IBAS:Imperceptible Backdoor Attacks in Split Learning with Limited Information

Peng Xi, Shaoliang Peng*, Wenjuan Tang

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
xipeng@hnu.edu.cn, wenjuantang@hnu.edu.cn, slpeng@hnu.edu.cn

Abstract

Split learning, as a distributed learning framework, effectively addresses the issue of limited computing resources. However, despite achieving a separation of data and computation, recent studies have pointed out that this framework still faces two major security challenges: privacy leakage and model security. Most current research focuses on the problem of privacy leakage, emphasizing how to prevent malicious servers from recovering or inferring the client's private data. However, the issue of model security in split learning has not received sufficient attention. This paper reveals the vulnerability of split learning to backdoor attacks. Since split learning cannot access client data directly, it can only guide the client model to incorporate backdoors through gradients. To address this issue, we design an attack framework that modifies intermediate activations to influence the gradients. We designed a parrot model that learns the client's feature space, enabling the server to obtain the intermediate activations of poisoned data. During the forward pass, some of the intermediate activations and labels transmitted from the client to the server are replaced with poisoned activations and target labels. This replacement method effectively integrates the backdoor task into the model while partially retaining the main task. This approach ensures that the main task is preserved while seamlessly embedding the backdoor task. Our attack framework minimizes reliance on client knowledge and ensures that the attack process remains undetectable by the client. Through extensive experiments, we demonstrated high attack success rates using triggers such as BadNet, SIG, Blended, and WaNet, while minimizing the impact on the main task.

Introduction

Artificial intelligence has demonstrated its potential to solve complex problems in various fields, such as automatic driving (Tan et al. 2024) and pathological diagnosis (Moor et al. 2023). However, these applications typically require vast amounts of data and computational resources. Split learning has been proposed to address data privacy and resource constraints (Poirot et al. 2019). In the split learning framework, a neural network is divided into client and server parts. The client is responsible for performing preliminary computations and sending intermediate activations to the server, which then completes more complex subsequent computations. Since the server cannot directly access the raw data and client model (Abuadbba et al. 2020), split learning is considered a secure and reliable computational method.

However, the server controls the direction of client optimization, and split learning may encounter model security issues. Backdoor attacks are a typical model security problem. Backdoor attacks cause the model to output a specified label for data containing a specific identifier, while leaving normal data unaffected. Typically, backdoor attacks are carried out by poisoning the training data, thereby embedding a backdoor into the model (Li et al. 2024). However, in split learning, the server does not have access to the training data. As a result, the server cannot implement a backdoor attack by poisoning the training data in split learning. In 2023, Tajalli (Tajalli, Ersoy, and Picek 2023) was the first to propose the possibility of injecting a backdoor on the server side in split learning. Although Tajalli proposed two attack frameworks to demonstrate split learning is susceptible to backdoor attacks, the experimental results demonstrated that the impact of backdoor attacks in split learning is significantly mitigated, indicating a relatively weak effect.

Inspired by the above work, we propose a question: Is there a type of backdoor attack that can adversely affect the functionality or performance of a split learning model? To answer this question, we analyze two major challenges that backdoor attacks in split learning face. Challenge 1: The simultaneous optimization of both the primary task and the backdoor task poses a significant challenge for the server during an attack in split learning scenarios. Challenge 2: The server needs to comprehend the output characteristics of the trigger after it passes through the client model to ensure the alignment between the trigger and the backdoor task.

To address the aforementioned challenges, we propose a novel attack strategy that introduces a backdoor task during model training by modifying intermediate activations. First, we train a parrot model to learn the client model's feature space. The poisoned data processed by the parrot model produces outputs that closely approximate those of the client model. We then replace the intermediate activations output by the client with the poisoned intermediate activations generated by the parrot model. Simultaneously, we change the labels corresponding to these modified intermediate activations to the target labels. During loss calculation, the un-

*Corresponding author

modified portion represents the main task, while the modified portion represents the backdoor task. Finally, through backpropagation, the client model is optimized to satisfy the requirements of both the main task and the backdoor task.

The main contributions of this paper are summarized as follows:

- We propose a general backdoor attack approach suitable for split learning. This framework supports multiple triggers, allowing attackers to customize them based on specific environmental conditions. Moreover, our framework requires only a small public dataset, resulting in very low attack costs.
- We propose a method for determining the client's model structure and demonstrate its feasibility through experiments and theoretical analysis. This method reduces the attacker's reliance on prior knowledge of the client, allowing for successful attacks even when the client's model structure is unknown. Moreover, other attack schemes that depend on client knowledge can benefit from our method as well.
- We comprehensively evaluate the effectiveness and generality of the backdoor attack framework. We tested the attack results under various conditions. Experimental results show that even with a public dataset of only 2000 samples, our attack framwork can achieve an attack success rate of nearly 90% across various attack scenarios. Furthermore, our framwork does not affect the performance of the main task, demonstrating its high stealthiness.

Related Work and Challenges

In this section, we mainly introduce the latest research advancements in model security for split learning and attack detection in split learning.

Security for Split Learning

Data reconstruction attacks and label inference attacks are significant data security challenges faced by split learning. UnSplit (Erdoğan, Küpçü, and Çiçek 2022), proposed by Ege Erdoğan, involves the server randomly initializing a model with the same architecture as the client model. Using a coordinate gradient descent strategy, the parameters are first updated by fixing the input and then updating the input by fixing the parameters. Through iterative optimization, the mean squared error between the model and the client model's output is minimized. Through this process, the attacker can gradually infer the client's original input data and model parameters. Liu (Liu et al. 2024) proposed a clustering-based label inference method. This method uses gradients and intermediate activation values as features, with each label represented by an auxiliary sample as the centroid. Subsequently, K-means clustering is applied to cluster the data, thus obtaining the predicted labels.

Backdoor attacks are a significant model safety issue currently faced in split learning. Backdoor attacks were initially proposed by BadNet (Gu et al. 2019), which implemented these in centralized training by adding poisoned data with triggers to the training set. This approach allows the model to function normally on clean data but to produce specific erroneous outputs on data containing triggers.

In 2023, Tajalli (Tajalli, Ersoy, and Picek 2023) first proposed two methods to implement backdoor attacks in split learning. In the first approach, the server processes two client networks simultaneously: a normal client network and a proxy client network, the latter being trained with poisoned data containing triggers. After performing forward propagation, the server calculates the average loss of these two networks to generate gradients, which are then sent back to both networks, potentially influencing the normal network with backdoor behaviors. The second approach involves training two split networks simultaneously, one with a clean dataset and the other with a poisoned dataset. The output from both models can be used to train an autoencoder, which transforms clean intermediate activations into poisoned activations. This autoencoder, when embedded into a normally trained network, can effectively insert a backdoor. Since both approaches yielded suboptimal results, Tajalli suggests that split learning offers some resistance to backdoor attacks.

Subsequently, Yu et al. proposed two attack framworks (Yu et al. 2024)(Yu et al. 2023), both drawing on the FSHA method (Pasquini, Ateniese, and Bernaschi 2021). FSHA introduces an active hijacking strategy, which can guide the client model's feature space into a fake model trained by the server.In the first attack framwork, a GAN network is used to hijack the client's feature space into a backdoored fake model. Therefore, the success rate of the attack and the accuracy of the main task both depend on the availability of a public dataset. If the public dataset is too small, it significantly impacts the accuracy of the main task.In the second attack framwork, the fake model first learns the client's feature space and then fine-tunes the model using a public dataset to enhance the backdoor. Finally, a GAN network is used to hijack the client's feature space. This method improves the accuracy of the main task, but since the client undertakes two tasks, it increases the risk of the attack being detected.

Existing Defense framworks

There are already some attack detection framworks for split learning. SplitGuard (Erdogan, Küpçü, and Cicek 2022) can detect whether the server is performing the expected task by having the client transmit incorrect labels. If the server is conducting a normal classification task, the gradients transmitted by the server will show reverse gradients due to the incorrect labels. However, if a GAN network is being used to hijack the feature space, the gradients transmitted by the server will not show significant changes. Fu proposed the SplitSpy diagnostic method (Fu et al. 2023), which can detect whether the client is performing classification tasks through continuous gradient observation. This method can identify changes in the client's tasks. Additionally, SplitSpy emphasizes that defense mechanisms should not heavily rely on hyperparameters, as an excessive number of hyperparameters can undermine the generality and stability of the solution.

To evade attack detection frameworks like SplitGuard and SplitSpy, it is essential to avoid using GAN networks. Addi-

tionally, it is crucial to prevent mid-training modifications to the client's tasks and to minimize the use of hyperparameters. These measures not only reduce the risk of detection by such methods but also maintain the generality and stability of the attack strategy.

Threat Model

The threat model is designed to create a universal attack framework for implanting backdoor in split learning models. Our attack framework adheres to the following requirements and assumptions:

- **Semi-honest attacks without model knowledge.** The attack is initiated by a semi-honest server, which does not require knowledge of the client model's structure or parameters but needs access to a limited set of data samples.
- **Protocol-following gradient attacks.** The server adheres to the split learning protocol, allowing attackers to modify client network parameters only indirectly through backward gradients.
- **Minimal impact of the main task.** To minimize the risk of detection, the training process should have minimal impact on the primary task.
- **Applies multiple trigger attacks.** The attack framework has a certain degree of versatility and is capable of implementing attacks on multiple types of triggers. This adaptability allows it to accommodate a variety of attack strategies, thus enhancing its applicability in different operating environments and scenarios.

The attack results of this threat model ensure that for any input data x_i combined with the trigger δ gets poisoned data:

$$x_i' = g(x_i, \delta)$$

This poisoned data satisfies:

$$F_s\left(F_c\left(x_i'\right)\right) = y_t$$

where y_t is the target label in the backdoor attack, indicating the specific output set by the attacker.

Approach to Backdoor Attacks

The server can guide the optimization direction of the client network by controlling the gradients. To direct the server towards optimizing the backdoor task, the loss calculation needs to include the loss of the backdoor task. In this section, we provide a detailed description of the backdoor attack framwork. The attack framwork includes three stages: (1) learning the client's feature space; (2) computing the intermediate activation values of the poisoned data mapping and modifying the intermediate activation values transmitted by the client; (3) using the modified intermediate activation values to calculate the loss, thereby obtaining the gradients to optimize the client model.

Learning client-side feature spaces

In this attack scenario, the server can only access the intermediate activations and their corresponding labels transmitted from the client, without knowing the specific structure and parameters of the client model. This setup enhances the security and data privacy of the client model. To accurately guide the client model towards optimizing the backdoor, the attacker must understand the mapping relationship between the input data and the intermediate activations. Therefore, we need to learn the client's feature space and understand the intermediate activations of the poisoned data after passing through the client.

To address this, we propose training a parrot model F_p. The parrot model mimics the client's behavior to learn the distribution of the client's feature space, thereby meeting the following conditions:

$$F_c\left(x_i\right) \approx F_p\left(x_i\right)$$

We propose the following strategy: During the standard training cycles of split learning, the client model F_c collaborates with the server model F_s for learning. After completing each training cycle, to ensure that F_p is trained in the same server environment as F_c, we need to freeze the parameters of F_s. The primary purpose of this is to maintain consistency in F_s while training F_p. Next, we use a public dataset D_s to train the parrot model F_p and employ the same loss function. Figure 1 illustrates the training process of the parrot model. Although the learning rate of the parrot model is a hyperparameter, we only need to ensure that the parameter is the same as the learning rate of the client model.

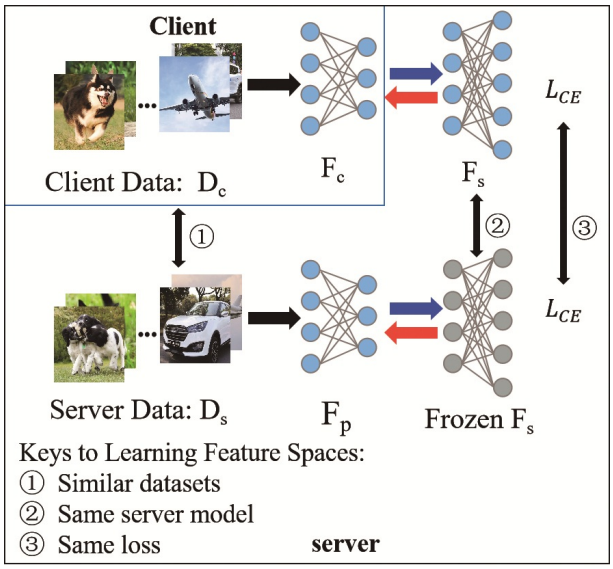

Figure 1: The server learns the feature space of the client model.

The parrot model shares the same parameters with the server model to which the client model is connected. Given that the server model possesses a robust capability to recognize the intermediate activation values output by the client model, optimizing the parrot model also results in its output of intermediate activation values more closely approximating the feature space of the client model. Additionally, the parrot model is trained using a similar dataset and the

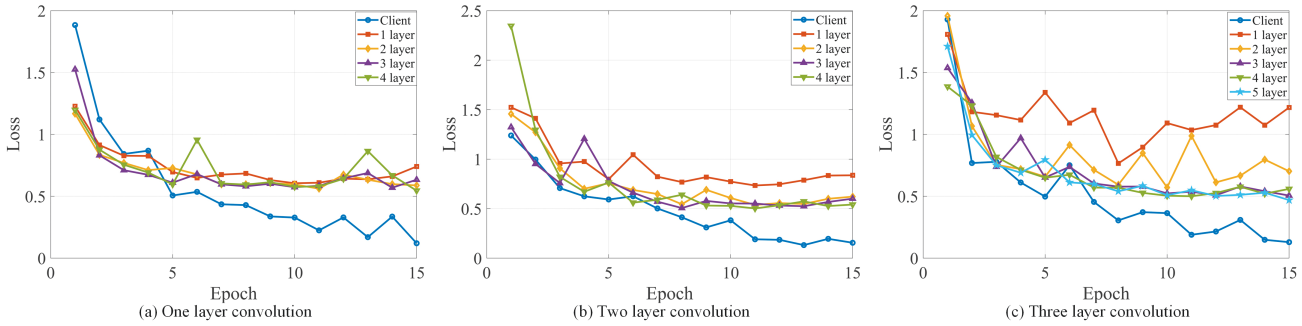

Figure 2: Loss under different parrot models.

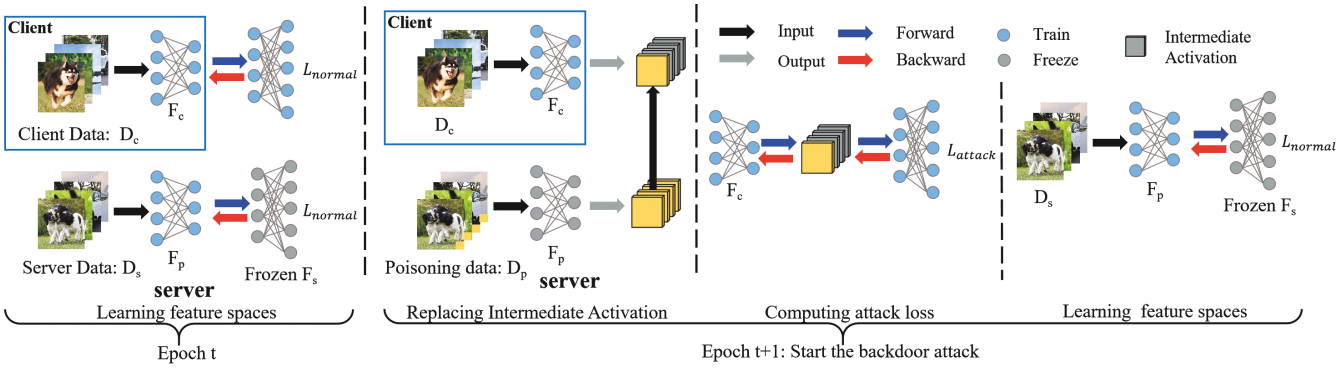

Figure 3: Attack Framework.

same loss function. The consistency of these three conditions—shared parameters, similar dataset, and identical loss function—is crucial for the successful training of the parrot model.

The structure of the parrot model has a decisive impact on learning the client's feature space. Therefore, choosing the parrot model structure is very important. When the client's model structure is known, the parrot model can be trained using the same structure. When the client's structure is unknown, we propose a method to determine the client's model structure. Figure 2 shows the comparison of loss curves between parrot models with different numbers of convolutional layers and the client model. The loss decreases as the number of layers increases. When the number of layers in the parrot model is greater than or equal to the number of layers in the client model, the loss no longer changes. This is because shallow neural networks struggle to simulate the features of deep neural networks, whereas deep networks can effectively mimic the feature space of shallow networks. Therefore, we can determine the client's model structure by analyzing the loss. In training the parrot model, we train multiple parrot models from shallow to deep layers. In sustained three epochs, satisfy:

$$\text{loss}_1 > \text{loss}_2 > \ldots > \text{loss}_s \cong \text{loss}_{s+1}$$

It can be concluded that the structure of the model F_p^s is equal to the structure of the client model.

Replace intermediate activation

After t epochs, we determine the parrot model Fsbest, which most closely approximates the functionality of the client model F_c. Then, we replace the intermediate activation for the next epoch. We add the trigger δ to the public dataset D_s, obtaining a poisoned dataset D_p. The poisoned intermediate activation values are then computed using the parrot model as follows:

$$H_p = F_p\left(D_p\right) \approx F_c\left(D_p\right)$$

Where H_p is $H_p = \{h_1^p, h_2^p, \ldots, h_m^p\}$.

In the t+1-th epoch, the client transmits the intermediate activation values $H = \{h_1, h_2, \ldots, h_n\}$ and labels $Y = \{y_1, y_2, \ldots, y_n\}$ to the server. Then, we randomly select a portion of the activation values in H_p to replace the corresponding values in H. Define a collection of indexes $\{1, 2, \ldots, n\}$ denotes k indexes randomly selected from I:

$$I = \{i_1, i_2, \ldots, i_k \mid i_j \in \{1, 2, \ldots, n\}\}$$

Specifically, the new intermediate activation set and label set can be represented as:

$$H' = \{h_i^p \mid i \in I\} \cup \{h_j \mid j \in \{1, 2, \ldots, n\} \backslash I\}$$

$$Y' = \{y_t \mid i \in I\} \cup \{y_j \mid j \in \{1, 2, \ldots, n\} \backslash I\}$$

Where k is the number of intermediate activation that are replaced. k/n is the poisoning rate, i.e., the ratio of the number of replaced values to the total number. During training, the poisoned intermediate activation can be evenly distributed across each batch.

Calculating Attack Loss

Before the $t+1$th epoch, the client sends intermediate activation values and labels to the server. The loss calculated by the server is:

$$L_{normal} = L\left(F_s(H); Y\right) \tag{1}$$

$$L_{normal} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\widehat{y}_i\left[y_i\right]\right) \tag{2}$$

When we use the modified intermediate activation values H' and the new labels Y', the new attack loss is defined as:

$$L_{attack} = L\left(F_s\left(H'\right); Y'\right) \tag{3}$$

$$L_{\text{attack}} = -\frac{1}{N}\left\{\sum_{i\in I}\log\left(\widehat{y}_i\left[y_t\right]\right) + \sum_{j\notin I}\log\left(\widehat{y}_j\left[y_j\right]\right)\right\} \tag{4}$$

By comparing L_{normal} and L_{attack}, we can see that L_{attack} includes both the loss of the main task and the loss of the backdoor task. Through the combination of these two parts, we ensure the optimization of the main task while also optimizing the backdoor task. From the above attack process, it is evident that the malicious server's attack behavior is imperceptible to the client. Algorithm 1 is the pseudocode for our attack method.

Algorithm 1: Backdoor Attack in Split Learning

Require: Number of training iterations: N, Target class: y_t, Trigger: δ, Initial models: F_c and F_s, Datasets: D_c and D_s

Ensure: Poisoned dataset: $D_p = f(D_s, \delta)$

for $i = 1$ **to** t **do**
  Optimize F_c and F_s using D_c
  Freeze F_m, optimize F_p using D_s
end for
for $i = t+1$ **to** N **do**
  Compute intermediate activations for D_c using F_c: $H = \{h_1, h_2, \ldots, h_n\}$
  Compute intermediate activations for D_p using F_p: $H_p = \{h_1^p, h_2^p, \ldots, h_m^p\}$
  Replace part of H with H_p to obtain H' and Y': $H' = \{h_1^p, h_2, \ldots, h_n\}$, $Y' = \{y_t, y_2, \ldots, y_n\}$
  Calculate attack loss: $L_{attack} = L(F_s(H'); Y')$
  Update F_c and F_s using gradients from L_{attack}
end for
return F_c and F_s

Experiments

In this section, we evaluate the effectiveness of our attack framwork and set up tests for various trigger attacks to assess the generality of the attack.

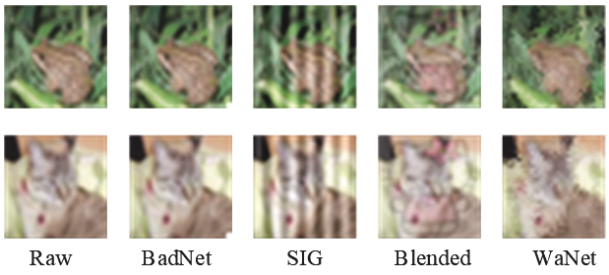

Figure 4: The clean samples and backdoor samples.

Experiment Setup

Our experiments were conducted on the CIFAR-10 dataset. We utilized 40,000 data points as the client training set and 10,000 data points as the public dataset. The experiments were based on a modified ResNet18 architecture to test the attacks under various client network conditions. We created three different model structures $Model_1$, $Model_2$ and $Model_3$. In our experiments, we defaulted the size of the public dataset to 10,000 data points, the model used was $model_3$, and the batch size was set to 32. The optimizer for all models uses Adam and a default learning rate. Additionally, in each batch, we modified only one intermediate activation. The experimental setup initiates attacks from the 11th epoch onwards, using the first 10 epochs to determine the structure of the parrot model. We also tested four backdoor attack strategies: BadNet (Gu et al. 2019), SIG (Barni, Kallas, and Tondi 2019), Blended (Chen et al. 2017) and WaNet (Nguyen and Tran 2021). These triggers encompass a range of types, from local to global, dynamic to static, and visible to invisible triggers. Figure 4 shows these triggers.

Evaluation metrics. To evaluate the efficacy of our backdoor attack strategies, we considered two key metrics: the accuracy on clean data (ACC) and the attack success rate (ASR), which denotes the probability that poisoned data are classified as the target label. An optimal backdoor attack strategy should not compromise ACC to enhance ASR. Therefore, a superior attack framwork should simultaneously exhibit high ACC and ASR, embodying a balanced approach which is crucial for assessing the effectiveness of backdoor techniques.

Results and Analysis

Comparison of approaches. We compared our approach against existing split learning framworks. Table 1 presents the attack results. Our experiments demonstrate that only our framwork can successfully integrate a backdoor into the model without compromising the main task's performance. In contrast, other framworks failed to balance ASR and ACC. Tests conducted without any attacks showed that the average accuracy of the main task in split learning was 87.20%. Our attack had minimal impact on the main task. Our attack can even achieve the effect of poisoning the training data. Moreover, our framwork is versatile, capable of implementing various triggers proposed in centralized attack scenarios. This demonstrates its general applicability

	Data poisoning		Surrogate Client		Injector Autoencoder		Chronic Poisoning		Ours	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNet	86.23	99.12	80.40	11.91	10.49	92.42	38.74	83.80	86.22	96.92
SIG	85.52	98.74	81.72	17.55	11.12	56.31	39.46	86.44	86.32	98.62
Blended	84.50	98.91	80.79	14.60	12.37	44.97	56.85	89.93	85.11	84.34
WaNet	84.76	99.92	65.28	2.62	12.85	80.29	10.00	100	84.24	77.85

Table 1: Comparative Experiments on Attack framworks.

and effectiveness in maintaining operational integrity while embedding malicious functionalities.

Starting rounds of attack. We conducted attacks under two assumptions: in one scenario, the server is aware of the client model's structure from the outset; in the other, the server is initially unaware of the client model's structure and must determine it after 10 epochs. Figure 5 presents the attack effects under these two scenarios, including the main task accuracy and the attack success rate when the parrot model and the client model are integrated with the server model. The results indicate that our approach of modifying intermediate activation values achieves substantial attack effectiveness, whether the attack is initiated from the start or partway through the process. Additionally, the accuracy of the parrot model and the client model demonstrates that the parrot model effectively mimicked the client's feature space.

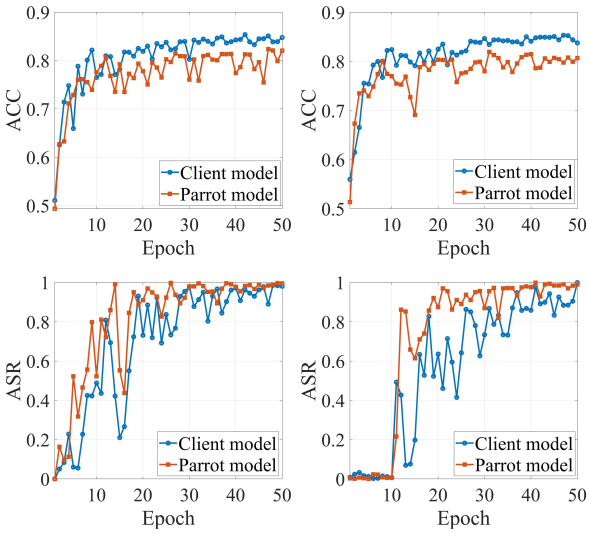

Figure 5: Comparison of attack effects with different starting rounds: left (starting at round 0) and right (starting at round 10).

Effects of parrot model structure. Initially, this study explores the impact of parrot model architecture selection on the effectiveness of simulation attacks. For this purpose, we conducted detailed tests on $Model_3$ to evaluate the performance of parrot models with varying numbers of layers in mimicking the client model. The experimental results, as shown in Table 2, reveal a key insight: the effectiveness of the attack significantly improves when the number of layers in the parrot model equals or exceeds those of the client model. This phenomenon can be attributed to the parrot model's ability to more deeply learn and replicate the client model's feature space when its number of layers matches or surpasses that of the client. Especially in the multi-layer architecture of neural networks, additional layers help in more intricately capturing and simulating the underlying features of the client network. This is because deeper network structures can learn more complex abstract representations, thereby enhancing the model's fit to higher-level features. Particularly when facing computational constraints, shallower networks struggle to effectively simulate deep feature representations. Based on these findings, we recommend selecting the parrot model's number of layers according to the client model's layer count and the server's specific needs to optimize the effectiveness of backdoor attacks.

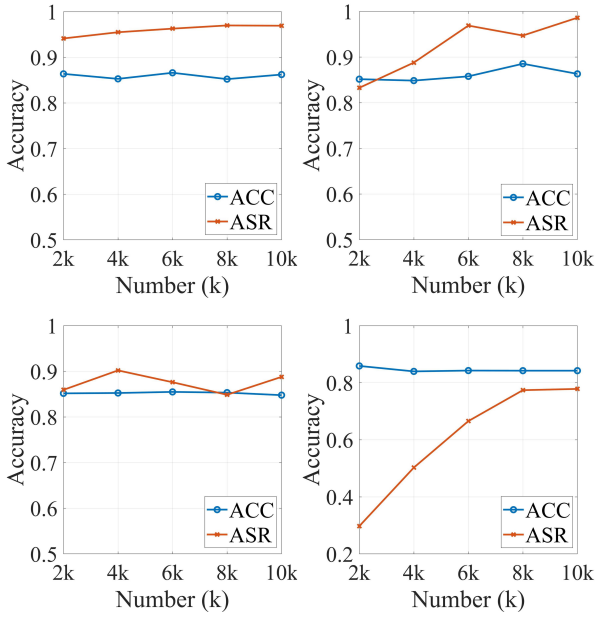

Figure 6: Attack performance with different sample sizes of public datasets. From left to right and top to bottom: BadNet, SIG, Blended, and WaNet.

Impact of Split Strategy on Attacks. Split learning adapts to diverse requirements and environments by adjusting the flexibility of its splitting strategies. Employing different splitting strategies results in changes to the network architecture, which subsequently impacts both the training process and the effectiveness of attacks. We tested

	One layer		Two layers		Three layers		Four layers		Five layers	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
BadNet	84.71	14.13	87.26	94.33	86.22	96.92	83.50	98.69	86.84	89.99
SIG	85.35	19.47	85.44	94.43	86.32	98.62	85.44	99.70	85.62	98.42
Blended	84.68	10.54	85.52	71.80	84.43	71.75	84.70	80.83	84.76	91.84
WaNet	85.34	2.93	83.85	2.496	85.11	84.34	84.70	85.83	86.05	84.03

Table 2: Attack performance of the parrot model with different convolutional layers under three convolutional layers on the client side.

	$Model_1$		$Model_2$		$Model_3$	
	ACC	ASR	ACC	ASR	ACC	ASR
BadNet	85.48	98.42	86.07	98.82	86.22	86.92
SIG	82.79	97.19	84.48	97.42	86.32	98.62
Blended	83.64	93.21	84.82	92.17	85.11	84.34
WaNet	83.09	63.78	83.63	78.90	84.24	77.85

Table 3: Attack Performance of Different Segmentation Strategies.

three different splitting strategies, with the results recorded in Table 3. Our research shows that the attack framework we designed is adaptable to various splitting strategies and achieves the anticipated performance in both the main task and backdoor attacks.

Impact of datasets and poisoning rates on attacks. Our attack strategy relies only on a small public dataset for training and optimization. To assess the impact of the public dataset size on the attack success rate, we tested the success rates of the SIG attack with varying data volumes. The experimental results are displayed in Figure 6. These results indicate that a reduction in data volume decreases the attack success rate. This decrease occurs because a smaller dataset limits the parrot model's ability to learn the spatial features of the client model. However, this decrease does not impact the accuracy of the main task. Our attack has a high success rate despite the fact that the public dataset has only 2000 samples. This shows that our attack framwork has a significant advantage in data utilization efficiency and does not rely on a large amount of public data. These findings highlight the superior performance of our attack framwork, which remains effective even under resource constraints.

Furthermore, we explored the effect of the poisoning rate on the attack's effectiveness. The poisoning rate refers to the proportion of data in the dataset that is injected with backdoor triggers, which is directly linked to the stealthiness of the attack. We tested the effectiveness of the attack with different batch sizes. We modified one intermediate activation per batch, resulting in a poisoning rate of 1/batchsize. Our experimental results, as illustrated in Figure 7, demonstrate that the attack maintains a high success rate even at a poisoning rate as low as 1/128. In centralized backdoor attacks, a poisoning rate below one percent is often considered extremely low, yet our strategy exhibits good effectiveness even at such a reduced rate. This finding suggests that our attack strategy can maintain good effectiveness even at very low poisoning rates, with minimal impact on the performance of the main task. These results underscore the stealthiness and efficiency of our attack framwork, indicating that it would be difficult to difficult to implement in practical applications.

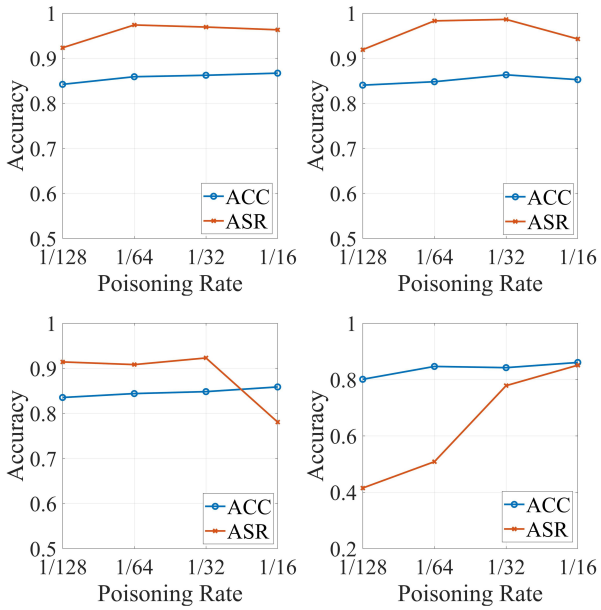

Figure 7: Attack performance at different public poisoning rates. From left to right and top to bottom: BadNet, SIG, Blended, and WaNet.

Conclusion

In this paper, we present a universal backdoor attack framework that reveals potential vulnerabilities in the security of federated learning models. This framework introduces backdoor tasks by modifying intermediate activation values during the training process, achieving a high attack success rate while maintaining the performance of the primary task. Furthermore, our approach requires the configuration of only one hyperparameter, which aligns with the learning rate of the clients, effectively simplifying the implementation process and reducing complexity. Notably, our method does not rely on GANs and does not require adjustments to the clients' tasks during training, making it difficult for existing defense mechanisms, such as SplitGuard and SplitSpy, to detect our attacks. These design choices significantly enhance the stealthiness and practicality of our approach.

Acknowledgments

This work was supported by NSFC-FDCT Grants 62361166662; National Key R&D Program of China 2023YFC3503400, 2022YFC3400400; National Natural Science Foundation of China No. 62202157; The Innovative Research Group Project of Hunan Province 2024JJ1002; Key R&D Program of Hunan Province 2023GK2004, 2023SK2059, 2023SK2060; Top 10 Technical Key Project in Hunan Province 2023GK1010; Key Technologies R&D Program of Guangdong Province (2023B1111030004 to FFH). The Funds of State Key Laboratory of Chemo/Biosensing and Chemometrics, the National Supercomputing Center in Changsha, and Peng Cheng Lab.

References

Abuadbba, S.; Kim, K.; Kim, M.; Thapa, C.; Camtepe, S. A.; Gao, Y.; Kim, H.; and Nepal, S. 2020. Can We Use Split Learning on 1D CNN Models for Privacy Preserving Training? In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '20, 305–318. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367509.

Barni, M.; Kallas, K.; and Tondi, B. 2019. A New Backdoor Attack in CNNS by Training Set Corruption Without Label Poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, 101–105.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Erdogan, E.; Küpçü, A.; and Cicek, A. E. 2022. SplitGuard: Detecting and Mitigating Training-Hijacking Attacks in Split Learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, WPES'22, 125–137. New York, NY, USA: Association for Computing Machinery. ISBN 9781450398732.

Erdoğan, E.; Küpçü, A.; and Çiçek, A. E. 2022. UnSplit: Data-Oblivious Model Inversion, Model Stealing, and Label Inference Attacks against Split Learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, WPES'22, 115–124. New York, NY, USA: Association for Computing Machinery. ISBN 9781450398732.

Fu, J.; Ma, X.; Zhu, B. B.; Hu, P.; Zhao, R.; Jia, Y.; Xu, P.; Jin, H.; and Zhang, D. 2023. Focusing on Pinocchio's Nose: A Gradients Scrutinizer to Thwart Split-Learning Hijacking Attacks Using Intrinsic Attributes. In *NDSS*.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7: 47230–47244.

Li, Y.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2024. Backdoor Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1): 5–22.

Liu, J.; Lyu, X.; Cui, Q.; and Tao, X. 2024. Similarity-Based Label Inference Attack Against Training and Inference of Split Learning. *IEEE Transactions on Information Forensics and Security*, 19: 2881–2895.

Moor, M.; Banerjee, O.; Abad, Z. S. H.; Krumholz, H. M.; Leskovec, J.; Topol, E. J.; and Rajpurkar, P. 2023. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956): 259–265.

Nguyen, A.; and Tran, A. 2021. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*.

Pasquini, D.; Ateniese, G.; and Bernaschi, M. 2021. Unleashing the Tiger: Inference Attacks on Split Learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, 2113–2129. New York, NY, USA: Association for Computing Machinery. ISBN 9781450384544.

Poirot, M. G.; Vepakomma, P.; Chang, K.; Kalpathy-Cramer, J.; Gupta, R.; and Raskar, R. 2019. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*.

Tajalli, B.; Ersoy, O.; and Picek, S. 2023. On Feasibility of Server-side Backdoor Attacks on Split Learning. In *2023 IEEE Security and Privacy Workshops (SPW)*, 84–93.

Tan, K.; Wu, J.; Zhou, H.; Wang, Y.; and Chen, J. 2024. Integrating Advanced Computer Vision and AI Algorithms for Autonomous Driving Systems. *Journal of Theory and Practice of Engineering Science*, 4(01): 41–48.

Yu, F.; Wang, L.; Zeng, B.; Zhao, K.; Pang, Z.; and Wu, T. 2023. How to backdoor split learning. *Neural Networks*, 168: 326–336.

Yu, F.; Zeng, B.; Zhao, K.; Pang, Z.; and Wang, L. 2024. Chronic Poisoning: Backdoor Attack against Split Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 16531–16538.