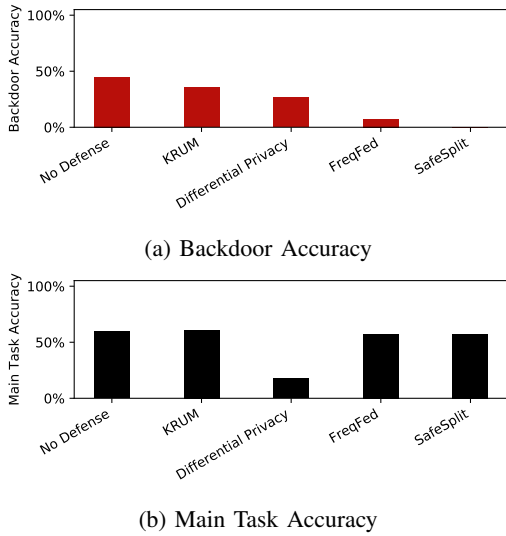

(b) Main Task Accuracy

Fig. 7: Comparison of different state-of-the-art defense techniques against SafeSplit using Main Task Accuracy (MA) and Backdoor Accuracy (BA).

were proposed for the distributed learning scheme Federated Learning. Based on the categorization of Fereidooni *et al.* [14], we selected 3 representative backdoor defenses for FL and adapted them for SL. Particularly, we evaluated FreqFed [14] as an approach that aims to classify all benign and poisoned models, KRUM [8] that focuses on correctly identifying a subset of benign models, and Differential Privacy [5], [35] that aims to mitigate the backdoor without identifying the attackers. Notably, defending backdoor attacks in SL raises significant challenges (see Sect. III-C), which SafeSplit addresses. To avoid any unjustified disadvantages for the existing techniques, we adapted them, using Euclidean distance of the updates for Krum and integrated SafeSplit's circlewise structure into FreqFed. For Differential Privacy no adaptions were necessary, as noising and clipping are independent of the SL's structure. Fig. 7 shows the BA and MA values for a non-IID scenario using an IID rate of 0.6. All defenses except Differential Privacy maintain a comparable MA. However, despite the adaptions, only SafeSplit is able to mitigate the attack and reduce the BA to 0%.

G. Performance in the Absence of Attacks.

An important objective for a practical defense is to not negatively affect the training process (see O2 in Sect. III-C). To measure the impact of SafeSplit on the training of a model, we conducted several experiments starting from a random base when no attack is performed and no defense is applied or SafeSplit is deployed. After 50 rounds of training without defense, the MA reached 69.30%, while when SafeSplit was used, the MA achieved 66.6%; thus, also, when deploying SafeSplit, no significant drop in MA was observed.

H. Performance for Different Model Architectures.

We evaluated the effectiveness of SafeSplit using different model architectures. Tab. VI shows the effectiveness of SafeSplit for 4 different model architectures that are frequently used in distributed learning [10], [14], [20]. As the table shows, SafeSplit successfully mitigates the backdoor for all models while maintaining the MA on a similar level as without the attack.

In addition, we also evaluated different positions for the cutting layer for the ResNet-18 model to simulate different backbone sizes. Resnet-18 consists of a convolutional layer a batch normalization layer, a ReLu layer, a max-pooling layer, 4 blocks and a linear layer. Each block consists of 4 convolutional layers, several batch normalization layers, and a down-sampling layer. We simulated different backbone sizes by assigning different numbers of blocks (2,3,4) to the backbone and clients. We observed SafeSplit to effectively mitigate the attack and reduce the BA to 0%.

In summary, we evaluated SafeSplit for different datasets and data scenarios, attack types and settings, and client numbers and compared it against different baseline defenses. SafeSplit was always able to mitigate the backdoor attack and reduced in all experiments the BA to less than 5%.

VII. SECURITY CONSIDERATIONS

In the following, we discuss the security aspects of our scheme to fulfill our security and functional objectives and challenges (cf. Sect. III-B).

To address objective O1 (Prevent Backdoor Attacks), a backdoor defense must fulfill the security requirement of significantly reducing the attack impact. In the following, we will first discuss the risk of server-side attacks, before summarizing SafeSplit's resilience against backdoor attacks (Backdoor Resilience), including its robustness against defense-adapted attack strategies that aim to make the DNNs' parameters inconspicuous (Adaptive Attacks) as well as attack strategies that aim to prevent that the actually poisoned parameters are analyzed by the defense (Analysis Evasion).

Effectiveness against Different Adversary Models. We introduced the first defense against backdoor attacks in Split Learning, focusing on client-side attacks. As mentioned in Section III-B, server-side backdoor attacks have already been the subject of investigation in the literature [13]. In contrast, client-side backdoor attacks remained an open challenge, as we discussed in Section III-B. Moreover, in typical practical settings, we expect the server side to be a well-protected cloud instance, unlike the client side, which can be any mobile device with less protection. In Sect. VI, we extensively evaluated SafeSplit in various scenarios using different attack settings.

Backdoor Resilience. In the Sect. VI, we evaluated the effectiveness of SafeSplit in various scenarios and attack settings. Particularly, we used five state-of-the-art image recognition benchmark datasets (CIFAR-10, MNIST, FMNIST, GTSRB,

TABLE VI: Effectiveness of SafeSplit for different Deep Neural Network Architectures using the CIFAR-10 dataset.

| | No Defense | | | | SafeSplit | |
| | No Attack | | Attack | | Attack | |
Model	BA	MA	BA	MA	BA	MA
ResNet-18	0.0	69.3	59.3	66.6	0.0	62.7
Simple CNN	0.0	64.1	78.0	62.7	0.0	60.4
GoogLeNet	0.0	63.5	16.7	57.9	0.0	60.2
VGG11	0.0	47.6	76.7	49.7	0.0	43.0

CIFAR-100; see Tab. II), different numbers of clients (see Fig. 5), IID settings (see Tab. III), 4 different model architectures (see Tab. VI) and various scenarios. In addition, we evaluated different attack settings, particularly different ratios of poisoned data (see Tab. V) and attack strategies (see Sect. VI-E). We observed that in all cases SafeSplit mitages the attack successfully, thus achieving O1 (see Sect. III-B).

Adaptive Attacks. A powerful adversarial evasion strategy concerns manipulating the training process to ensure the analyzed backbone model does not show high angular distance values. In this paper, we consider a sophisticated adversary, being aware of the defense and having full control over the individual clients (see Sect. III-B). In Sect. VI-E, we evaluated several defense agnostic attacks that aim to circumvent SafeSplit by leveraging loss-constrain and integrating the angular distance and frequency distance into the loss function, restricting the distance to benign models, and training the backbone only with benign data. However, SafeSplit was able to successfully mitigate all of these attacks, showing SafeSplit's robustness even against sophisticated adaptive attack strategies.

Analysis Evasion. In our adversary model, the adversary has full control over the head and tail (see Sect. III-B) and can, before forwarding them to the next client, even entirely replace the values. Thus, the forwarded values are not necessarily those obtained by the training. SafeSplit counters this attack by analyzing the backbone stored on the server, where they can only be changed in a well-controlled manner (backpropagation). This ensures that the model used for the following clients is the same for backdoor detection, effectively preventing time-to-check-time-to-use attacks. Another strategy to distract angular metrics in other distributed settings is upscaling the model's parameters without changing them [5], [7]. However, as only the server has access to the backbone, this strategy is restricted to the head and tail that reside at the clients. Therefore, the backbone would need to remain inconspicuous to circumvent SafeSplit. However, in Sect. VI, we demonstrated that changing the backbone is essential for injecting the backdoor, such that SafeSplit is also robust against scaling attacks. Thus, SafeSplit effectively and significantly reduces the risk of adversaries injecting backdoors into the model while marginally impacting the benign main task accuracy, fulfilling our functional and security objectives and requirements.

VIII. RELATED WORK

This section provides an overview of the recent research progress in the fields of privacy, security, and reliability for split learning.

Security of Split Learning. As mentioned in Sect. I, split learning has emerged as a promising alternative to Federated Learning, offering substantial reductions in the computational resources required by participants [19], [46], [50]. However, the inherent data and model control separation in split learning has raised various security concerns that can be categorized as the vulnerability to: data reconstruction [38], [13], [18], [64], label inference attacks [29], [15], [32], [6], and backdoor attacks [49], [63], [22], [6]. While we only focus on the backdoor attacks in this paper, we briefly explain the other attack vectors for completeness.

In a data reconstruction attack, the adversary resides on the server and tries to recover the original training data from the feature vector uploaded by the clients. These attacks are further divided into two categories [64]: passive attacks, where the server does not disrupt the standard training process, only leveraging the intermediate steps of the training to gain information about the samples and create an attack model [13], [18]. Active data reconstruction attacks instead manipulate the training process to reconstruct the client data [38], [64] and obtain better results but faces the risk of being more easily detectable by the client defenses [13], [16].

In vanilla Split Learning, it is assumed that the clients cannot access the target labels. At the same time, the server cannot associate each label with specific samples to maintain privacy. Therefore, label inference attacks involve an adversary having control of both, the server and some clients to aim and steal label information of the data missing from the adversary's samples [29], [15], [32], [6]. The defense mechanisms proposed to counteract label inference attacks primarily rely on random disturbance and differential privacy [61], [31].

Lastly, in the context of Split Learning, the objective of backdoor attacks shifts towards implanting backdoors into the connected model. However, the challenge differs depending on whether the adversary controls the server or the clients. In the first scenario, the server aims to successfully compromise the portion of the model that the clients possess and can attain this using surrogate clients [49] or shadow models [63]. In the client-side backdoor attack, the adversary aims to compromise both, the model residing in the server and, as an additional challenge, the backdoor needs to persist even on the portion of models possessed by the other victim clients. To achieve this, the malicious clients can employ auxiliary models [63] or poison the training by submitting ad-hoc trigger vectors during training [22], [6]. However, we stress that the existing literature on client-side backdoor attacks was proposed only for the vanilla Split Learning framework, where it is assumed that the clients do not have access to ground truth label information [6]. Hence, mitigating client-side attacks in SL has remained an open challenge we aim to tackle in this paper for the first time, especially in the context of U-shaped SL.

Backdoor Defenses in Federated Learning (FL). In the context of other distributed collaborative learning paradigms, such as FL the objective of backdoor attacks is to implant

backdoors into the global model. However, the challenge lies in preserving the efficacy of the local model's backdoor post-aggregation on the server side. Notable contributions in this domain include the works of Bagdasaryan *et al.* [5] and Xie *et al.* [59], Saha *et al.* [42], Shumailov *et al.* [44], and Wenger *et al.* [55], aim to compromise model integrity through subtle or overt manipulation of the training data.

Multiple defenses have been proposed to mitigate these attacks in FL [14], [17], [8], [5], [53], [10], [28]. KRUM calculates the pairwise Euclidean Distances and, analogously to SafeSplit, sums up the neighborhood to obtain a score. Afterward, the model with the lowest KRUM score is selected as the aggregated model. However, the Euclidean distance can be circumvented as shown by various works for FL [14] and in Sect. VI for SL. Further, in SL, it is important to select the latest model rather than the model with the smallest score. FreqFed [14] also leverages the observation that training a model for new behavior results in large changes in a model's low-frequency components. To detect backdoored models, clustering is applied to the models' low frequencies. However, FreqFed is not applicable to the Split Learning domain due to the sequentiality of the training steps and other challenges as mentioned in Sect. III-C. Bagdasaryan *et al.* [5] consider using Differential Privacy, but as our evaluation Sect. VI shows, neither adding random noise to the backbone nor restricting the $L_2 -$ norm of the updates helps in counteracting an advanced attack in SL. FoolsGold [17] is a FLdefense that assumes high similarity between model updates submitted by the adversary, penalizing clients that submit similar (sybils) updates during aggregation. As we discussed in Sect. I differently from FL, each client does not start training from the same global model, but instead, each starts training from a different base model. Therefore, malicious clients will each impact in different ways the updates on the backbone. Wang *et al.* [53] propose a model poisoning defense that analyzes the latent space of the second to last layer of the DNN, to detect malicious behavior. However, as we describe in Sect. III-A in the U-shaped SL framework, the server does not have access to the last layers, which are instead controlled by the clients, further as outlined in Sect. III-C the server should not have access to any training or validation data. Therefore, it would be unable to analyze the behavior of the last layers due to its lack of access to data. Cao *et al.* introduced a secure aggregation protocol, where each client is assigned into random subsets of clients, and for each subset a separate distributed learning process is executed, resulting in separate models for each subset. Then, during inference, each global model is queried separately, and the final label prediction is determined by majority voting. Unfortunately, repeating the training process and inference for each sample multiple times results in an impractical overhead. Furthermore, the defense has been broken for a non-tiny number of malicious clients. Furthermore, in the case of highly non-IID data or a limited number of participant clients, dividing the training into multiple subsets will impact the overall MA of the trained models.
In comparison, SafeSplit employs different metrics to detect

backdoored models by analyzing models from the static and dynamic perspectives. The circlewise rollback mechanism allows skipping detected poisoned models while choosing the latest benign model to prevent reverting benign training contributions.

IX. CONCLUSION

In this paper, we proposed SafeSplit, the first defense against client-side backdoor attacks in SL. Unlike existing methods for other distributed learning schemes, SafeSplit employs a rollback mechanism in which we conduct static (frequency) and dynamic (rotational) analysis to detect poisoned model updates and limit attack impact. Our extensive evaluation demonstrated SafeSplit's effectiveness across various scenarios, attack settings, and defense-agnostic strategies.

ACKNOWLEDGMENT

REFERENCES

[1] Health Insurance Portability and Accountability Act, 1996. https://www.govinfo.gov/content/pkg/PLAW-104publ191/pdf/PLAW-104publ191.pdf.

[2] General Data Protection Regulation, 2018. https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[3] Pytorch, 2022. https://pytorch.org.

[4] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[5] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How To Backdoor Federated Learning. In *AISTATS*, 2020.

[6] Yijie Bai, Yanjiao Chen, Hanlei Zhang, Wenyuan Xu, Haiqin Weng, and Dou Goodman. VILLAIN: Backdoor attacks against vertical split learning. In *USENIX Security*, 2023.

[7] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*. PMLR, 2019.

[8] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NIPS*, 2017.

[9] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *NDSS*, 2021.

[10] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Provably secure federated learning against malicious clients. In *AAAI Conference on Artificial Intelligence*, 2021.

[11] Jorge Castillo, Phillip Rieger, Hossein Fereidooni, Qian Chen, and Ahmad Sadeghi. Fledge: Ledger-based federated learning resilient to inference and backdoor attacks. In *ACSAC*, 2023.

[12] Wen-Hsiung Chen, C. Harrison Smith, and S. C. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on communications*, 1977.

[13] Ege Erdogan, Alptekin Küpçü, and A Ercument Cicek. Splitguard: Detecting and mitigating training-hijacking attacks in split learning. In *Workshop on Privacy in the Electronic Society*, 2022.

[14] Hossein Fereidooni, Alessandro Pegoraro, Phillip Rieger, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning. In *NDSS*, 2024.

[15] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *USENIX Security*, 2022.

[16] Jiayun Fu, Xiaojing Ma, Bin B Zhu, Pingyi Hu, Ruixin Zhao, Yaru Jia, Peng Xu, Hai Jin, and Dongmei Zhang. Focusing on pinocchio's nose: A gradients scrutinizer to thwart split-learning hijacking attacks using intrinsic attributes. In *NDSS*, 2023.

[17] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *RAID*, 2020.

[18] Xinben Gao and Lan Zhang. PCAT: Functionality and data stealing from split learning by Pseudo-Client attack. In *USENIX Security*, 2023.

[19] Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A Camtepe, Hyoungshick Kim, and Surya Nepal. End-to-end evaluation of federated learning and split learning for internet of things. *arXiv preprint arXiv:2003.13376*, 2020.

[20] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

[22] Ying He, Zhili Shen, Jingyu Hua, Qixuan Dong, Jiacheng Niu, Wei Tong, Xu Huang, Chen Li, and Sheng Zhong. Backdoor attack against split neural network-based vertical federated learning. *IEEE Transactions on Information Forensics and Security*, 2023.

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

[24] Sanjay Kariyappa and Moinuddin K Qureshi. Exploit: Extracting private labels in split learning. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. IEEE, 2023.

[25] Torsten Krauß and Alexandra Dmitrienko. Mesas: Poisoning defense for federated learning resilient against adaptive attackers. In *CCS*, 2023.

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

[27] Kavita Kumari, Phillip Rieger, Hossein Fereidooni, Murtuza Jadliwala, and Ahmad-Reza Sadeghi. Baybfed: Bayesian backdoor defense for federated learning. In *IEEE S&P*. IEEE Computer Society, 2023.

[28] Huimin Li, Phillip Rieger, Shaza Zeitouni, Stjepan Picek, and Ahmad-Reza Sadeghi. Flairs: Fpga-accelerated inference-resistant & secure federated learning. *arXiv preprint arXiv:2308.00553*, 2023.

[29] Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. *arXiv preprint arXiv:2102.08504*, 2021.

[30] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[31] Junlin Liu and Xinchen Lyu. Clustering label inference attack against practical split learning. *arXiv preprint arXiv:2203.05222*, 2022.

[32] Junlin Liu, Xinchen Lyu, Qimei Cui, and Xiaofeng Tao. Similarity-based label inference attack against training and inference of split learning. *IEEE Transactions on Information Forensics and Security*, 2024.

[33] Yunlong Mao, Zexi Xin, Zhenyu Li, Jue Hong, Qingyou Yang, and Sheng Zhong. Secure split learning against property inference, data reconstruction, and feature space hijacking attacks. In *European Symposium on Research in Computer Security*. Springer, 2023.

[34] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative Machine Learning without Centralized Training Data. Google AI, 2017.

[35] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning Differentially Private Language Models Without Losing Accuracy. In *ICLR*, 2018.

[36] Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. Byzantine-Robust Federated Machine Learning through Adaptive Model Averaging. In *arXiv preprint:1909.05125*, 2019.

[37] Ahmed Nasir, Natarajan T, and R Rao Kamisetty. Discrete cosine transform. *IEEE Transactions on Computers*, 1974.

[38] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *CCS*, 2021.

[39] Maarten G. Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

[40] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, 2019.

[41] Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Crowdguard: Federated backdoor detection in federated learning. In *NDSS*, 2024.

[42] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *AAAI*, 2020.

[43] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending Against Poisoning Attacks in Collaborative Deep Learning Systems. In *ACSAC*, 2016.

[44] Ilia Shumailov, Zakhar Shumaylov, Dmitry Kazhdan, Yiren Zhao, Nicolas Papernot, Murat A Erdogdu, and Ross J Anderson. Manipulating sgd with data ordering attacks. *NeurIPS*, 2021.

[45] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*. Computational and Biological Learning Society, 2015.

[46] Abhishek Singh, Praneeth Vepakomma, Otkrist Gupta, and Ramesh Raskar. Detailed comparison of communication efficiency of split learning and federated learning. *arXiv preprint arXiv:1909.09145*, 2019.

[47] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *International joint conference on neural networks*. IEEE, 2011.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE conference on computer vision and pattern recognition*, 2015.

[49] Behrad Tajalli, Oğuzhan Ersoy, and Stjepan Picek. On feasibility of server-side backdoor attacks on split learning. In *IEEE Security and Privacy Workshops (SPW)*. IEEE, 2023.

[50] Chandra Thapa, Mahawaga Arachchige Pathum Chamikara, and Seyit A Camtepe. Advancements of federated learning towards privacy preservation: from federated learning to split learning. *Federated Learning Systems: Towards Next-Generation AI*, pages 79–109, 2021.

[51] Daniel J Velleman. *How to prove it: A structured approach*. Cambridge University Press, 2019.

[52] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[53] Ning Wang, Yang Xiao, Yimin Chen, Yang Hu, Wenjing Lou, and Y Thomas Hou. Flare: defending federated learning against model poisoning attacks via latent space representations. In *Asia Conference on Computer and Communications Security*, 2022.

[54] Zhongde Wang. Fast algorithms for the discrete w transform and for the discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984.

[55] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *IEEE conference on computer vision and pattern recognition*, 2021.

[56] Alexander Wong, Mohammad Javad Shafiee, and Michael St Jules. Micronnet: A highly compact deep convolutional neural network architecture for real-time embedded traffic sign classification. *IEEE Access*, 6:59803–59810, 2018.

[57] Xiao, Wu Xiaolin, and Liu Bolin. A study on quantization effects of dct based compression. *IEEE International Conference on Image Processing (ICIP)*, 2017.

[58] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[59] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *ICLR*, 2020.

[60] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*. Springer, 2019.

[61] Xin Yang, Jiankai Sun, Yuanshun Yao, Junyuan Xie, and Chong Wang. Differentially private label protection in split learning. *arXiv preprint arXiv:2203.02073*, 2022.

[62] Ziyuan Yang, Yingyu Chen, Huijie Huangfu, Maosong Ran, Hui Wang, Xiaoxiao Li, and Yi Zhang. Robust split federated learning for u-shaped medical image networks. *arXiv preprint arXiv:2212.06378*, 2022.

[63] Fangchao Yu, Lina Wang, Bo Zeng, Kai Zhao, Zhi Pang, and Tian Wu. How to backdoor split learning. *Neural Networks*, 168:326–336, 2023.

[64] Fangchao Yu, Bo Zeng, Kai Zhao, Zhi Pang, and Lina Wang. Chronic poisoning: Backdoor attack against split learning. In *AAAI*, 2024.

[65] Sergey Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[66] Chenchen Zhou, Hongbo Cao, Yingying Zhao, Sicong Zhao, and Yongqi Sun. Label inference attack based on soft label towards two-party split learning. In *IEEE International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*. IEEE, 2023.

[67] Xiaochen Zhu, Xinjian Luo, Yuncheng Wu, Yangfan Jiang, Xiaokui Xiao, and Beng Chin Ooi. Passive inference attacks on split learning via adversarial regularization. *arXiv preprint arXiv:2310.10483*, 2023.

APPENDIX

A. Deep Neural Network (DNN)

A DNN is a mathematical function denoted as $F(X; W)$, with X representing the input data samples and W denoting the network's parameters (comprising weights and biases). This network is structured into several layers, denoted as F_i, where $i \in \{1, \cdots, L\}$. The first layer, the input layer, is labeled as F_1, while the final layer, termed the output layer, is designated as F_L. The intermediate layers are commonly referred to as hidden layers. In a feed-forward neural network, data moves in a unidirectional path, starting from the input layer, traversing through the hidden layers, and ultimately reaching the output layer.
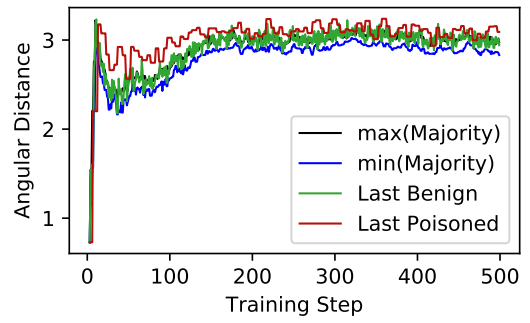
B. Poisoning Attack

In machine learning security, poisoning attacks represent a scheme in which model parameters are intentionally manipulated during or after training to introduce abnormal behaviors. Targeted or backdoor attacks alter the DNN's training stealthily to generate specific mispredictions when the model is presented with inputs containing predetermined triggers. For example, a trigger could be a red pixel placed in the upper left corner of an input image, which must be incorporated into the training dataset through data poisoning, and the abnormal behavior would be the classification of all samples with the red pixel into a specific predetermined class. The success rate for the backdoor attack is calculated based on the prediction performance on triggered data, denoted as Backdoor Accuracy (BA), while maintaining the expected behavior on benign inputs, as indicated by a high model or main-task accuracy (MA).
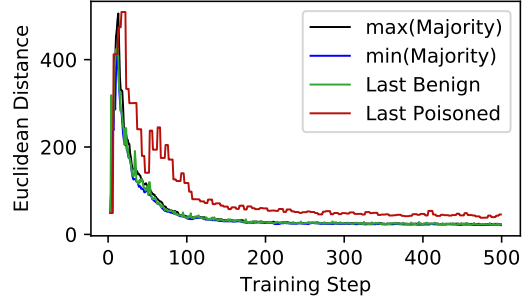
Following Fig. 1, we can see how an adversary must manipulate one or more clients to execute backdoor attacks in the Split Learning configuration. The adversary trains the poisoned local head and tail portion to poison the backbone in the central server and the subsequent head and tail training of the future clients. The critical objective is to poison the U-shaped aggregated model's prediction without incurring any noticeable behavior change to the server backbone and the other clients' head and tail, thus undermining the integrity and reliability of the entire system [22], [6].

C. Discrete Cosine Transform

In signal processing, the Discrete Cosine Transform (DCT) [37] is employed to break down a signal into its frequency components, providing insights into the underlying dynamics and transitions within the signal[57]. The DCT takes a sequence of numbers and represents them as a combination

(a) Rotational Distance Scores

(b) Frequency Distance Scores

Fig. 8: Rotational and Frequency distance scores for an IID-rate of 1.0.

of simple wave patterns (sinusoids) with different frequencies and sizes.

Mathematically, DCT transformations are invertible functions that convert an input sequence of N real numbers into the coefficients (values that multiply the wave patterns) of N orthogonal cosine basis functions (wave patterns independent of each other) with increasing frequencies. The DCT components (the coefficients) are ordered by significance (importance), with the first coefficient representing the sum of the input sequence normalized by length. Lower-order coefficients correspond to lower signal frequencies (slower repeating waves) and indicate the sequence's patterns (general trends or shapes in the data). These are known as low-frequency components. The 2-D DCT of a signal S (e.g., a matrix of size N by M) at frequencies k and l ($X(k,l)$), is given by the following equation [12], [54], [4], [14].

$$X(k,l) = a_k a_l \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} S(m,n) cos\left(\frac{k\pi}{2M}(2m+1)\right) cos\left(\frac{l\pi}{2N}(2n+1)\right)$$
(8)

Where a_k, a_l are dependent on the values of k, and l, with the following rules:

$$a_k = \begin{cases} \sqrt{\frac{2}{MN}} & \text{for } k = 0 \\ 1 & \text{for } k = 1, 2, ..., M-1 \end{cases}$$
(9)

$$a_l = \begin{cases} \sqrt{\frac{2}{MN}} & \text{for } l = 0 \\ 1 & \text{for } l = 1, 2, ..., N-1 \end{cases}$$
(10)

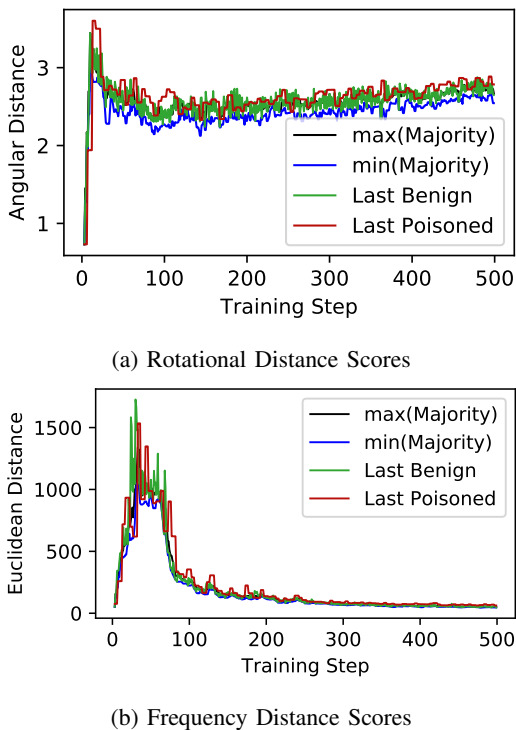(a) Rotational Distance Scores

(b) Frequency Distance Scores

Fig. 9: Rotational and Frequency distance scores for an IID-rate of 0.8.

As we elaborate in Sect. V-C, recent work showed that in the early stages of the training, mostly the low-frequency components change while the high-frequency components change during the fine-tuning, when the model is already close to convergence [40], [60]. In Sect. V-C, we describe how this can be used to help detect models with injected backdoor behavior.

D. Effectiveness of the Angular Distance

SafeSplit's selection mechanism relies on the Euclidean distance in the frequency domain and the rotational distance among server model states. SafeSplit uses the assumption that the majority of clients are benign. We establish a norm by accepting a set of $N/2 + 1$ clients with the least Euclidean and rotational distances. Fig. 8 illustrates a distinct gap between malicious and benign scores for an IID-rate of 1.0 and Fig. 9. The figures visualize how the different perspectives (static and dynamic) complete it each other. As long as malicious scores exceed the maximum score of the majority set for at least one of both metrics, a malicious client remains unselected, preventing any backdoor injection. Only occasionally, a benign score may surpass the maximum of the majority set, resulting in its exclusion. This exclusion can impact MA, albeit minimally, as demonstrated in Table II. Our evaluation reveals that the score gap between benign and malicious instances is more pronounced during the model's learning phase, gradually diminishing as MA converges. Hence, halting training at MA convergence proves advantageous.

E. Impact of Misclassifications on BA

For pixel-trigger backdoors the BA is often not 0%, even when the model is benign and not poisoned. As outlined by Fereidooni *et al.*, this phenomenon occurs due to the misclassification of samples by the model, if the MA is not 100%. Especially, samples that are similar to the backdoor target label are, if not recognized correctly, likely to be incorrectly classified as the backdoor target, even when they are independent of the presence of the backdoor trigger. For example, in the case of truck images, the model can misclassify them as car images. If a triggered sample is misclassified as the backdoor target by coincidence, it is counted as successful backdoor activation and increases the BA, although the model was trained to recognize the backdoor trigger [14].

To illustrate this, we show in Fig. 10 a confusion matrix for a poisoned test dataset evaluated on a benign model. In this case, the backdoor trigger, represented by a red rectangle, is intended to cause the model to predict class 2 (bird).

As the figure shows, although the model does not contain any backdoor, in 946 cases, label 2 is predicted, resulting in a BA of 10.5%.

Notably, the red rectangle also covers parts of the image, thus affecting the model's ability to recognize the actual object.

F. Evaluation of Further Attacks

In addition, we evaluated SafeSplit for a label swapping attack, where the predictions for all samples of two classes shall be swapped, thus realizing a mixture of backdoor and untargeted attack. For this experiment, we measured the effectiveness using the attack success rate (ASR), counting the fraction of samples belonging to both classes where the model predicts the swapped labels. We performed this experiment for all pairs of classes using the CIFAR-10 dataset. We observed that SafeSplit effectively reduces the ASR in average from 22.7% to 3.8%. Notably, the ASR for the undefended model (22.7%) is significantly smaller than for regular backdoor attacks such as the semantic backdoor. This is caused by the clean samples that are part of the benign clients' datasets, allowing them to reduce the attack's impact, while for backdoors, such as the semantic backdoor, benign clients pose only a negligible number of samples. Further, similar to the pixel-pattern backdoor, the BA for a benign model is not exactly 0% but 3.4% (cf. Sect. E).
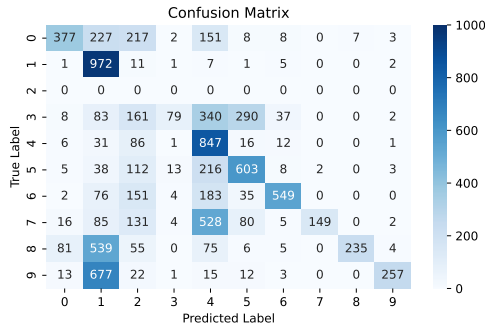
Fig. 10: Confusion matrix for triggered inputs applied on a benign model.

While also untargeted attacks fall outside our threat model's scope, since they can be straightforwardly detected and inherently differ from backdoor attacks, we tested a loss-maximization attack and showed that SafeSplit mitigates the attack and maintains the Main Task Accuracy (MA) at 64%.

G. Runtime Evaluation

We evaluated the runtime performance of SafeSplit and its individual components in dependence of the client number to determine its scalability. Since the defense is executed before every training, we performed for each client number an experiment running 50 rounds and measured the individual runtimes every time the defense was called. Thus, depending on the client number, we obtained measurements of 245 (5 clients) and 1470 (30 clients). Notably, we omitted measurements from the first round until every client provided at least one model. We trimmed the 5% highest and smallest values and averaged the remaining values. The results are shown in Fig. 11. As the figure shows, the runtime of SafeSplit scales linearly as the most time-consuming operations, such as the frequency transformation, are executed once per model. Notably, the runtime for the frequency transformation is plotted as an individual curve but also included in the curve for the Frequency Analysis.
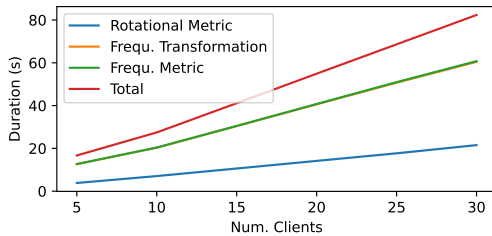

Fig. 11: Evaluation times of SafeSplit and its individual components for different client numbers.

Notably, all code was written in Python and no parallelization was used, although operations such as the frequency transformation can be easily parallelized. While runtime engineering is out of the scope of this work, this demonstrates the practical applicability of SafeSplit.