BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 01/01 pp1–14
DOI: 10.26599/BDMA.2024.9020035
Volume x, Number x, xxxx xxxx

# Backdoor Attack to Giant Model in Fragment-Sharing Federated Learning

Senmao Qi, Hao Ma, Yifei Zou*, Yuan Yuan*, ZhenZhen Xie, Peng Li, and Xiuzhen Cheng

**Abstract:** To efficiently train the billions of parameters in a giant model, sharing the parameter-fragments within the Federated Learning (FL) framework has become a popular pattern, where each client only trains and shares a fraction of parameters, extending the training of giant models to the broader resources-constrained scenarios. Compared with the previous works where the models are fully exchanged, the fragment-sharing pattern poses some new challenges for the backdoor attacks. In this paper, we investigate the backdoor attack on giant models when they are trained in an FL system. With the help of fine-tuning technique, a backdoor attack method is presented, by which the malicious clients can hide the backdoor in a designated fragment that is going to be shared with the benign clients. Apart from the individual backdoor attack method mentioned above, we additionally show a cooperative backdoor attack method, in which the fragment of a malicious client to be shared only contains a part of the backdoor while the backdoor is injected when the benign client receives all the fragments from the malicious clients. Obviously, the later one is more stealthy and harder to be detected. Extensive experiments have been conducted on the datasets of CIFAR-10 and CIFAR-100 with the ResNet-34 as the testing model. The numerical results show that our backdoor attack methods can achieve an attack success rate close to 100% in about 20 rounds of iterations.

**Key words:** Federated Learning (FL); giant model; backdoor attack; fragment-sharing

## 1 Introduction

Billions of parameters enable the giant models to efficiently address complex tasks with high performance. However, it also prevents the giant models from being applied in some computation/communication-constrained scenarios, e.g., the edge networks[1, 2]. To overcome this problem, Federated Learning (FL) with fragment-sharing has been considered as an efficient paradigm for multiple clients to train a giant model together. Specifically, when a group of clients cooperatively train a giant model, each of them only needs to train a fragment of parameters and share such a fragment with other clients. By doing this, each client significantly has its computation/communication overhead alleviated without losing too much performance on the whole giant model, as has been proved in Refs. [3, 4]. Some typical works include Refs. [5, 6]. Specifically, the clients in Ref. [5] only train some tunable parts of a giant model and share those parameters to realize the federated fine-tuning on a large language model. In

• Senmao Qi, Hao Ma, Yifei Zou, ZhenZhen Xie, and Xiuzhen Cheng are with School of Computer Science and Technology, Shandong University, Qingdao 266237, China. E-mail: senmaoqi@mail.sdu.edu.cn; haoma@mail.sdu.edu.cn; yfzou@sdu.edu.cn; xiezz21@sdu.edu.cn; xzcheng@sdu.edu.cn.
• Yuan Yuan is with Shandong University-Nanyang Technological University International Joint Research Institute on Artificial Intelligence, Shandong University, Jinan 250101, China. E-mail: yyuan@sdu.edu.cn.
• Peng Li is with School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 9658580, Japan. E-mail: pengli@u-aizu.ac.jp.
* To whom correspondence should be addressed.
Manuscript received: 2023-12-31; revised: 2024-04-16; accepted: 2024-05-20

Ref. [6], a flexible FL paradigm for giant models is proposed, which allows each client to share arbitrary fragment of the giant model with theoretical analysis provided. Overall, the FL with fragment sharing has become a popular approach to train a giant model for multiple clients in a distributed system.

Compared with the previous FL works in which the machine learning models are fully exchanged, FL with fragment sharing reduces the computation and communication thresholds for the clients to train a giant model cooperatively. Whereas, it also raises some new problems on the backdoor attacks and defenses on the model sharing. Specifically, in a general FL process with full model exchanged, a malicious client can share an elaborate model or gradient to other clients, so that a backdoor is injected in the global model[7–9]. We say a model is backdoor if it behaves normally on normal inputs while exhibiting the behavior desired by the malicious clients when facing the inputs with specified triggers. For example, a backdoor model can classify a cat wearing glasses as a dog with the glasses as the trigger, while it still recognizes a cat without glasses as a cat. Attackers can also use model aggregation to continuously infect other benign participants by poisoning their own local dataset[10–13]. However, all of these attack methods inject backdoor into the entire model[14, 15], which is not feasible if only fragments are shared. How to hide the backdoor behind a fragment of a model or gradient deserves further investigation. Meanwhile, from the perspective of backdoor defense, it also becomes harder for the benign clients to detect the backdoor from a fragment, when the full model is not delivered. The existing backdoor detection methods in FL[16, 17] based on full-model may not work. For example, Trimmed-mean[17] requires appropriate trimming of each dimension of the model parameters to obtain a robust global model. However, under FL with fragement sharing, there may not be too much overlap between individual fragments, thus rendering this method unusable.

In this paper, we investigate the backdoor attacks on giant model under the FL framework with fragment sharing. Specifically, we consider a decentralized FL system that contains a number of benign clients and malicious clients whose goal is to inject backdoor into the models of benign clients. During the training process, the clients share the specified fragments of the full model at the phase of model sharing. To launch the backdoor attack only with the fragment sharing, we propose a model parameter Fine-Tuning-based Backdoor Attack (FTBA) method. With the above FTBA method, the malicious client can embed a backdoor into its specified fragment by fine-tuning a specified portion of the model parameters on its local poisoned dataset, and continuously infects other benign clients during the model sharing phase. Meanwhile, to alleviate the problem of degraded attack effectiveness caused by embedding a backdoor into a single specific fragment, we also propose a cooperative FTBA (namely FTBA$^*$) algorithm. The FTBA$^*$ method enables multiple malicious clients to cooperatively embed a backdoor in multiple specified fragments. The backdoor characteristics are manifested when and only when these fragments are combined together on the benign client, while individual fragments do not possess significant backdoor characteristics. As a result, FTBA$^*$ greatly increases the flexibility and stealthiness of backdoor attacks, and is harder to detect. To summarize, our main contributions are as follows:

- To the best of our knowledge, this paper is one of the first that studies the potential vulnerability of backdoor attacks on giant models in federated learning with fragment sharing. We hope that our work can shed some light on the security of FL with fragment sharing and help the design of a secure distributed training paradigm for giant models.
- We propose FTBA method, which enables a malicious client to embed a backdoor in a specific fragment of the whole model. This backdoor fragment can successfully perform backdoor attacks on other benign clients. Meanwhile, we also propose FTBA$^*$ method, which can help multiple attackers to carry out cooperative attacks and transfer backdoor into multiple model fragments, thus achieving better backdoor attack results as well as significantly increasing the stealth and detection difficulty of backdoor attacks.
- We conduct extensive experiments on the CIFAR-10 and CIFAR-100 datasets[18] using the ResNet-34 model[19]. The numerical results show that our methods can successfully embed the backdoor in specific model fragments by fine-tuning the model parameters for some common backdoor attacks, such as BadNet[10], Blend[13], Trojan[20], and Adaptve_patch[12]. In most of the experiments, our methods can achieve close to 100% attack success rate. We also demonstrate that model parameter fine-tuning is an effective means of effectively concentrating a backdoor in a specific

model fragment by designing a series of comparison and ablation experiments and illustrating neural network visualization results. These experiments effectively demonstrate the correctness and superiority of our FTBA and FTBA* algorithms.

**RoadMap.** We organize the remainder of this paper as follows: Section 2 introduces the related work and Section 3 gives some FL system descriptions and the problem statement. The backdoor attack algorithm is shown in Section 4. Finally, we conduct detailed experiments and discuss the impact of different parameter settings on the experimental results in Section 5. Lastly, we conclude this work in Section 6.

## 2 Related Work

### 2.1 Backdoor attacks in FL

Since Bagdasaryan et al.[7] first revealed the backdoor vulnerability in FL, a series of backdoor attacks in FL have been proposed in the past decade. Overall, backdoor attacks in FL can be divided into two categories, one is called data poisoning, and the other is called model poisoning[21].

In data poisoning backdoor attack, the attacker will poison the clean dataset by adding a specific trigger to partial samples and modifying their true label, so that the normally training model stealthily contains a backdoor[10, 22, 23]. There are various trigger options for data poisoning, such as a visible pixel block[7, 10, 13, 20] and the invisible Gaussian noise[24]. The broad trigger also encompasses image transformations and in/out distribution samples. For example, Nguyen and Tran[11] used geometric transformations to deform images to poison the clean samples. Besides, some data poisoning methods generate edge distribution or out-of-distribution data as triggers to mislead the model into misclassification[25–27].

In model poisoning backdoor attacks, attackers stealthily inject a backdoor to the local/global model in a certain step of FL. For example, Bagdasaryan et al.[7] scaled the model during the model aggregation phase to replace the global model with a backdoored model. This model replacement method is widely adopted in some subsequent work[8, 9]. In order to increase the concealment of the replacement attacks, Bhagoji et al.[28] limited excessively giant model updates by modifying the loss function in the local training process to evade defense measures based on anomaly detection.

### 2.2 Fragment-sharing FL

Due to the excessive amount of neural network parameters[29], user privacy considerations[30], or heterogeneity considerations[31], clients in FL may cannot share all model parameters with other participants, where each client can only share a fragment of whole model. The key issue in fragment-sharing FL is to find a good fragment to represent the whole model or gradient. Therefore, some work make a trade-off between training efficiency and communication burden or privacy budget by considering these constraints in the local optimization objective to obtain a sparse fragment estimate of local model or gradient[32–34]. Besides, some research has pointed out that sharing partial parameters of the model (such as convolutional layers, batch normalization layers, etc.) can help clients train better personalized models[35, 36]. However, most of the above methods are difficult to adapt to dynamic resource FL scenarios. Therefore, Wang et al.[6] used a mask to get a sparse model for communication and proposed a resource-adaptive learning algorithm under arbitrary neuron assignments with theoretical convergence guaranteed. In this paper, our work is carried out on Ref. [6].

### 2.3 Parameter effective fine tuning

Parameter effective fine tuning is considered an effective, lightweight means of migrating a pre-trained giant model to downstream tasks. Common parameter-effective fine-tuning methods include model parameter fine-tuning[37], adapter fine-tuning[38], and Low-Rank (LoRa) adapters[39]. Model parameter fine-tuning refers to freezing the parameters in the model and only updating part of the parameters during downstream task training. It has been proven that fine-tuning a very small amount of data can achieve transfer learning from the original task to the downstream task[40]. Adapter fine-tuning refers to adding some trainable parameters (called adapters) between certain layers of the neural network and only updating the adapter during downstream task training. This method is considered a more flexible fine-tuning method that does not disrupt the original network structure and has been adopted by a large number of subsequent studies[41, 42]. To further reduce the training parameters of the adapter, Hu et al.[39] proposed an LoRa adapter and used residual connections to fine-tune the network parameters. Compared with adapter fine-tuning, LoRa

can achieve the same model fine-tuning performance with fewer training parameters. This method has been widely used in the fine-tuning of large language models[43].

## 3 FL System and Problem Definition

### 3.1 Decentralized federated learning

We consider a decentralized FL system that consists of $N$ clients, denoted by the set $V$. Each client $k$ has its own local dataset $D_k$, local model $\theta_k$, and exchanges data over the network. However, due to user or system constraints, each client $k$ can only send up to $\mathcal{B}_k$ bits of data during each communication process, which may be much less than $|\theta_k|$, especially in distributed training of large language models[44]. By training their own models locally and sharing the updates with others, all clients will achieve the following goal of our decentralized FL step by step:

$$\min_{\{\theta_1, \theta_2, ..., \theta_N\} \in \mathbf{R}^d} \sum_{k=1}^{N} \frac{|D_k|}{|D|} f_k(\theta_k; D_k) \tag{1}$$

where $f_k$ represents the local loss function of each client $k$ based on local dataset $D_k$, such as cross-entropy or mean square error loss. $D = \{D_1 \cup D_2 \cup \cdots \cup D_N\}$ is the entire dataset and $\theta_k$ is a $d$-dimensional local model of client $k$.

To optimize the objective of Formula (1) efficiently under model fragment sharing, the clients take the following synchronized training proposed in Ref. [6]. Initially, each client has its own local model $\theta_k^0$. In each discrete training round $i = 1, 2, \ldots$, the client $k$ first trains the local model on the local dataset, then sends a fragment of the model that meets the network bandwidth limit to other clients, and finally updates the local model after receiving other model fragments. The details of the synchronized training are given in the following:

- **Local training:** Each client $k$ trains its own local model $\theta_k^i$ to optimize the loss $f_k(\theta_k^i; D_k)$ on its dataset $D_k$, i.e., $\hat{\theta}_k^i = \theta_k^i - \eta_k^i \nabla_{\theta_k^i} f_k(\theta_k^i; D_k)$, where $\theta_k^i$ and $\hat{\theta}_k^i$ are the local models of client $k$ before and after the local training in round $i$, $\eta_k^i$ is the learning rate in each iteration, and $\nabla_{\theta_k^i} f_k(\theta_k^i; D_k)$ is the corresponding gradient of $f_k(\theta_k^i; D_k)$.
- **Fragment sharing:** Each client $k$ gets a sparse model $\bar{\theta}_k^i$ through mask $m_k$, i.e, $\bar{\theta}_k^i = \hat{\theta}_k^i \odot m_k$, where mask $m_k$ is a binary matrix of the same size as $\theta_k^i$ with $\|m_k\|_0 \leqslant \mathcal{B}_k$, and "$\odot$" denotes the Hadamard product.
- **Fragment aggregation:** Each client $k$ aggregates the received sparse models and its local model to generate a new one that will be used in the next round local training. In a mathematical formulation, we have $\theta_k^{i+1} = \text{Aggre}(\cup_{a=1}^{N} \{\bar{\theta}_a^i\})$. The Aggre( ) is an abstract aggregation function here. Specifically, in Ref. [6], each client performs parameter averaging for each dimension of the model. For convenience, in the remainder of this paper, we use Aggre( ) to refer to the model aggregation method used in Ref. [6]. The detailed description is given in Section 4.

By repeating the training rounds for sufficient times, all clients obtain the high-accuracy model on its own dataset.

### 3.2 Problem definition for general backdoor attack in Decentralized Federated Learning (DFL)

In general, when considering federated learning scenarios with backdoor attacks, clients always are divided into two categories: the set of malicious nodes $S_m$ and the set of benign clients $S_b$. For each attacker $k \in S_m$, it sends a tampered model $\tilde{\theta}_k$ to the benign clients. Besides, considering the constrained network bandwidth, $\tilde{\theta}_k$ sent by attacker $k \in S_m$ must also be a sparse model, and the non-zero position should be consistent with that in the binary matrix $m_k$. After the benign clients take an aggregation according to the legitimate models from other benign clients and the tampered models from the attackers, the aggregated model may contain a backdoor, i.e. the aggregated model behaves normally without trigger but acts in a malicious manner when facing triggers. Formally, let $x$ and $\varphi(x)$ represent the clean and manipulated data sample, respectively; $y$ and $\tau(y)$ represent the corresponding true label and target label that the malicious node hopes to induce, respectively. The optimization goal of a general backdoor attack in decentralized FL is

$$\begin{aligned}
&\min_{\substack{\cup \\ k \in S_m}\{\tilde{\theta}_h\}} \sum_{\substack{h \in S_b, \\ \{x, y\} \in D_h}} f_h(\theta_h; \{x, y\}) + f_h(\theta_h; \{\varphi(x), \tau(y)\}),\\
&\text{s.t., } \theta_h^{i+1} = \text{Aggre}(\{\cup_{h \in S_b} \{\bar{\theta}_h^i\}\} \cup \{\cup_{k \in S_m} \{\tilde{\theta}_k^i\}\},\\
&\quad \|\bar{\theta}_k^i\|_0 \leqslant \mathcal{B}_k,\ \bar{\theta}_k^i = \bar{\theta}_k^i \odot m_k,\ \forall\, k \in S_m,\\
&\quad \|\tilde{\theta}_h^i\|_0 \leqslant \mathcal{B}_i,\ \tilde{\theta}_h^i = \tilde{\theta}_h^i \odot m_h,\ \forall\, h \in S_m,\\
&\quad \cup_{i \in V} m_i = \mathcal{J}
\end{aligned} \tag{2}$$

The optimization objective of the attackers includes two parts. The first loss function encourages the victim to achieve high prediction accuracy on the clean inputs. The second loss function encourages a high attack success rate when facing inputs with triggers, that is, it outputs as the attackers desire. In addition, the four constraints in Formula (2) include the constraints on model aggregation in DFL and the constraints on communication. Specifically, the first constraint limits the aggregation method used by clients in DFL when they have received other models. The second and third constraints reflect the bandwidth constraints of the communication network and the sparsification constraints, respectively. Specifically, the transmitted model needs to satisfy that the number of non-zero elements does not exceed the network bandwidth limit, and it needs to ensure that the elements in the mask corresponding to the position of 1 need to be retained. The last constraint requires that the union of the masks of all clients is a matrix of ones $\mathcal{J}$, which means that the parameters of each dimension of the global model need to be trained by the client to ensure the convergence of the global model. In this paper, we assume that the communication bandwidth of each client is fixed and its mask does not change.

In conclusion, the main challenge of backdoor attacks under fragment shared DFL is that whether a benign client or a malicious client, it can only send the specified sparse model when sharing models, i.e., the last two constraints in goal of Formula (2), which is also the main difference between backdoor attacks in traditional FL.

**Our approach.** As mentioned above, the general backdoor attack in DFL needs to ensure that the model trained by the benign client accurately predicts on clean inputs, and outputs the desired results when facing inputs with triggers. To achieve this, we first generate a poisoned dataset using backdoor attack methods based on data poisoning[21], and train the model on this poisoned dataset. In addition, due to communication bandwidth limitations, attackers can only share model parameters specified by the masks. Therefore, during training, we use model fine-tuning to only fine-tune the shared model parameters. More details of the algorithm implementation are given in Section 4. Finally, we summarize all important symbols used in the paper in Table 1.

**Table 1 Important symbols.**

| Parameter | Definition |
|---|---|
| $N$ | Number of clients |
| $V$ | Set of clients |
| $D_k$ | Dataset of client $k$ |
| $\theta_k$ | Model parameters of client $k$ |
| $\mathcal{B}_k$ | Bandwidth limit of client $k$ |
| $f_k$ | Local loss function of client $k$ |
| $\eta_k$ | Learning rate of client $k$ |
| $\bar{\theta}_k$ | Sparse model of benign client $k$ |
| $\tilde{\theta}_k$ | Sparse model of malicious client $k$ |
| $m_k$ | Mask matrix of client $k$ |
| $B$ | Mini-batch of dataset |
| $S_b$ | Set of benign clients |
| $S_m$ | Set of malicious clients |
| $E$ | Epoch number |
| $d$ | Dimension of model parameters |
| $x$ and $\varphi(x)$ | Clean and manipulated samples |
| $y$ and $\tau(y)$ | True label and target label |
| $\nabla f(\cdot)$ | Gradient of function $f(\cdot)$ |
| $\odot$ | Hadamard product |
| $\|\|_0$ | Zero norm |
| Aggre ( ) | Aggregation function |

## 4 Methodology

In this section, we will illustrate model FTBA in fragment shared DFL in detail. We provide the specific implementation process of FTBA in Algorithm 1.

In the TFBA algorithm, the malicious client first needs to generate a poisoned dataset using existing

**Algorithm 1 FTBA for malicious clinet $v$**

**Input:** initialized model $\theta_k^0$, training data $D_k$, and mask $m_k$
**Output:** backdoored model $\theta_h, \forall h \in S_b$

Poison dataset $D_k$ by using data poisoning backdoor attack methods;
**for** $i = 0, 1, ...$ **do**
  Freeze all parameters in $\theta_k$ that have the same position as element 0 in $m_k$;
  **for** $j = 0, 1, ..., E-1$ **do**
    **for** each mini-batch $B \in D_k$ **do**
      $\theta_k^i = \theta_k^i - \eta_k \nabla f_k(\theta_k^i; B)$;
  $\tilde{\theta}_k^i = \theta_k^i \odot m_k$;
  Share $\tilde{\theta}_k^i$ to all other clients;
  Receive $\hat{\theta}_u^i$ from other client $u$ for $u \in V \setminus \{k\}$;
**for** each dimension $e = 0, 1, ..., d-1$ **do**
  count = $\sum_{u \in V \setminus \{k\}} \mathcal{I}(m_k(e)) + 1$;
  $\theta_k^{i+1}(e) = (\theta_k^i(e) + \sum_{\substack{u \in V \setminus \{k\}, \\ m_u(e)==1}} \hat{\theta}_u^i(e))/(\text{count})$;

backdoor attack methods based on data poisoning. Some common data poisoning methods include BadNet[10], Blend[13], Trojan[20], Adaptive_patch[20], etc. These methods select a portion of samples from the clean dataset, add triggers to them, and replace their true labels with the labels that the attacker wishes to induce. It has been proven that models trained normally on this poisoned dataset will have backdoor vulnerabilities[45]. Considering the fragment shared scenario, all participants, including malicious clients, can only send part of the model specified by the mask. Therefore, we use model fine-tuning techniques to embed the backdoor in the designated parameters of the model. Specifically, we freeze all parameters in the local model that are in the same position as the 0 elements in the mask, which represents parameters that will not be shared. During the training process, these frozen parameters participate in the calculation of forward inference, but they do not conduct the parameter update during backward propagation. Formally, the model parameter fine-tuning of malicious node $k$ on dataset $D_k$ can be represented as $\theta_k = \theta_k - \eta_k \nabla_{\theta_k^i} f_k(\theta_k; D_k) \odot m_k$. The malicious node will perform $E$ epochs of local model fine-tuning, and for each epoch, it performs mini-batch stochastic gradient descent on all mini-batches of dataset $D_k$. In the model sharing stage, the malicious client sends out the fine-tuned parameters, i.e., $\tilde{\theta}_k^i = \theta_k^i \odot m_k$. At the same time, the malicious client will receive sparse models sent by other participants. Finally, the malicious client uses the received sparse model for model aggregation. Specifically, for each dimension $e$ of the local model parameters, the malicious client will average the parameters of the $e$-th dimension with the parameters of the $e$-th dimension included in the received sparse model. Therefore, in the algorithm, we first need to use the function $\mathcal{I}()$ to count the number of participants who share $e$-th dimension parameter, where $\mathcal{I}()$ is a function that judges whether the input is 1. Then, it averages the parameters for $e$-th dimension.

## 5 Experiment

In this section, we conduct extensive experiments to verify the effectiveness of the FTBA algorithm. We choose four common data poisoning methods and launch backdoor attacks on other benign clients through model fine-tuning. The experimental results prove that our FTBA algorithm can provide a bridge for most data poisoning-based backdoor attacks in federated learning to be applied in fragment shared scenarios.

### 5.1 Experiment settings for FTBA

Our whole experiment is developed by a Python program with the support of Pytorch[46] for computer vision classification task, which is one of the most commonly used libraries in deep learning. All experiments are conducted on a Linux machine with two NVIDIA GeForce RTX 4090s and 128 GB main memory, implemented in Python 3.9 and using CUDA for parallel computing[47].

**Dataset.** We consider two common visual classification datasets, CIFAR-10 and CIFAR-100[18], which are both labeled subsets of the 80 million tiny images dataset. The CIFAR-10 dataset includes 60 000 color images of size 32 pixel × 32 pixel. It includes 10 classes, each with about 6000 images. The image size and number of images in the CIFAR-100 dataset are consistent with CIFAR-10. The difference is that CIFAR-100 includes 100 classes, each with about 600 images. Therefore, compared to the CIFAR-10 dataset, the CIFAR-100 dataset poses higher demands on the learning ability of neural networks.

**Model.** We use the classic residual neural network ResNet-34 for model training[19]. ResNet-34 is a convolutional neural network model that consists of 34 layers. It has a total of approximately 21.8 million parameters[48]. In the specific implementation, we use ResNet-34 that has been defined by torchvision.models.resnet34 for training, and its network architecture is shown in Fig. 1.

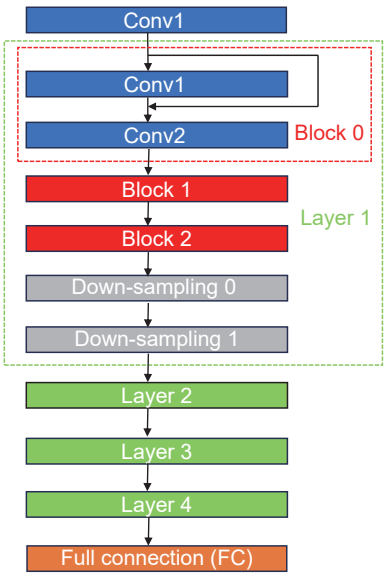


**Fig. 1 Schematic diagram of the architecture of ResNet-34.**

As can be seen from the Fig. 1, a standard ResNet-34 consists of a convolutional layer, four basic layers, and a linear layer. Each basic layer includes three residual blocks and two down-sampling layers. Each residual block contains two convolutional layers and is connected by residual connections. The entire network has a total of $1+(2\times3+2)\times4+1=34$ layers of learnable parameters.

**FL settings.** We consider a decentralized federated learning scenario with 10 clients, including 2 malicious clients. Each client obtains a Non-IID subset of the dataset through a Dirichlet function with $\delta=1$[49]. The Dirichlet function represents a multivariate probability distribution characterized by a vector of positive real values. It serves as a tool to partition the original dataset into distinct subsets, each exhibiting varying degrees of heterogeneity. Sampling from the Dirichlet function yields a collection of proportions, facilitating the segmentation of the CIFAR-10 or CIFAR-100 dataset. Consequently, each subset encompasses a diverse mix of images from different classes, thereby establishing a heterogeneous distribution. All clients undergo 10 rounds of global training, with each round of global training involving 5 epochs of local iterations, and the batch size is 64. Each client optimizes the local objective using the Adam optimizer with an initial learning rate of 0.001. Due to communication constraints, each client can only share part of the ResNet-34 parameters. The shared parameters of all clients are shown in Table 2.

**Table 2 Model partitioning in DFL. The bolded part represents the information of malicious clients. During the training process, benign clients update all parameters, while malicious clients fine-tune the specified parameters.**

| Parameters partitioning | Training parameter | Client ID |
|---|---|---|
| Conv1, Layer 1, Layer 2 | All layers | 5, 6, 8 |
| Layer 2, FC | All layers | 2, 3 |
| Layer 1 | All layers | 4, 7, 9 |
| Layer 3, Layer 4, FC | **Layer 3, Layer 4, FC** | **0, 1** |

**Data poisoning methods.** Since the FTBA algorithm needs to first obtain a poisoned dataset. We implement data poisoning using BadNet[10], Blend[13], Trojan[20], and Adaptive_patch[12], and the poisoning rate is 20%. For label modification, we adopt an "all-to-one" approach, that is, modify all correct labels to the same label. In order to compare their differences more intuitively, we visualize these four poisoning methods in Fig. 2. The triggers of the four data poisoning methods are different. Specifically, BadNet and Trojan add a black and white or color mosaic pixel block to the original airplane image. In Blend, the trigger is a Hello Kitty picture that is the same size as the original picture. The trigger of Adaptive_patch is calculated through an adaptive method, making it more difficult to detect with the naked eye. The above-mentioned attack methods are implemented in the backdoor benchmark platform developed by Li et al.[50]

**Evaluation metrics.** In the research about backdoor attacks, the following two metrics are considered: the Attack Success Rate (ASR) and Clean data Accuracy (CA)[21]. The former refers to the probability that an input with a trigger is successfully predicted as the target class specified by the attacker. The latter refers to the probability that clean input samples without triggers are correctly predicted as their true classes. For a successful backdoor attack strategy, the backdoor model should have a high ASR and CA.

### 5.2 Overall performance of FTBA

In this section, we verify the performance of our FTBA algorithm through multiple sets of experiments. We also discuss the impact of the proportion of parameters being attacked on the attack effect.

**Numerical results.** We first demonstrate the changes in the ASR and CA of all benign local models under the CIFAR-10 and CIFAR-100 datasets with the FTBA algorithm as the number of training rounds increases. The experimental results are shown in Fig. 3.

(a) Original image (b) BadNet (c) Blend (d) Trojan (e) Adaptive_patch

**Fig. 2 Visualized results of different data poisoning methods for an airplane sample in CIFAR-10.**

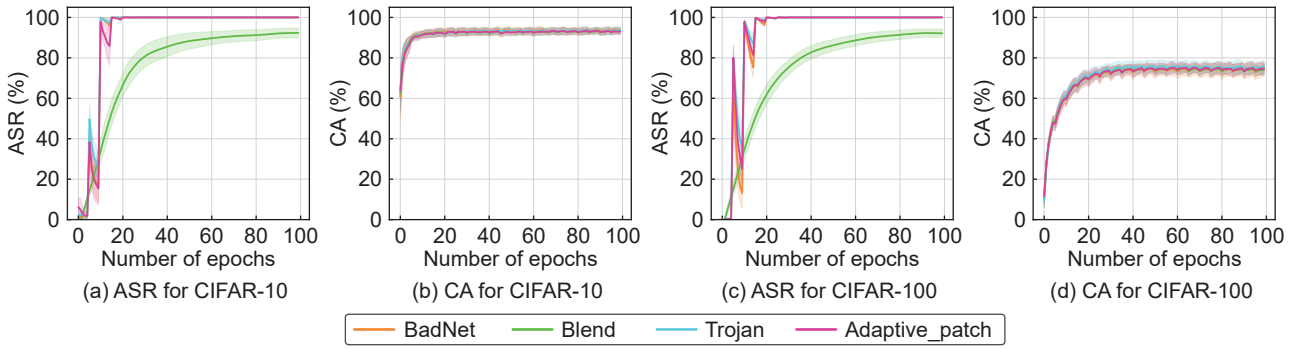


**Fig. 3 ASR and CA of benign clients under CIFAR-10 and CIFAR-100.**

From Fig. 3, it can be seen that whether in CIFAR-10 or CIFAR-100 dataset, our FTBA algorithm can quickly achieve a high ASR for all four attack methods. Specifically, for the BadNet, Trojan, and Adaptive_patch attack methods, they can quickly converge and achieve close to 100% ASR within the first 20 training epoch. For the Blend attack method, although its convergence speed is slower, it also achieves an ASR of over 60% in the 20th round and can reach an ASR of over 90% in the final 100 rounds. At the same time, we notice that the FTBA algorithm can still maintain a high clean test rate while achieving a high ASR. On the CIFAR-10 and CIFAR-100 datasets, all four methods achieve a test accuracy of over 90% and close to 80%, which is close to the test accuracy of the model trained normally.

**Feature visualization results.** To more intuitively demonstrate the attack effect of the FTBA algorithm, we compare the feature distributions of the local models of benign clients and malicious clients. Specifically, we conduct experiments on the CIFAR-10 dataset using four different attack methods. We randomly select a benign and a malicious client and visualize the feature distribution of their local models for benign inputs and inputs with triggers after 100 rounds of training. For the ResNet-34 model, we use the output of Layer 4 in Fig. 2 as the features extracted from the corresponding input images, and use t-distributed stochastic neighbor embedding[51] to reduce the features to a two-dimensional plane for visualization. The visualization results are shown in Fig. 4.

It can be seen that the Trojan and Adaptive_patch attack methods show a more significant difference in feature level for clean inputs and inputs with triggers. At the same time, we can find that the FTBA algorithm can migrate this characteristic to the local model of the benign client. However, for the BadNet and Blend attack methods, clean inputs and inputs with triggers do not show a strong difference at the feature level, and this characteristic is also reflected in the local model of the benign client.

Therefore, from the perspective of feature visualization, our FTBA algorithm can effectively migrate the features of the backdoor model to the local models of other benign clients, which also proves the effectiveness of the FTBA backdoor attack.

**Effectiveness of fine-tuning.** In this section of the experiment, we validate the effectiveness of model parameter fine-tuning. Specifically, we compare the ASR and CA of the local model trained by the malicious client under the conditions of model parameter fine-tuning and full parameter training as the number of training epochs change. The experimental results are shown in Fig. 5, From which, it can be clearly seen that fine-tuning part of the parameters of ResNet-34 (Layer 3, Layer 4, and FC) and full retraining do not show significant differences in convergence speed and final accuracy. This fully demonstrates the effectiveness of model fine-tuning.

**Impact of the proportion of parameters being attacked.** In the FTBA algorithm, a key indicator is the proportion of parameters that the attacker can send, which directly reflects the proportion of parameters being attacked. Therefore, in this experiment, we explore the impact of this indicator on the ASR and CA of the attacked model by changing the proportion of attacked parameters. Specifically, we consider four different attack proportions, namely BadNet$^{+}$ attacking all parameters of ResNet-34, BadNet attacking Layer 3, Layer 4, and FC layer, BadNet$^{-}$ attacking Layer 4, FC layer, and BadNet$^{--}$ only attacking FC layer. The experimental results are shown in Fig. 6.

From the experimental results, it can be seen that the

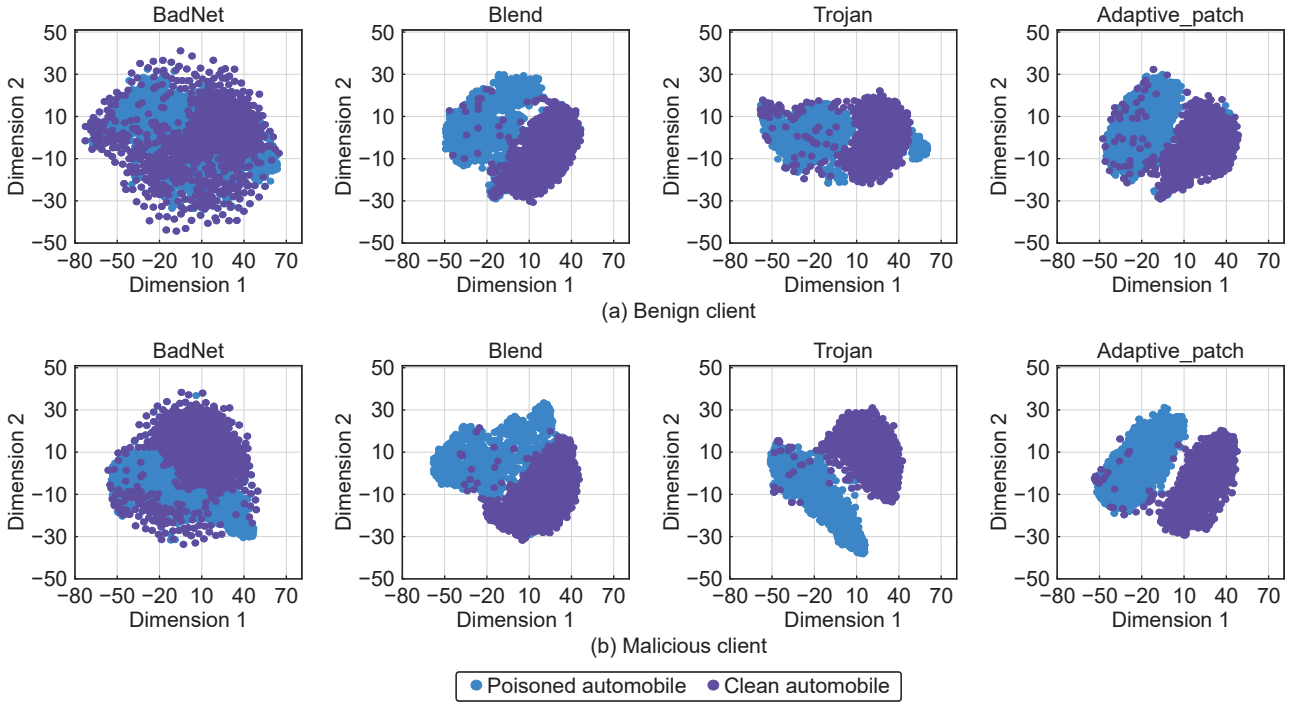


**Fig. 4 Feature visualization results under different backdoor attacks for clients.**

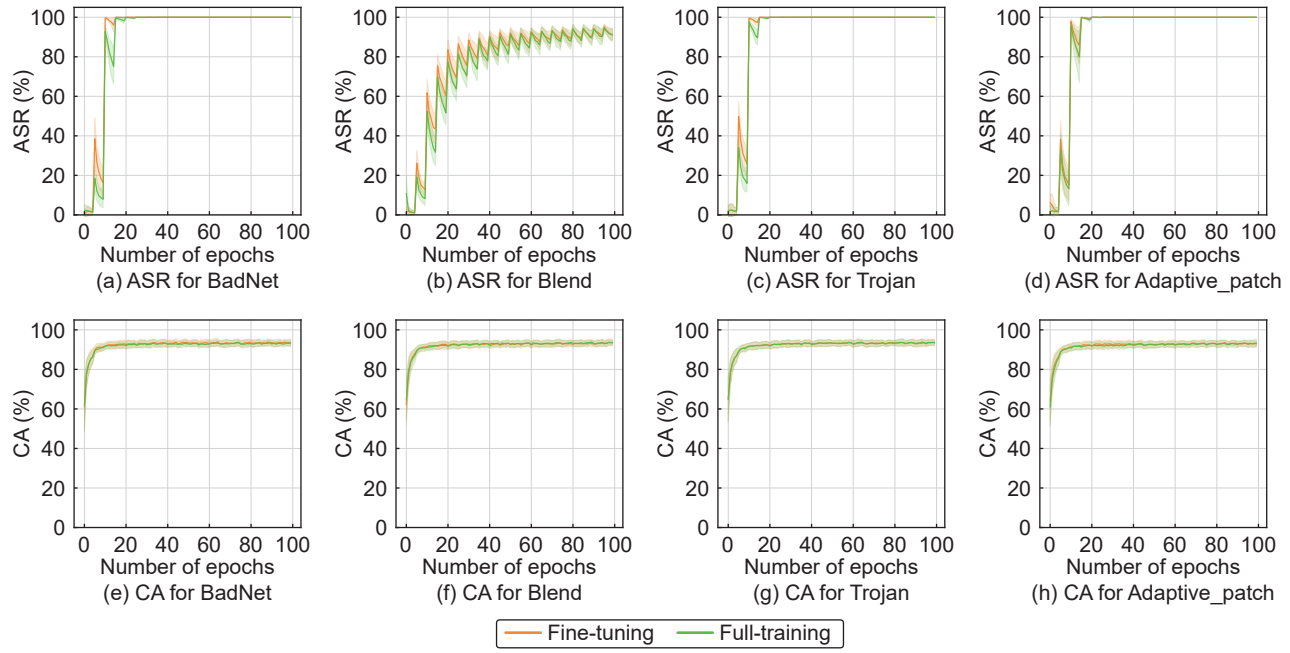


**Fig. 5 Model fine-tuning vs. model full-training for malicious clients under fragment shared DFL.**

proportion of attacked parameters has a significant impact on ASR, but does not affect the CA results. Specifically, by increasing the proportion of attacked parameters, the efficiency of the attack can be improved. For example, Blend will achieve a faster and better attack success rate than Blend$^-$. However, this improvement will gradually saturate. For example, for the four poisoning methods, the full parameter attack does not produce any improvement compared to the attack on Layer 3, Layer 4, and FC layer. At the same time, we notice that when only attacking the FC layer, all four attack methods will fail.

### 5.3 Discussion about cooperative FTBA

In the above experiments, we assume that the layers

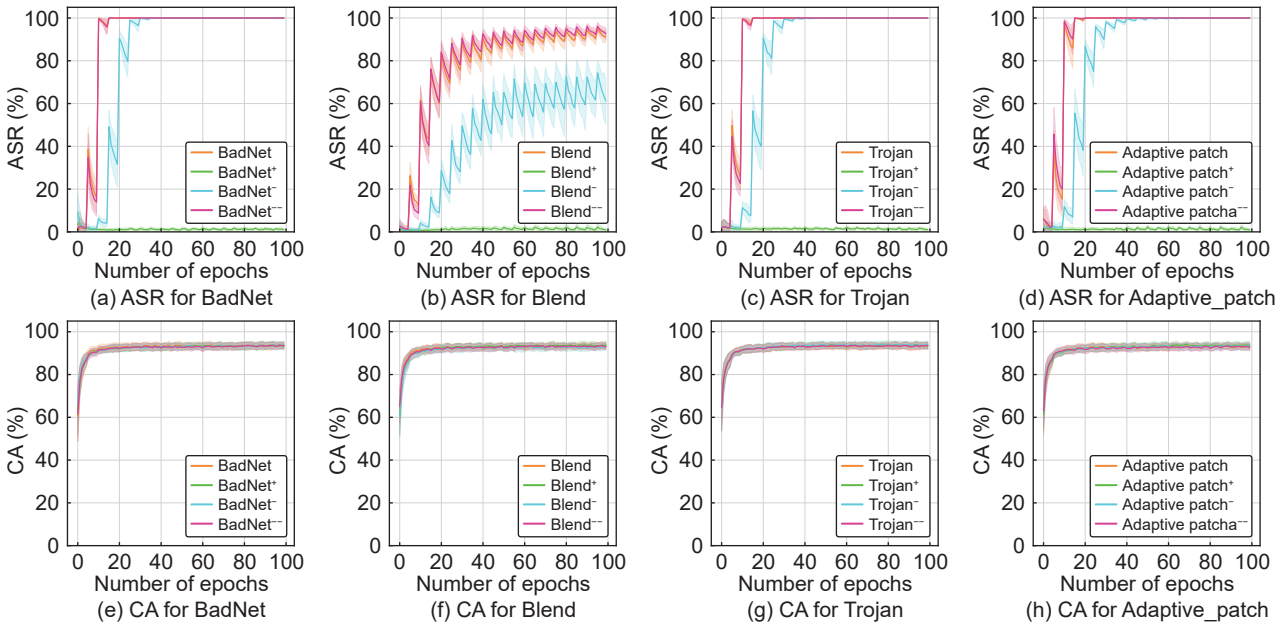


**Fig. 6 ASR and CA for benign clients under different proportion of parameters being attacked.**

attacked by the two attackers are the same, as this can demonstrate the best attack effect. However, in practical problems, it may be difficult for FL systems to ensure that multiple attackers can attack the same layer. Therefore, in this section, we discusse how to better improve the attack effect by hiding the backdoor in multiple model fragments. For this purpose, we propose a cooperative attack algorithm FTBA*, which can effectively alleviate the problem of reduced backdoor attack effectiveness when attackers attack different targets. The pseudocode is shown in Algorithm 2.

Compared to the FTBA algorithm, the FTBA* algorithm needs to know the mask of all attackers. It also introduces a fine-tune mask $m_t$ to guide the fine-tuning of local model, which represents the attack parameter positions of all attackers (as shown in Line 2). Specifically, in FTBA*, the malicious client $k$ fine-tunes all the attack targets of all attackers during local fine-tuning, but only sends specific parameters specified by the mask $m_k$ when sending. The fine-tune mask $m_t$ reflects a consensus reached by all attackers, enabling cooperative attacks under fragment shared conditions among attackers. In a sense, FTBA* allows attackers to conduct fragment-shared FL, which implements distributed backdoor poisoning attacks, so it is more effective.

To demonstrate the effectiveness of FTBA*, we conduct experiments by using the CIFAR-10 dataset

**Algorithm 2 FTBA* for malicious clinet $k$**

**Input:** initialized model $\theta_k^0$, training data $D_k$, mask $m_v, \forall v \in S_m$
**Output:** backdoored model $\theta_h, \forall h \in S_b$

1 Poison dataset $D_k$ by using data poisoning backdoor attack methods;
2 $m_f = \bigcup_{k \in S_m} m_k$;
3 **for** $i = 0, 1, ...$ **do**
4  Freeze all parameters in $\theta_k$ that have the same position as element 0 in $m_f$;
5  **for** $j = 0, 1, ..., E-1$ **do**
6  **for** each mini-batch $B \in D_k$ **do**
7  $\theta_k^i = \theta_k^i - \eta_k \nabla f_k(\theta_k^i; B)$;
8  $\tilde{\theta}_k^i = \theta_k^i \odot m_k$;
9  Share $\tilde{\theta}_k^i$ to all other clients;
10  Receive $\hat{\theta}_u^i$ from other client $u$ for $u \in V \setminus \{k\}$;
11 **for** each dimension $e = 0, 1, ..., d-1$ **do**
12  count = $\sum_{u \in V \setminus \{k\}} \mathcal{I}(m_k(e)) + 1$;
13  $\theta_k^{i+1}(e) = (\theta_k^i(e) + \sum_{\substack{u \in V \setminus \{k\}, \\ m_u(e)==1}} \hat{\theta}_u^i(e))/(\text{count})$;

and ResNet-34. We let Attacker 1 attack Layer 3 of ResNet-34, and Attacker 2 attack Layer 4 and the FC layer of ResNet-34. We compare the changes in ASR of four attack methods under FTBA* and FTBA with the number of training epochs in Fig. 7. At the same time, we perform ablation on Attackers 1 and 2 to reflect the effect of cooperation.

From the experimental results, it can be easily seen that the cooperative attack strategy FTBA* can effectively alleviate the problem of decreased attack

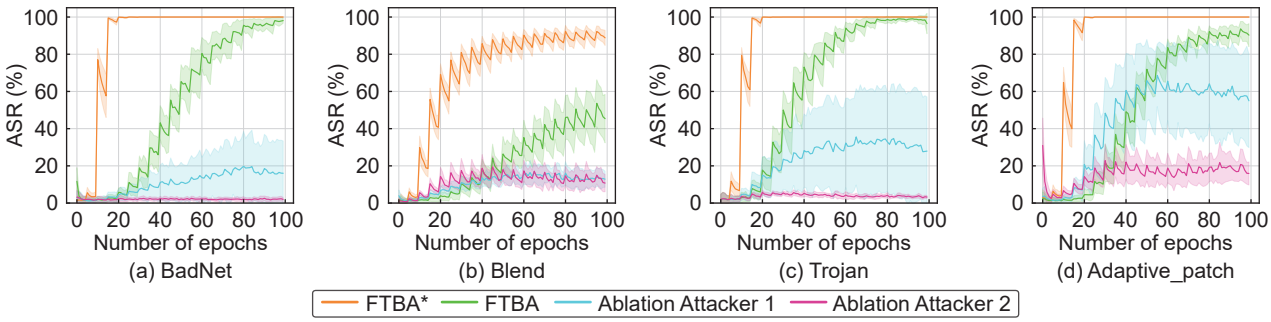


**Fig. 7 ASR for benign clients comparison between FTBA* and FTBA.**

effectiveness when the attack targets are inconsistent. Specifically, when the attack targets are inconsistent, except for the Blend attack, the other three attack methods have reached a nearly 100% ASR in the first 20 rounds. However, FTBA requires 100 or even more rounds of iterations to achieve a similar ASR. For the Blend attack, FTBA* has a significant improvement over FTBA in both attack speed and ASR. At the same time, by comparing the ablation experiments, we find that in the FTBA*, the model fragments sent by a single attacker have a weaker attack effect on the benign client, and in most attack methods, they can only produce an average ASR of less than 20%. This also poses a greater challenge to the defense work of FTBA* if the attackers achieve a specific attack combination consensus. Therefore, we believe that when the attack targets are inconsistent, the FTBA* method can achieve better attack results.

## 6 Conclusion

In this paper, we show that the backdoor attacks still exist when the giant model is trained in federated learning with fragment sharing. To this end, we propose a model fine-tuning-based backdoor attack that effectively embeds a backdoor in a designated model fragment and infects other benign clients during the training process. Meanwhile, we also propose a cooperative backdoor attack strategy, which can hide the backdoor in multiple model fragments, greatly enhancing the effectiveness and detection difficulty of the backdoor attack. We hope that our work can raise concerns about the security of federated learning for giant models and inspire the design of secure distributed training frameworks for giant models. It is worth mentioning that extending the FTBA method to dynamic network environments will be a topic of our future research.

## Acknowledgement

This work was supported by the National Natural Science Foundation of China (Nos. 62102232, 62122042, and 62302247), the Shandong Science Fund for Excellent Young Scholars (No. 2023HWYQ-007), and the Postdoctoral Fellowship Program of CPSF (No. GZC20231460).

## References

[1] Z. Wang, K. Liu, J. Hu, J. Ren, H. Guo, and W. Yuan, AttrLeaks on the edge: Exploiting information leakage from privacy-preserving co-inference, *Chin. J. Electron.*, vol. 32, no. 1, pp. 1–12, 2023.

[2] X. Pang, Z. Wang, D. Liu, J. C. S. Lui, Q. Wang, and J. Ren, Towards personalized privacy-preserving truth discovery over crowdsourced data streams, *IEEE/ACM Trans. Netw.*, vol. 30, no. 1, pp. 327–340, 2022.

[3] A. Hilmkil, S. Callh, M. Barbieri, L. R. Sütfeld, E. L. Zec, and O. Mogren, Scaling federated learning for fine-tuning of large language models, in *Proc. 26th Int. Conf. Applications of Natural Language to Information Systems*, Saarbrücken, Germany, 2021, pp. 15–23.

[4] J. H. Ro, T. Breiner, L. McConnaughey, M. Chen, A. T. Suresh, S. Kumar, and R. Mathews, Scaling language model size in cross-device federated learning, in *Proc. 1st Workshop on Federated Learning for Natural Language Processing* (*FL4NLP 2022*), Dublin, Ireland, 2022, pp. 6–20.

[5] C. Chen, X. Feng, J. Zhou, J. Yin, and X. Zheng, Federated large language model: A position paper, arXiv preprint arXiv: 2307.08925, 2023.

[6] Y. Wang, X. Zhang, M. Li, T. Lan, H. Chen, H. Xiong, X. Cheng, and D. Yu, Theoretical convergence guaranteed resource-adaptive federated learning with mixed heterogeneity, in *Proc. 29th ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, Long Beach, CA, USA, 2023, pp. 2444–2455.

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, How to backdoor federated learning, in *Proc. 23rd Int. Conf. Artificial Intelligence and Statistics*, Palermo, Italy, 2020, pp. 2938–2948.

[8] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J. Y. Sohn, K. Lee, and D. Papailiopoulos,

Attack of the tails: Yes, you really can backdoor federated learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, p. 1348.

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

[10] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, BadNets: Evaluating backdooring attacks on deep neural networks, *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[11] A. Nguyen and A. Tran, Wanet- imperceptible warping-based backdoor attack, arXiv preprint arXiv: 2102.10369, 2021.

[12] Y. Yu, Y. Wang, W. Yang, S. Lu, Y.-P. Tan, and A. C. Kot, Backdoor attacks against deep image compression via adaptive frequency trigger, in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR 2023*), vol. 2023, pp. 12250–12259.

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

[14] X. Zhou, M. Xu, Y. Wu, and N. Zheng, Deep model poisoning attack on federated learning, *Future Internet*, vol. 13, no. 3, p. 73, 2021.

[15] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, Neurotoxin: Durable backdoors in federated learning, in *Proc. 39th Int. Conf. Machine Learning*, Baltimore, MD, USA, 2022, pp. 26429–26446.

[16] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 118–128.

[17] D. Yin, Y. Chen, R. Kannan, and P. L. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in *Proc. 35th Int. Conf. Machine Learning*, Stockholmsmässan, Sweden, 2018, pp. 5650–5659.

[18] A. Krizhevsky, Learning multiple layers of features from tiny images, Master dissertation, University of Toronto, Japan, 2009.

[19] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.

[20] Y. Liu, S. Ma, Y. Aafer, W. C. Lee, J. Zhai, W. Wang, and X. Zhang, Trojaning attack on neural networks, presented at 25th Annual Network and Distributed System Security Symp., San Diego, CA, USA, 2018.

[21] T. D. Nguyen, T. Nguyen, P. Le Nguyen, H. H. Pham, K. D. Doan, and K. S. Wong, Backdoor attacks and defenses in federated learning: Survey, challenges and future research directions, *Eng. Appl. Artif. Intellig.*, vol. 127, p. 107166, 2024.

[22] Y. Liu, T. Zou, Y. Kang, W. Liu, Y. He, Z. Yi, and Q. Yang, Batch label inference and replacement attacks in black-boxed vertical federated learning, arXiv preprint arXiv: 2112.05409, 2021.

[23] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, Invisible backdoor attack with sample-specific triggers, in *Proc. 2021 IEEE/CVF Int. Conf. Computer Vision* (*ICCV*), Montreal, Canada, 2021, pp. 16443–16452.

[24] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems, *IEEE Intern. Things J.*, vol. 8, no. 5, pp. 3310–3322, 2021.

[25] A. Saha, A. Subramanya, and H. Pirsiavash, Hidden trigger backdoor attacks, in *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, no. 7, pp. 11957–11965, 2020.

[26] T. D. Nguyen, P. Rieger, M. Miettinen, and A. R. Sadeghi, Poisoning attacks on federated learning-based IoT intrusion detection system, in *Proc. Workshop on Decentralized IoT Systems and Security*, San Diego, CA, USA, 2020, pp. 1–7.

[27] K. Y. Yoo and N. Kwak, Backdoor attacks in federated learning by rare embeddings and gradient ensembling, in *Proc. 2022 Conf. Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates, 2022, pp. 72–88.

[28] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. B. Calo, Analyzing federated learning through an adversarial lens, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 634–643.

[29] J. Jiang, X. Liu, and C. Fan, Low-parameter federated learning with large language models, arXiv preprint arXiv:2307.13896, 2023.

[30] T. Nguyen and M. T. Thai, Preserving privacy and security in federated learning, *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 833–843, 2024.

[31] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, Federated learning with Non-IID data, arXiv preprint arXiv: 1806.00582, 2018.

[32] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs, in *Proc. Interspeech 2014*, Singapore, 2014, pp. 1058–1062.

[33] S. U Stich, J. B. Cordonnier, and M. Jaggi, Sparsified SGD with memory, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4452–4463.

[34] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, and Y. Zhao, Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff, *IEEE Access*, vol. 6, pp. 16665–16677, 2018.

[35] J. Mills, J. Hu, and G. Min, Multi-task federated learning for personalized deep neural networks in edge computing, *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 630–641, 2022.

[36] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, Towards personalized federated learning, *IEEE Trans. Neural Network. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, 2023.

[37] A. Mathew, P. Amudha, and S. Sivakumari, Deep learning techniques: An overview, in *Proc. Int. Conf. Advanced Machine Learning Technologies and Applications*, Singapore, 2021, pp. 599–608.

[38] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, Parameter-efficient transfer learning for NLP, in *Proc. 36th*

*Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 2790–2799.
[39] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, Lora: Low-rank adaptation of large language models, arXiv preprint arXiv:2106.09685, 2021.
[40] A. Jeddi, M. J. Shafiee, and A. Wong, A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning, arXiv preprint arXiv: 2012.13628, 2020.
[41] R. He, L. Liu, H. Ye, Q. Tan, B. Ding, L. Cheng, J.-W. Low, L. Bing, and L. Si, On the effectiveness of adapter-based tuning for pretrained language model adaptation, arXiv preprint arXiv:2106.03164, 2021.
[42] Y. L. Sung, J. Cho, and M. Bansal, VL-ADAPTER: Parameter-efficient transfer learning for vision-and-language tasks, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition*, New Orleans, LA, USA, 2022, pp. 5217–5227.
[43] X. Wang, L. Aitchison, and M. Rudolph, LoRA ensembles for large language model fine-tuning, arXiv preprint arXiv:2310.00035, 2023.
[44] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy, Challenges and applications of large language models, arXiv preprint arXiv: 2307.10169, 2023.
[45] L. Truong, C. Jones, B. Hutchinson, A. August, B. Praggastis, R. Jasper, N. Nichols, and A. Tuor, Systematic evaluation of backdoor data poisoning attacks on image classifiers, in *Proc. 2020 IEEE/CVF Conf. Computer Vision and pattern Recognition Workshops*, Seattle, WA, USA, 2020, pp. 3422–3431.
[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, in *Proc. 33rd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, p. 721.
[47] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu, Optimization principles and application performance evaluation of a multithreaded GPU using CUDA, in *Proc. 13th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, 2008, pp. 73–82.
[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., PyTorch: An imperative style, high-performance deep learning library, in *Proc. Advances in Neural Information Processing Systems 32* (*NeurIPS 2019*), Vancouver, Canada, 2019, pp. 8024–8035.
[49] T. M. H. Hsu, H. Qi, and M. Brown, Measuring the effects of non-identical data distribution for federated visual classification, arXiv preprint arXiv: 1909.06335, 2019.
[50] Y. Li, M. Ya, Y. Bai, Y. Jiang, and S.-T. Xia, Backdoorbox: A python toolbox for backdoor learning, arXiv preprint arXiv:2302.01762, 2023.
[51] L. van der Maaten and G. Hinton, Visualizing data using t-SNE, *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

**Senmao Qi** received the BEng degree from Shandong University, China in 2021. He is currently a PhD candidate at School of Computer Science and Technology, Shandong University, China. His research interests include distributed machine learning, AI security, and wireless network.

**Hao Ma** is currently an undergraduate student at School of Computer Science and Technology, Shandong University, China. His research interests include federated learning and AI security.

**Zhenzhen Xie** received the MEng and PhD degrees in computer science from Jilin University, China in 2014 and 2021, respectively. She is currently a postdoctoral researcher at School of Computer Science and Technology, Shandong University, China. Her research areas are edge computing, IoTs, and federated learning.

**Yifei Zou** received the BEng degree from Wuhan University, China in 2016, and the PhD degree from The University of Hong Kong, China in 2020. He is currently an assistant professor at School of Computer Science and Technology, Shandong University, China. His research interests include wireless networks, ad hoc networks, and distributed computing.

**Yuan Yuan** received the BS degrees from Shanxi University, China in 2016, and the PhD degree from Shandong University, China in 2021. She is currently a postdoctoral researcher at Shandong University-Nanyang Technological University International Joint Research Institute on Artificial Intelligence, Shandong University, China. Her research interests include distributed computing and distributed machine learning.

**Peng Li** received the PhD degrees in computer science from The University of Aizu, Japan, where he is currently an associate professor. His research interests mainly focus on wired/wireless networking, cloud/edge computing, distributed AI systems, and blockchain. He has authored or co-authored over 100 papers in major conferences and journals. He won the 2020 Best Paper Award of *IEEE Transactions on Computers*. He serves as the chair of SIG on Green Computing and Data Processing in IEEE ComSoc Green Communications and Computing Technical Committee. He is a guest editor of *IEEE Journal of Selected Areas on Communications*, the editor of *IEEE Open Journal of the Computer Society* and *IEICE Transactions on Communications*. He is a senior member of IEEE.

**Xiuzhen Cheng** received the MEng and PhD degrees in computer science from University of Minnesota, Twin Cities, USA in 2000 and 2002, respectively. She was a faculty member at Department of Computer Science, The George Washington University, USA in 2002–2020. Currently she is a professor of computer science at Shandong University, China. Her research focuses on blockchain computing, security and privacy, and IoTs. She is a fellow of IEEE.