TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/16 pp2449–2467
DOI: 10.26599/TST.2024.9010264
Volume 31, Number 5, October 2026

Research Article

ELTFL: A Trusted Heterogeneous Federated Learning in Edge Scenario

Jinshan Lai, Ruijin Wang*, Chuanwen Luo, and Fengli Zhang

Abstract: Federated learning (FL) enables deployment of smart applications, where the intelligent model is trained with distributed data to achieve high accuracy and generalization capabilities. However, resource-constrained terminal devices are unable to train a global model at scale, leading to device dropouts and reduced data utilization. To achieve effective FL among heterogeneous models, we propose the edge learning based trusted heterogeneous federated learning scheme (ELTFL), which allows clients with different heterogeneous models to participate in FL through the application of knowledge distillation (KD). Specifically, ELTFL utilizes integrated distillation to build a heterogeneous multi-model training architecture for FL, which downloads global models to edge servers while preserving heterogeneous models of terminal devices. The edge server utilizes a small set of datasets along with probabilistic outputs from terminal devices to train the global model, and performs global model aggregation in the cloud. In addition, to safeguard against gradient leakage attacks, we introduce a lightweight iterative masking technique between edge servers and terminal devices to ensure the privacy and reliability of intermediate information. Extensive experiments on the CIFAR10 and CIFAR100 datasets demonstrate that ELTFL performs well in terms of accuracy, robustness to data heterogeneity, communication overhead, and privacy protection.

Key words: federated learning; knowledge integration; privacy-preserving; edge computing

1 Introduction

Federated learning (FL) enables multiple end devices to jointly train an accurate global model without exchanging raw data, making it a promising approach for privacy-preserving machine learning[1–5]. Despite its potential, FL faces two significant challenges in real-world applications that need to be addressed to ensure its effective deployment. The first major challenge is the heterogeneity of terminal devices. Terminal devices often have limited computation, memory, and energy resources, making it difficult to train complex global models. For example, smart bracelets can only deploy the visual geometry group 8 (VGG8) model, while smart terminals can deploy the VGG19 model[6]. The model heterogeneity of end devices leads to the inability of FL to perform normal aggregation. The second challenge concerns data privacy. Although FL prevents the direct sharing of raw data, model parameters transmitted between devices and the central server can still expose sensitive information. Attackers could potentially infer private data from these parameters during model updates,

• Jinshan Lai, Ruijin Wang, and Fengli Zhang are with School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail: 202311090910@std.uestc.edu.cn; ruijinwang@uestc.edu.cn; fzhang@uestc.edu.cn.
• Chuanwen Luo is with School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China, and also with Engineering Research Center for Forestry Oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China. E-mail: chuanwenluo@bjfu.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2024-02-20; revised: 2024-06-24; accepted: 2024-12-25

posing a significant threat to user privacy.

In terms of model heterogeneity, existing research has proposed offline knowledge distillation (KD), in which a pre-trained model assists heterogeneous models during co-training[7–11]. However, these approaches are mostly used for personalised FL, where they do not facilitate training a unified global model. In addition, some methods[12, 13] establish a global model on the server that corresponds to each client model. While this allows for aggregation according to individual model structures, it dose not get a unified global model. In terms of data privacy, some scholars have proposed cryptographic methods[14–16], such as multi-party secure computation and homomorphic encryption[17], to protect transmitted data. However, they consume huge computational resources. Other scholars have implemented differential privacy mechanisms, which trade off some degree of accuracy for enhanced data privacy protection. However, this compromise on accuracy is often undesirable in scenarios that require high precision[18].

To address the above challenges, we shift the gradient space to a specific unified region to facilitate the aggregation of heterogeneous models. In addition, we propose a simple masking scheme to accommodate the resource constraints of terminal devices. By leveraging KD, models on terminal devices engaged in the same task can exchange and aggregate data knowledge. This aggregated knowledge contributes to train global model at the edge, which in turn updates the terminal model. A simple mask can secure the transmitted data without adding extra computational and storage burden to the terminal devices, making it suitable for FL in edge scenarios. However, finding a unified aggregation space for heterogeneous models remains a challenge due to the structural differences across models. To overcome this, we utilize the last layer of the model classifier as a common aggregation space, drawing on the benefits of KD[19]. First, we train the terminal models with private data and upload the output of their last layer to the edge server. The edge server uses its own data, along with outputs from multiple terminal devices, to train the global model. The aggregated output subsequently guides the training of terminal models. Finally, the cloud aggregates the model parameters of multiple edge servers. In summary, our contributions are as follows:

(1) We establish a heterogenous multi-model training architecture for FL based on KD, which concatenates decentralized terminal devices to train an accurate global model by integrating knowledge from other heterogeneous models.

(2) We propose an edge-lightweight iterative mask scheme, which only needs to compute the generated mask at the beginning of training. Furthermore, a timestamp is added to model outputs to ensure the privacy and freshness of the transmitted data.

(3) Extensive experiments on image classification tasks demonstrate that the edge learning based trusted heterogeneous federated learning scheme (ELTFL) exhibits robust performance in terms of accuracy, privacy protection, and communication overhead.

The rest of this paper is structured as follows. Section 2 provides an overview of related works on FL, focusing on model heterogeneity and privacy issues. Section 3 presents the description of three tier architecture of FL. Section 4 presents the details in ELTFL. Section 5 provides the theoretical analysis, and in Section 6, we discuss experimental results, followed by conclusion and future work in Section 7.

2 Related Work

In this section, we describe the related work of ELTFL, including FL for heterogeneous models, KD in FL, and privacy-preserving in FL.

2.1 Federated learning for heterogeneous model

Compared with the ideal situation where the terminal models are consistent, real-world terminals have different computational powers, communication capacities, and memories. Consequently, FL with heterogeneous models is more aligned with real-world scenarios, as different terminals prefer to use models that match their respective resources. For example, Li and Wang[12] identified this necessity and proposed heterogenous federated learning via model distillation (FedMD), which utilizes transfer learning to achieve personalized FL under heterogeneous models. Similarly, Lin et al.[13] proposed ensemble distillation for robust model fusion in federated learning (FedDF), which trains global models on the server in alignment with terminal model types and facilitates mutual learning between models through KD. Jiang et al.[20] introduced federated learning algorithm based on knowledge distillation (FedDistill), where an optimal terminal model is selected as the global model, and other terminal models are optimized via KD. Additionally, federated optimization and privacy

preservation under heterogeneous models are under research[10, 21, 22]. Recently, Sattler et al.[23] proposed leveraging unlabeled auxiliary data in federated learning (FedAUX), which focuses on FL of heterogeneous models from the perspective of dataset partitioning. Furthermore, personalized FL has been employed to address the challenges of model heterogeneity. For instance, Wang et al.[24] proposed personalized federated learning via heterogeneous model reassembly (pFedHR), which tackles the problem of heterogeneous personalized models as a matching optimization task. Similarly, Yi et al.[25] introduced a heterogeneous personalized FL framework based on low-rank adaptation (LoRA). However, these approaches consistently fail to construct an effective global model. In contrast, ELTFL extends traditional FL into three layers and introduces global models at the edge server. This approach enables global model aggregation in the cloud, while supporting knowledge transfer at the terminal level.

2.2 Knowledge distillation in federated learning

In federated knowledge distillation (FKD), Zhu et al.[26] proposed data-free knowledge extraction method tailored for heterogeneous FL, which enhances the model's generalization performance. However, its performance is limited by the lack of guidance derived from data knowledge. To address this limitation, Zhang et al.[27] introduced a knowledge extraction method without data to fine-tune the global model in the server, aiming to alleviate direct model aggregation. Additionally, Gong et al.[28] introduced the utilization of public data for one-time offline knowledge extraction, which tackles the challenges of heterogeneous FL and privacy concerns. However, this approach heavily relies on the availability of public data. Alternatively, Han et al.[29] proposed a bilateral knowledge extraction technique with comparative learning as the core component. This method enables FL systems to operate without requiring clients to share data features. In edge computing scenarios, Chen et al.[30] proposed an FL architecture based on knowledge distillation to address distributed optimization challenges under resource-constrained conditions. Similarly, Qiao et al.[31] introduced a federated adversarial training framework leveraging knowledge distillation to enhance model robustness through global prototypes. Although the aforementioned methods bypass data aggregation or offline training, they may not be suitable for real-time edge scenarios. In contrast, ELTFL allows individual terminals to train their own suitable models, which is more suitable for edge scenarios with limited computing resources.

2.3 Privacy-preserving in federated learning

In distributed deep learning, numerous data privacy-preserving methods have been proposed[32, 33]. In FL, for example, Kumari et al.[34] proposed a secure authentication scheme based on elliptic curve cryptography for Internet of Things (IoT) and cloud servers, offering improved efficiency and security against various known attacks. Bonawitz et al.[17] proposed a multiparty secure multi-party computation (SMPC) under FL, utilizing secret sharing[35] to create a double mask for perturbing the model gradient. This scheme mitigates the problem of gradient leakage during transmission. However, it requires significant communication overhead, as the terminal device needs to communicate with the edge server multiple times for each training round. Moreover, the transmission of high-dimensional gradient consumes substantial bandwidth, potentially causing delays, especially in edge scenarios with numerous terminal devices.

To address the above challenges, Wang et al.[36] proposed privacy protection scheme for federated learning under edge computing (PPFLEC), which employs double masks at both edge server and cloud levels, reducing the number of entities involved in the aggregation process. However, this approach still requires multiple rounds of communication and data uploading from the terminal device to the edge server, increasing the risk of data leakage.

Another privacy-preserving approach is the federated differential privacy scheme proposed by Ref. [37]. This scheme adds noise perturbations to the model gradient to prevent attackers from reconstructing terminal data. However, selecting an appropriate noise level poses a challenge. If the noise is too large, the availability of the gradient may be reduced, while too small noise may result in gradient leakage. Wang et al.[38] proposed a model compression method on the federated averaging (FedAvg) architecture, which reduces the amount of transmitted data and communication overhead. This method partially conceals the original gradient, thereby enhancing data privacy. However, model compression can unpredictably impact accuracy, which may be unacceptable in certain scenarios. In

heterogeneous device and data settings, Sui et al.[39] proposed federated learning via ensemble distillation (FedED), which combines KD with FL to improve the model accuracy. However, this approach is based on traditional cloud servers and may not be suitable for edge scenarios. Additionally, it requires the server to disclose the training dataset during model distillation, which poses a risk of privacy leakage.

Furthermore, Li et al. proposed a robust elliptic curve cryptography (ECC) based provable secure authentication protocol for IoT, offering security and efficiency for IoT applications[40]. Li et al. proposed a privacy-preserving data aggregation scheme for mobile edge computing, which significantly reduces communication costs, making it suitable for multi-access edge computing (MEC) based IoT applications[41]. Recently, Yazdinejad et al.[42] proposed a robust privacy-preserving FL model capable of effectively addressing the challenges posed by malicious encryption. Besides, Luo et al.[43] introduced a lightweight randomized response differential privacy method, which protects data privacy while quantifying the similarity of client data. In comparison, ELTFL enables privacy-preserving information transfer in FL without excessive communication overhead.

3 Problem Description

In this section, we provide a brief overview of the cloud-edge model based on FL, discussing its inherent challenges and potential improvement strategies. Additionally, we outline the relevant parameters that will be used throughout the paper.

3.1 Preliminary

The symbols and associated descriptions for parameters involved in our method are shown in Table 1. Table 2 lists the associated algorithms used for mask addition.

KD employs the informative soft label information from a teacher model to guide the training of a student model, facilitating knowledge transfer and enhancing the accuracy of the student model. Specifically, for the same dataset, both the teacher model and the student model utilize predicted values to calculate the distillation loss, denoted as L_{KD}, at a distillation temperature T. This loss is combined with the student model's individual loss, denoted as L_{CE}, to train the student model. The calculation equations are

Table 1 Description of relevant parameter.

No.	Parameter name	Parameter description
1	e_i	The i-th terminal device
2	D_i^{train}	Private training dataset of e_i
3	D_i^{test}	Private test dataset of e_i
4	O_i	Output result in each training of e_i
5	M_i	Local model of e_i
6	ω_i and ω_i^t	Model parameter of e_i and model parameter of e_i in the training round t
7	k	The k-th category of label
8	ω_{global} and $\omega_{\mathrm{global}}^t$	Global model parameter and model parameter of global model in round t

Table 2 Related cryptographic algorithm.

No.	Parameter name	Parameter description
1	KA.param(·)	Algorithm for generating parameter pp
2	KA.gen(·)	Public and private key generation algorithm
3	KA.agree(·)	Symmetric key generation algorithm
4	AE.enc(·)	Key encryption algorithm
5	AE.dec(·)	Key decryption algorithm

$$L_{\mathrm{KD}} = \mathrm{KL}(s_{\mathrm{l}}, s_{\mathrm{p}}) \tag{1}$$

$$L_{\mathrm{CE}} = \mathrm{CE}(h_{\mathrm{p}}, h_{\mathrm{l}}) \tag{2}$$

where KL is the divergence and CE is cross entropy. s_{l} and s_{p} are the soft label of teacher models and prediction of student models at distillation temperature T, and h_{p} and h_{l} are the prediction and label of student models at distillation temperature T.

For Diffie-Hellman (DH) key negotiation, one party generates random numbers s and t as private keys, and then uses the negotiated parameter pp and generator G to generate the public keys G^s and G^t. The equation is as follows:

$$\begin{aligned} G^s &= G^s(\mathrm{mod}(\mathrm{pp})), \\ G^t &= G^t(\mathrm{mod}(\mathrm{pp})) \end{aligned} \tag{3}$$

Then exchange the public key and compute the symmetric key $S_{s,t}$. The equation is as follows:

$$S_{s,t} = (G^t)^s(\mathrm{mod}(\mathrm{pp})) \tag{4}$$

3.2 Three tier architecture of federated learning

The three-layer FL architecture consists of three main components: cloud, edge servers, and terminal devices. This architecture extends the traditional FL by introducing the concept of the edge layer, which acts as an intermediate layer between the cloud and the terminal, as shown in Fig. 1.

As depicted in Fig. 1, the model training process is

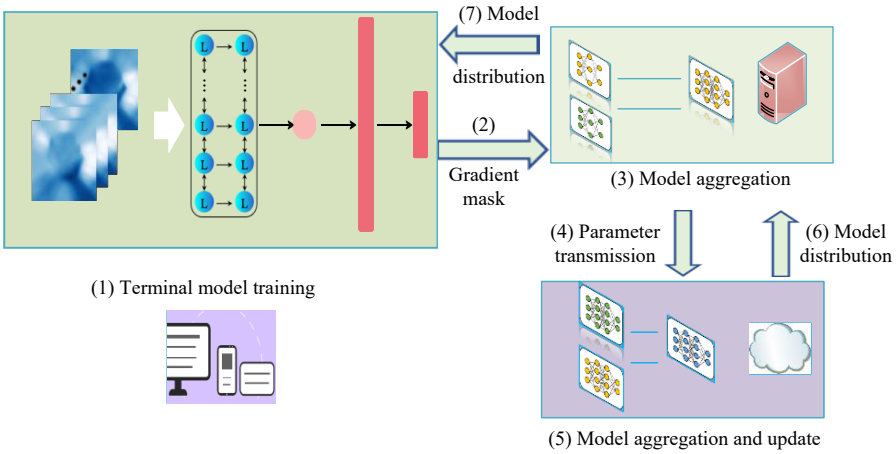

Fig. 1 Three tier architecture of FL. The model training process takes place on the terminal devices, while the model aggregation occurs on the edge servers and the cloud server. L is linear layer.

conducted locally on terminal devices. The edge servers act as the aggregator for models uploaded by terminal devices, while the cloud aggregates the models from different edge servers. Specifically, upon receiving the distributed models from the edge servers, each terminal device employs the private datasets to train the model locally. Through repeated iterative training, the gradient updates are uploaded to edge servers after adding mask. At edge servers, gradient updates are unmasked and aggregated with the aggregation strategy. Subsequently, the parameters are updated using the aggregated gradient, and then uploaded to the cloud.

The cloud collects parameters uploaded from each edge server, and applies an aggregation strategy to combine them. These aggregated parameters are then employed for the current training round, while the model's performance is evaluated. However, the model encounters challenges such as heterogeneous data and significant communication overhead. To address these challenges, our improvement ideas are as follows:

Considering the task M issued by the cloud, different entities participating in FL process can choose different models based on their hardware capabilities while achieving the same function. It is assumed that the server requires a global model to achieve higher accuracy and generalization. However, some terminal devices may have limited resources and can only train small local models, which might have different architectures compared to the global model. In this case, some tiny models can be selected at the terminal and some large models can be used as global models at the cloud and edge servers. To achieve this, we introduce knowledge refinement to enable local and global models to learn from each other. In addition, to safeguard model parameter privacy, we add iteration masks to the model outputs to protect data privacy with the minimal communication overhead.

4 Our Scheme

In ELTFL, consider that terminal devices and edge server clusters have similar datasets. Following the architecture mentioned in Section 3, ELTFL has three entities: terminal devices, edge servers, and cloud. We assume that the edge servers and cloud are honest and curious, while the edge servers include coordinators and trainers who have their own datasets. The cloud includes aggregators and coordinators and the rest of the modules are ignored.

4.1 Overall design

Assume that the terminal device set is denoted as $I = e_1, e_2, \ldots, e_i, \ldots, e_n$, where each terminal device e_i possesses a private dataset D_i which is divided into training dataset D_i^{train} and test dataset D_i^{test}. These datasets are utilized for local model training and testing on the respective terminal devices. To effectively train the model on edge servers, we incorporate integrated distillation learning. Besides, each edge server u has clustered dataset (the edge server and all terminals communicating with it are treated as a cluster) D_u^{p}, which is utilized for performing global model training. The data for the cluster data and the terminal data originate from the same domain, for example, both are subsets of the CIFAR10 dataset. The cloud plays the role of coordinating the edge server cluster and

aggregating models across multiple clusters. Based on the model architecture shown in Fig. 1, we propose an FL architecture for heterogeneous models with high accuracy, low overhead, and privacy preservation. The overall architecture of ELTFL is shown in Fig. 2.

Specifically, the terminal devices receive models from the cloud and edge servers. Due to variations of computing power and memory capacity, some terminal devices are not capable of training global models. Therefore, different models are utilized for training. The edge server allocates the model to terminal devices, and we do not discuss the model allocation method here. As illustrated in Fig. 2, the terminal device e_i utilizes the private dataset D_i to train the local model M_i to obtain O_i and encrypts O_i using an iterative mask, as described in Section 4.4, which is subsequently uploaded to the edge server. The edge server collects the output O_i from each terminal, removes the mask, saves the data, and computes the average value in one dimension to obtain the outmean. The outmean and distillation temperature T are utilized to compute the KL divergence to guide the training of global and local models. The trained global model is uploaded to the cloud, the outmean and distillation temperature T are sent to each terminal device. Using these parameters, each terminal completes the training of its local model. The cloud aggregates the parameters uploaded by the edge cluster, validates the model's performance, and determines whether to conduct the next round of training.

4.2 Terminal local model training

The terminal device e_i obtains the model M_i allocated by the edge server and utilizes the private dataset D_i^{train} to train it locally, where $D_i^{\text{train}} = (x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{id}, y_{id})$, d is the length of the dataset D_i^{train}. The sample x_i of the training dataset D_i^{train} is used as the input of the model M_i, and the output O_i is obtained from forwarding propagation, where O_i represents the probability distribution predicted by the model. The loss function is obtained by cross entropy between O_i with the data label y_i. The hard loss L_i^{hard} is calculated

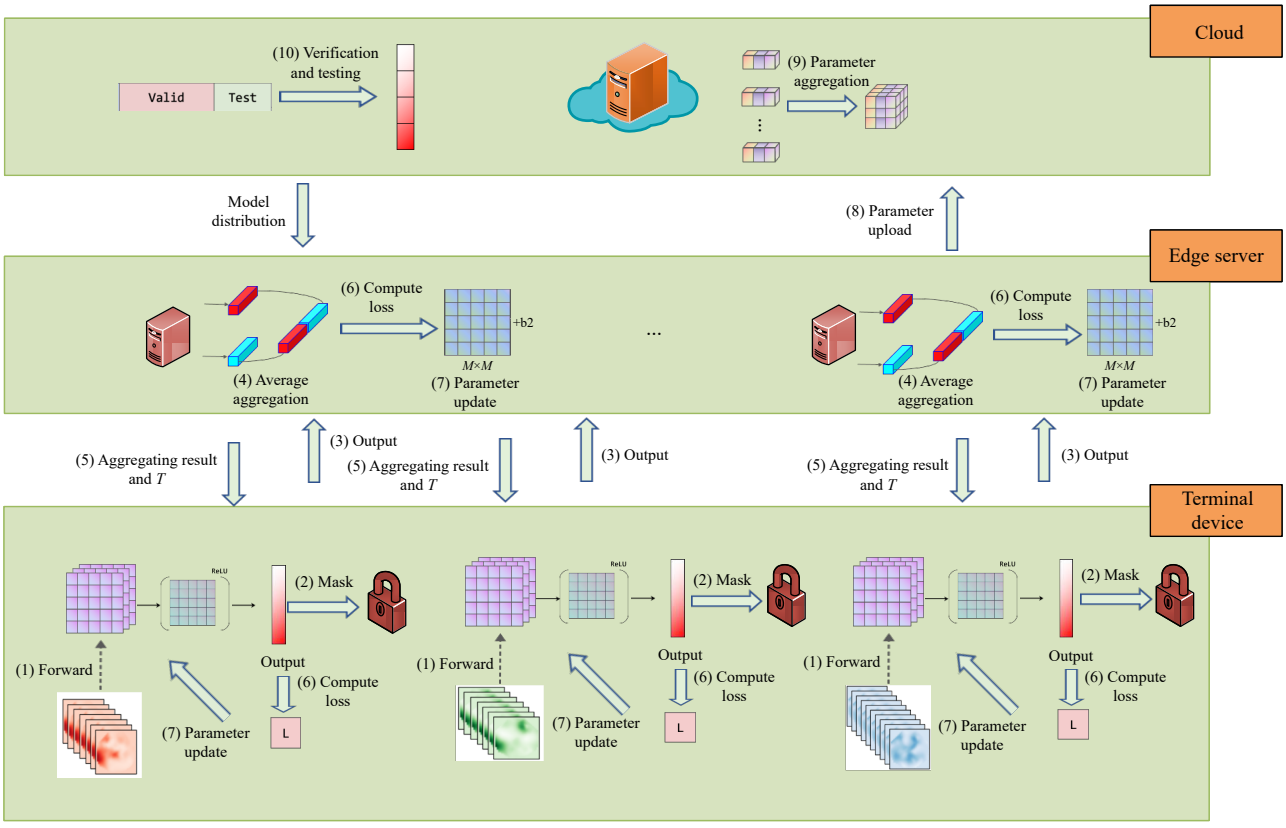

Fig. 2 General architecture of ELTFL. End devices locally train local models of different model architectures and integrate knowledge with the corresponding edge servers by means of probabilistic outputs. Through the integrated knowledge, the end devices and edge servers are trained to obtain stronger local and global models respectively, while the edge servers send the trained local models to the cloud for aggregation to obtain better global models. b2 represents the bias of the model.

by

$$L_i^{\text{hard}} = \sum_{k=1}^{K} c_k \ln(q_{ik}) \tag{5}$$

where c_k represents the probability of the k-th category in the data label y_i, K represents the total number of categories, and q_{ik} represents the probability of the k-th category on e_i. Specifically, the computation of q_{ik} is

$$q_{ik} = \frac{\exp(O_{ik})}{\sum_j \exp(O_{ij})} \tag{6}$$

where O_{ik} represents the k-th component of output O_i, O_{ij} represents the probability of all possible categories on the i-th client, and j represents the index of the j-th category. While computing the loss function, e_i encrypts the output O_i to obtain E_i and sends E_i to the edge server, waiting for the edge server to feed back the distillation temperature T and the outmean. Then, e_i utilizes T and outmean to compute the distillation loss L_i^{soft} of model M_i. First, e_i computes the probability distribution of the local model at the distillation temperature T with

$$q_{ik\_T} = \frac{\exp\frac{O_{ik}}{T}}{\sum_j \exp\frac{O_{ij}}{T}} \tag{7}$$

where q_{ik_T} indicates the probability of the k-th category of e_i at distillation temperature T. Then, e_i computes the probability distribution of aggregated results at the distillation temperature T by

$$p_{k\_T} = \frac{\exp\left(\frac{\text{outmean}_k}{T}\right)}{\sum_j \exp\left(\frac{\text{outmean}_j}{T}\right)} \tag{8}$$

where p_{k_T} is the probability distribution of the k-th category of the aggregation results at the distillation temperature T, and outmean_k is the k-th component of the aggregation results, corresponding to the probability of the k-th category. Finally, the distillation loss L_i^{soft} is

$$L_i^{\text{soft}} = T^2 \sum_{k=1}^{K} p_{k\_T} \ln(q_{ik\_T}) \tag{9}$$

where L_i^{soft} represents the distillation loss of the i-th terminal device, computed by the cross entropy of the aggregation result's probability and the local model's probability at the distillation temperature T. e_i combines the hard loss and distillation loss as the loss function of this round, which is expressed as

$$\text{Loss}_i = (1-\alpha_i)L_i^{\text{hard}} + \alpha_i L_i^{\text{soft}} \tag{10}$$

where α_i is the adjustment factor, which is an adjustment for self and distillation loss. e_i utilizes Loss_i to compute the partial derivative of the model parameters to obtain the gradient G_i.

$$G_i = \nabla \text{Loss}_i = \frac{\partial \text{Loss}_i}{\partial \omega_i} \tag{11}$$

Finally, e_i back propagation updates parameters, which is shown in Fig. 2, Step 7. The update equation is as follows:

$$\omega_i = \omega_i - \eta_i G_i \tag{12}$$

where ω_i is the model parameter on e_i, and η_i is the learning rate. At this point, one round of training is completed and the updated local model is obtained. It is worth mentioning that the test set is used to verify the accuracy of the model.

4.3 Global model distillation training

The edge server u receives the probabilistic output from each terminal device, and for the same task, the probability distribution of the output is of the same shape, so it can be directly weighted and averaged, and the average of the computed results is the outer mean. The equation is as follows:

$$\text{outmean} = \frac{1}{|I_u|} \sum_{i=1}^{|I_u|} O_{i,u} \tag{13}$$

where $|I_u|$ represents the number of terminal devices under edge server u, and O_i, u represents the output of client i communicating with edge server u. Once the average aggregation result is obtained, the edge server u proceeds by broadcasting this result to each terminal device. This broadcast assists the terminal devices in completing subsequent training. Additionally, the edge server utilizes the outmean obtained from the aggregation and clustred dataset D_u^{p} to train the global model. The detailed process is illustrated in Fig. 3.

As decpited in Fig. 3, the edge server u utilizes the clustered dataset D_u^{p} to train a model to obtain an output O_{global}. When the distillation temperature is 1, the edge server u utilizes cross entropy to compute its loss, and the equation is as follows:

$$L_u^{\text{hard}} = \sum_{k=1}^{K} c_k \ln(q_k) \tag{14}$$

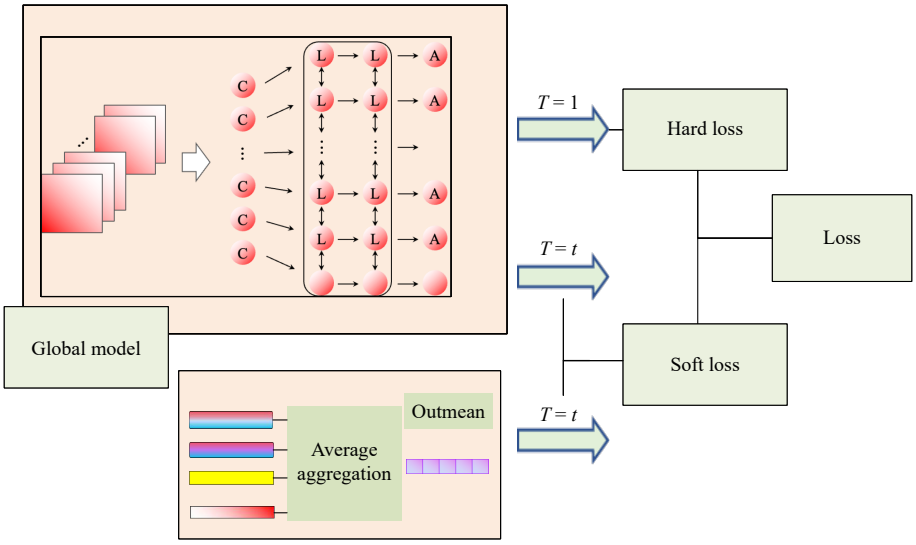

Fig. 3 Global model training architecture. The edge server first aggregates terminal probability distributions to obtain outmean, and combines outmean and cluster data to train the global model. C is channel, L is linear layer, A is activation.

where q_k denotes the probability of the k-th classification, and is computed by

$$q_k = \frac{\exp(O_{\text{global\_}k})}{\sum_j \exp(O_{\text{global\_}j})} \tag{15}$$

where $O_{\text{global\_}k}$ represents the probability of the edge server outputting the k-th category using the cluster data D_u^{p} trained. Then, the edge server u computes the probability distribution at the distillation temperature T

$$q_{k\_T} = \frac{\exp\left(\frac{O_{\text{global\_}k}}{T}\right)}{\sum_j \exp\frac{\left(O_{\text{global\_}j}\right)}{T}} \tag{16}$$

The edge server u utilizes p_{k_T} and q_{k_T} to compute the distillation loss L_u^{soft}, which is formulated as

$$L_u^{\text{soft}} = T^2 \sum_{k=1}^{K} p_{k\_T} \ln(q_{k\_T}) \tag{17}$$

The edge server then combines its hard loss and distillation loss to compute the total loss function as follows:

$$L_{\text{global}} = (1-\alpha)L_u^{\text{hard}} + \alpha L_u^{\text{soft}} \tag{18}$$

where α is the harmonic loss factor of the edge server. Finally, the edge server performs a gradient update on the model parameters according to the loss function L_{global} to obtain the updated global model ω_u and uploads the updated global model parameters to the cloud for the final aggregation.

The cloud receives the global model parameters uploaded by each edge cluster, and since the model parameters uploaded by each edge cluster are of the same architecture, updates are directly aggregated using the average weights. The updated equation is

$$\omega_{\text{global}} = \frac{1}{|U|}\sum_{u=1}^{|U|}\omega_u \tag{19}$$

where $|U|$ represents the number of edge servers. Once the global model parameters are obtained through cloud aggregation, the test set and verification set are used for testing and verification.

4.4 Edge-iterative mask addition model

When the edge server u distributes the task, the number of terminal devices participating in this training is recorded as set I_u. For $\forall i \in I_u$, two pairs of public and private key pairs are generated according to the parameter pp, and the generation formula is

$$(S_i^{\text{sk}}, S_i^{\text{pk}}) \longleftarrow \text{KA.gen(pp)} \tag{20}$$

where S_i^{sk} represents the generated private key, and S_i^{pk} represents the corresponding public key. The edge server u utilizes the parameter pp to generate a public-private key pair.

$$(d_u^{\text{sk}}, d_u^{\text{pk}}) \longleftarrow \text{KA.gen(pp)} \tag{21}$$

where d_u^{sk} represents another generated private key, and d_u^{pk} represents the corresponding public key. For each terminal device e_i in the cluster, the edge server u employs the DH key exchange algorithm to generate an

encrypted symmetric key $s_{u,i}$. The process is carried out as follows:

$$s_{u,i} \longleftarrow \text{KA.agree}(S_i^{\text{sk}}, d_u^{\text{pk}}) \tag{22}$$

According to the symmetry of $s_{u,i}, s_{u,i} = s_{i,u}$, the key will be used for the encrypted transmission of the model output mask.

Before the first round of training, each device i in the terminal device set I randomly generates a random number seed seed_i and uses the pseudo random number generator (PRG) to generate a random vector b_i. Meanwhile, it is extended to a vector b_{prai} with the same dimension as the output O_i^1 as the output mask. The equation is as follows:

$$b_{\text{prai}} = \text{expand}(\text{PRG}(\text{seed}_i), O_i^1) \tag{23}$$

where O_i^1 represents the result obtained by the first training round. The mask b_{prai} is added to O_i^1 to obtain E_i^1, and $s_{u,i}$ is used as the key to encrypt the model parameter. The encryption equation is as follows:

$$e_{i,u} = \text{AE.enc}(s_{u,i}, E_i^1, b_{\text{prai}}, I, \text{timestamp}) \tag{24}$$

where timestamp is used to ensure the integrity of the transmitted information. Once the edge server obtains $e_{i,u}$, it utilizes the symmetric key to decrypt. The decryption equation is as follows:

$$E_i^1, b_{\text{prai}}, I, \text{timestamp} = \text{AE.dec}(s_{i,u}, e_{i,u}) \tag{25}$$

In the t-th round training, the mask results are computed as follows:

$$E_i^t = E_i^{t-1} + O_i^t \tag{26}$$

where O_i^t represents the result of e_i training in the t-th round and E_i^{t-1} represents the mask result in the $(t-1)$-th round. Since the edge server u has each round of mask results, it can quickly remove the mask, obtain the result set, and perform subsequent distillation after aggregation. Once the edge server u completes distillation and training to obtain the global parameter ω_u, it uploads global parameters to the cloud. The cloud, in turn, employs the parameters from each set to execute parameter updates for this specific round. The model distillation training algorithm is presented in Algorithm 1. F represents the forward propagation of the neural network model, O_i^t is the ouput of client i at the t-th round, $O_{i,u}^t$ is the output of client i communicating with edge server u at the t-th round, L_{global}^t is the loss value at the t-th round, and w_{global}^t is the global model weight at the t-th round. The iterative mask algorithm is presented in Algorithm 2.

Algorithm 1 Model ensemble distillation training algorithm

Require: Terminal device set I, terminal local model M_i, dataset D_i, where $D_i = x_{id}, y_{id}$, public dataset D_u^{p}, batch size, training round epoch, and learning rate η.
Ensure: The final global model

for t in epoch **do**
  for device i in I **do**
    for x_{id}, y_{id} in D_i **do**
      $O_{id}^t = F(x_{id}, \omega_i^t)$
      O_i^t.append(O_{id}^t)
    end for
    Add mask to O_i^t and send it to edge server u
    Receive, the t-th round outmaean, outmean^t, and compute loss using Eq. (10)
    Update local model parameter using Eqs. (11) and (12)
  end for
  for edge server u in U **do**
    Receive $O_{i,u}^t$ from the terminal device set I
    Compute outmean^t using Eq. (13) Send outmean^t to the terminal device
    Train the global model with the dataset D_u^{p} yielding the output $O_{\text{global}}^t = F(D_u^{\text{p}}, \omega_u)$
    Compute total loss L_{global}^t using Eq. (18) and update parameter $\omega_u^t = \omega_u^{t-1} - \eta \frac{\partial L_{\text{global}}^t}{\partial \omega_u^t}$
  end for
  for server Aggregation to form global model w_{global}^t using Eq. (19)
end for

5 Theoretical Analysis

In this section, we conduct a theoretical analysis of ELTFL. Our analysis focuses on two key aspects: communication cost and privacy protection. We aim to evaluate whether ELTFL can effectively ensure data privacy protection while achieving intelligent task training with high accuracy and low communication cost.

5.1 Communication cost analysis

Existing approaches commonly employ FL to transmit model parameters trained on terminal devices to the edge server. Nonetheless, they incur substantial traffic and considerable cost due to the transmission of parameters with identical dimensions. To mitigate these challenges, ELTFL integrates FL with KD to reduce communication cost and protect data privacy to

Algorithm 2 Iterative mask algorithm

Require: Finite field F, parameter pp, output O_i of terminal device set I, and number of training rounds epoch.

Ensure: Probability distribution of terminal encryption.

1: Edge server uses the parameter pp to generate a key pair $(d_u^{\mathrm{sk}}, d_u^{\mathrm{pk}})$

2: e_i generates public and private key pairs $(S_i^{\mathrm{sk}}, S_i^{\mathrm{pk}})$ using the parameter pp

3: **for** i in range I **do**

4: $s_{u,i} \longleftarrow \mathrm{KA.agree}(S_i^{\mathrm{sk}}, d_u^{\mathrm{pk}})$

5: **end for**

6: **for** i in range I **do**

7: Initialize mask $b_{\mathrm{prai}} = \mathrm{expand}(\mathrm{PRG}(\mathrm{seed}_i), O_i^1)$

8: $E_i^1 = b_{\mathrm{prai}} + O_i^1$

9: **end for**

10: **for** t in range epoch **do**

11: **for** i in range I **do**

12: $E_i^t = E_i^{t-1} - O_i^t$

13: $e_{i,u} = \mathrm{AE.enc}(s_{u,i}, E_i^t, I, \mathrm{timestamp})$

14: Send it to edge server u to get O_i^t

15: $E_i^t, I, \mathrm{timestamp} = \mathrm{AE.dec}(s_{i,u}, e_{i,u})$

16: $O_i^t = E_i^t - E_i^{t-1}$

17: **end for**

18: **end for**

a certain extent. The detailed communication overhead comparison is shown in Table 3.

In Table 3, m represents the number of samples in each round of training, n represents the vector dimension, I represents the number of terminal devices, and U represents the number of edge servers. Since FedAvg and SMPC are two-tier architectures, we extend them to three tiers where edge servers and cloud servers only aggregate model parameters.

Taking the convolutional neural network (CNN) as an example, the parameters of each layer can be represented as four-dimensional tensors (m, n_{c}, p, and q), with a total of k layers, where n_{c} represents the number of channels, p represents the height of the convolution kernel, and q represents the width of the convolution kernel. The nested mask matrix, required for adding masks to gradients or parameters, contains $m \times n_{\mathrm{c}} \times p \times q \times k$ data. Assuming the tensor dimensions are the same, the communicated data will amount to mn^4. This not only consumes significant bandwidth but also places a high computational burden on generating the nested matrix, particularly when multiplying high-dimensional tensors. From Table 3, we observe that both FedAvg and SMPC involve communicating data of size mn^4. In PPFLEC, datasets are aggregated on the terminal device and transmitted to the edge server. Since the image data are a three-dimensional tensor, the amount of data is mn^3. In ELTFL, only the training results and vector masks of private data need to be transmitted to the edge server, resulting in an amount of data equalling to mn. By reducing the high-dimensional gradient or parameter to one dimension, the communication volume is significantly reduced compared to other approaches.

Table 3 Communication overhead comparison between FedAvg, SMPC, PPFLEC, and ELTFL under CNN. Bold indicates the optimal result.

Scheme	Communication overhead (data volume)	Number of communications per round
FedAvg	$O(mn^4)$	$I+U$
SMPC	$O(mn^4)$	$5I+U$
PPFLEC	$O(mn^3)$	$I+5U$
ELTFL	$\boldsymbol{O(mn)}$	$\boldsymbol{I+U}$

In terms of the number of communication rounds, FedAvg requires two rounds of communication: one between the terminal devices and the edge server, and another between the edge server and the cloud. The number of time is $2(I+U)$. On the other hand, SMPC requires multiple communications due to the addition of dual masks. In each round of training, it needs to communicate five times between edges and one time between edge clouds. Therefore, the total number of time is $5I+U$, which is not applicable to edge scenes due to the high communication overhead. PPFLEC extends the cloud to the edge, but it requires adding a mask five times in the edge cloud and communicating at the edge once for each of these times. In contrast, ELTFL proposes an iterative mask that can realize the secure transmission of model results without multiple communications in a round of training. It only needs to communicate once during a key agreement. Once the training begins, it is the same as FedAvg.

Furthermore, the edge server has the capability to distill knowledge from the collection of terminal devices, enabling accelerated convergence and improved model accuracy. This KD process allows the edge server to leverage the data insights from each terminal device. Meanwhile, the edge server also plays a positive feedback role in the local model training on the terminal device. For edge scenes with

heterogeneous data and devices, the edge server has advantages in training intelligent tasks with the same goal. It applies to FL in the edge scene.

5.2 Privacy analysis

In this section, we analyze the privacy protection capabilities of ELTFL with respect to private terminal data, focusing on the model architecture and the application of iteration masks.

In ELTFL, the transmission between edge servers and terminal devices involves transmitting model training results instead of model gradients. For instance, using e_i as an example, we replace the model gradient G_i with the model result O_i. This approach reduces traffic and effectively mitigates the risk of gradient leakage attacks, particularly when the model architecture is treated as a black box. Furthermore, since training data remain local, attackers are unable to access loss values or compute gradients for conducting gradient disclosure attacks. However, transmitting model results between edges introduces the potential risk of model theft and privacy disclosure, especially when different models are involved. The output results may contain input information, potentially disclosing sensitive training set data. To address this, ELTFL incorporates an iteration mask into the output results, preventing attackers from pinpointing specific samples through the mask. Additionally, timestamps ensure the freshness of the results, protecting against replay attacks by adversaries. The effectiveness of privacy protection is demonstrated in Section 6.5.

6 Experimental Analysis

In this section, we start by outlining the hardware and software configurations used in the experiment. Subsequently, we provide an overview of the experimental results and analysis of ELTFL in three aspects: accuracy, computational and communication overhead, and privacy protection effectiveness. The results demonstrate that ELTFL can achieve superior accuracy and training efficiency while effectively safeguarding privacy.

6.1 Experimental setup

Dataset. We use the CIFAR10 and CIFAR100 datasets, where the CIFAR10 dataset contains 10 categories and the CIFAR100 dataset contains 100 categories. In addition, CIFAR10 and CIFAR100 contain 50 000 training set images and 10 000 test set images, respectively. Within each edge cluster, data are distributed using the Dirichlet distribution and independent and identically distributed (IID) distribution, where parameter β of the Dirichlet distribution is 0.10.

Implementation detail. We use one NVIDIA GeForce RTX4090 as the cloud and two graphics processing units (GPUs, an RTX3090 and an RTX3080) as edge servers, each connected to 12 terminal devices for training local models. We first select 20% of the data evenly from the overall dataset and distribute it equally to two edge servers. Secondly, we divide the remaining 80% of the data into 24 terminal devices via Dirichlet (0.10) and IID distributions, and then distribute the 24 terminal devices equally to the two edge clusters. Each terminal device employs PyTorch for model training, with a batch size of 50. We utilize stochastic gradient descent (SGD) as the optimizer, setting the learning rate to 0.1 with a decay rate of 0.998. The number of local iterations E is set to 20 rounds, with global training extending over 300 rounds. The local sampling rate is fixed at 0.15. To facilitate comparison with consistency-based methods such as FedAvg, we standardize the local models of ELTFL to residual network 8 (ResNet8), ResNet18, VGG16, VGG19, and shuffleNet version 1 (ShuffleV1). These local models are randomly assigned to each terminal device.

Baseline. ELTFL is evaluated alongside other state-of-the-art approaches mentioned below.

- **FedAvg:** FedAvg[38] is a classic FL method that trains a global model by trading client model parameters. It solely supports the training of homogeneous models.
- **SMPC:** SMPC[35] is a secure aggregation algorithm based on FedAvg that utilises multi-party secure computation to achieve secure transmission of model parameters or gradients.
- **PPFLEC:** PPFLEC[36] is a privacy-preserving FL paradigm designed for the cloud-edge-terminal architecture, aimed at preventing information leakage.
- **FedDF:** FedDF[13] accomplishes FL among heterogeneous models by sharing knowledge through the unlabeled data output generated by client models and by training a global classifier.
- **FedDistill:** FedDistill[20] capitalizes on the shared logits of clients and uses the aggregated logits as a regularization term to guide the training of client models. We select one of the client models to act as a

surrogate for the server model.

● **FedMD:** FedMD[12] facilitates knowledge sharing between heterogeneous models through collaboration with an auxiliary dataset. The server aggregates and shares the logits from clients. Analogous to FedDistill, we select one client model to serve as the global model.

For data IID distribution, we compare ELTFL with FedAvg, SMPC, and PPFLEC. For Drichlet distribution, we compare ELTFL with FedDF, FedDistill, and FedMD, the last three algorithms mainly solve the data heterogeneity problem in FL.

Evaluation indicator. We evaluate ELTFL based on several metrics, including accuracy, training time, communication time, and attack success rate (ASR). The definitions of these evaluation metrics are as follows:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\% \tag{27}$$

where N_{correct} is the number of correctly classified samples, and N_{total} is the total number of samples.

The training time for each round is determined by the maximum training time among all clients in that round.

$$T_{\text{train}} = \max_{i \in \{1,2,\ldots,N_{\text{client}}\}} T_{\text{i}} \tag{28}$$

where T_i represents the training time of the i-th client, and N_{client} represents the total number of clients participating in federated learning.

The communication time for each round is determined by the maximum communication time among all clients in that round.

$$T_{\text{comm}} = \max_{i \in \{1,2,\ldots,N_{\text{client}}\}} T_{\text{comm},i} \tag{29}$$

where $T_{\text{comm},i}$ represents the communication time of the i-th client.

The total time required for the training process is calculated as

$$T_{\text{total}} = (T_{\text{train}} + T_{\text{comm}}) \times N_{\text{round}} \tag{30}$$

where N_{round} is the total number of communication rounds. The attack success rate is obtained as

$$\text{ASR} = \frac{N_{\text{successful}}}{N_{\text{acc}_{\text{total}}}} \times 100\% \tag{31}$$

where $N_{\text{successful}}$ is the number of successful gradient leakage attacks, and $N_{\text{acc}_{\text{total}}}$ is the total number of attacks attempted.

6.2 Accuracy under data IID

We start with a comparative accuracy comparison experiment under data IID. We utilize the CIFAR10 and CIFAR100 datasets as experimental data and employ the ResNet20 model as the global model for training on edge servers. We separately use 1, 2, and 3 small models as local models to train on the terminal device, distributing these models equitably across devices. For CIFAR10 and CIFAR100 datasets, ResNet8, ResNet18, ShuffleV1, VGG16, and VGG19 are used as local models to obtain the test accuracy. The test accuracy of local model and global model is presented in Table 4.

From Table 4, it is evident that the test accuracy of the global model varies to some extent with different combinations of local models. This variation is attributed to the differences in the aggregation results of probability distributions from different model combinations. However, all combinations achieve high accuracy, indicating the effectiveness of knowledge integration. The corresponding test loss values and accuracy rates of CIFAR10 can be seen in Fig. 4, and corresponding test loss values and accuracy rates of

Table 4 Test accuracy of ELTFL for global and local models under CIFAR10 and CIFAR100 datasets IID.

CIFAR10, IID				CIFAR100, IID			
Global model	Local model	Test accuracy (%)	Global test accuracy (%)	Global model	Local model	Test accuracy (%)	Global test accuracy (%)
ResNet20	ResNet8	87.17	92.45	ResNet20	ResNet8	59.04	69.24
	ResNet8	87.40			ResNet8	59.03	
ResNet20	ResNet18	94.28	92.16	ResNet20	ResNet18	77.13	67.43
	VGG16	93.15			VGG16	73.56	
ResNet20	ResNet18	94.52	92.11	ResNet20	ResNet18	76.93	68.19
	VGG16	93.35			VGG16	72.13	
	VGG19	91.72			ResNet8	56.98	
ResNet20	ResNet8	87.26	92.80	ResNet20	ResNet8	56.73	66.98
	VGG8	92.87			VGG8	71.87	
	ShuffleV1	93.23			ShuffleV1	69.35	

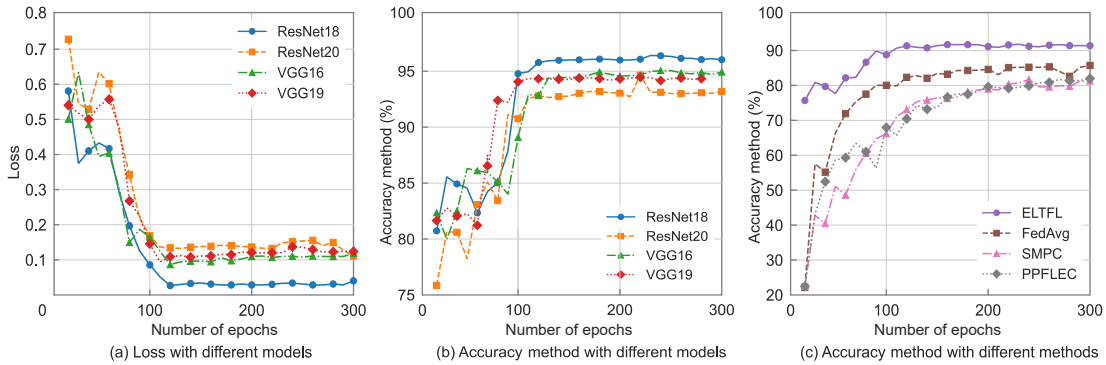

Fig. 4 Loss value and accuracy of CIFAR10 dataset under ELTFL (ResNet18, VGG16, and VGG19).

CIFAR100 can be seen in Fig. 5.

As illustrated in Figs. 4 and 5, ELTFL enables rapid convergence and high accuracy for both local and global models. In the case of CIFAR10 and CIFAR100 datasets, the convergence occurs within 100 training rounds. This is due to the effective utilization of average probability distributions from different clients, which enhances the understanding of global data distributions by both the local and global models. As a result, the test accuracy is improved, and the convergence process is accelerated.

To further verify the performance advantage of global model of ELTFL under IID condition, we compare it with FedAvg, SMPC, and PPFLEC and obtain global model accuracy comparison diagrams, as shown in Figs. 4 and 5. ELTFL utilizes three local models to participate in training, with ResNet20 as the global model.

As demonstrated in Figs. 4 and 5, ELTFL surpasses the other three baselines in convergence speed and accuracy for both datasets. This superiority can be attributed to the knowledge integration of ELTFL at the edge server side, enabling the global model to learn a more comprehensive data distribution globally. Furthermore, we separately compare ELTFL's accuracy on the local model with the other three baseline methods, and the results are presented in Table 5.

As shown in Table 5, the convergence of ELTFL's local models consistently achieves higher accuracy compared to the other three baselines when trained individually. This not only demonstrates the versatility of ELTFL in handling various combinations of heterogeneous models, but also highlights its effectiveness in enhancing the accuracy of local models through knowledge integration. In contrast, SMPC and PPFLEC sacrifice accuracy to ensure privacy protection, resulting in lower accuracy. However, ELTFL successfully strikes a balance between accuracy and privacy by employing lightweight masking and sharing knowledge integration, thereby improving model accuracy to some extent.

6.3 Accuracy under data heterogeneity

To validate the accuracy of ELTFL under non-IID data, we introduce additional baseline methods beyond those used in the IID setting. Table 6 presents the accuracy results of ELTFL and the baseline methods across varying degrees of data heterogeneity.

As shown in Table 6, ELTFL consistently

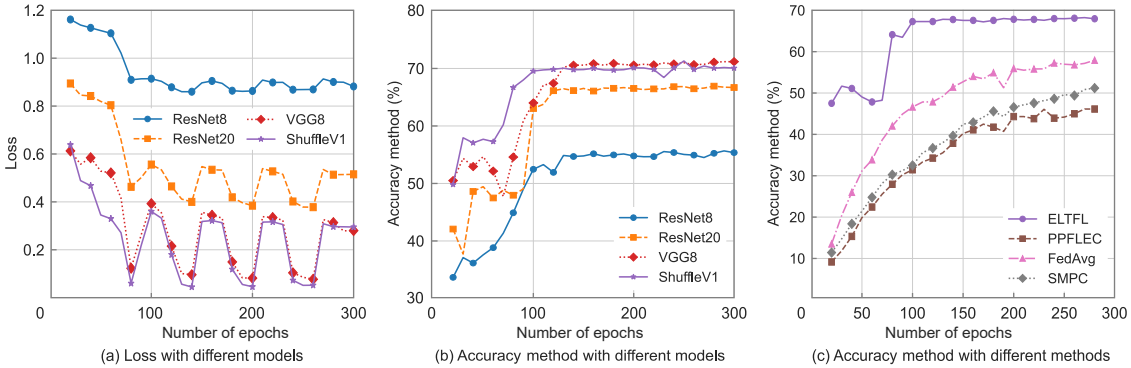

Fig. 5 Loss value and accuracy of CIFAR100 dataset under ELTFL (ResNet8, VGG8, and ShuffleV1).

Table 5 Accuracy between ELTFL and baseline. Bold indicates the optimal value.

(%)

Dataset	Model	ELTFL	FedAvg	SMPC	PPFLEC
CIFAR10, IID	ResNet18	**94.51**	83.64	81.33	81.56
	VGG16	**93.35**	91.62	80.17	80.76
	VGG19	**91.72**	84.50	77.46	78.43
	ResNet20	**92.80**	84.54	78.65	79.32
CIFAR100, IID	ResNet18	**76.93**	61.33	60.54	61.57
	ResNet8	**56.98**	40.60	40.21	41.25
	VGG16	**73.56**	68.73	67.43	68.75
	VGG8	**71.87**	30.01	29.88	32.15
	ShuffleV1	**69.35**	59.16	54.15	62.14
	ResNet20	**69.24**	56.69	55.32	58.24

outperforms other methods in accuracy across datasets with different degrees of data heterogeneity. This highlights the capability of ELTFL to handle non-IID data in edge computing scenarios. The superior accuracy achieved by ELTFL can be attributed to its effective integration and sharing of knowledge through edge servers.

6.4 Efficiency analysis

This section provides a comparative analysis of the time cost between ELTFL and other schemes, focusing on computational and communication overheads. We evaluate training time per round and communication time per round, comparing ELTFL with FedAvg, SMPC, and PPFLEC on the CIFAR10 and CIFAR100 datasets. For this experiment, ELTFL utilizes three local models (ResNet18, VGG16, and VGG19), with ResNet20 as the global model. The results of the comparison are presented in Tables 7 and 8.

Table 7 shows that when training the ResNet20 model using the CIFAR10 dataset, ELTFL achieves convergence in 100 rounds. The number of rounds required for convergence is lower compared to the benchmark FedAvg, as well as significantly lower than SMPC and PPFLEC. The integration of multimodal distillation in ELTFL contributes to faster convergence. The iterative mask employed by ELTFL has a smaller impact on model accuracy compared to the double mask and hash function used by SMPC and PPFLEC. In each training round, ELTFL requires a training time of 110.7 s, slightly higher than that of FedAvg due to iterative mask computation, remaining much lower than that of SMPC and PPFLEC, underscoring the lightweight nature of ELTFL. Regarding communication time per round, ELTFL has the shortest communication time, as it transmits data with lower dimensions during communication, which is the product of the model result dimension and the data batch dimension. In contrast, the other three schemes transmit data with the dimensions of the model

Table 6 Comparison of accuracy at different levels of data heterogeneity. Bold indicates the optimal value.

(%)

Dataset	Setting	FedAvg	SMPC	PPFLEC	FedDF	FedDistill	FedMD	ELTFL
CIFAR10	$\alpha = 0.01$	19.47	18.43	19.32	25.40	21.53	42.84	**57.32**
	$\alpha = 0.10$	69.36	68.64	69.46	71.43	67.28	74.57	**75.67**
	$\alpha = 100.00$	81.25	80.87	81.34	81.47	81.64	82.34	**88.33**
CIFAR100	$\alpha = 0.01$	15.32	14.77	15.23	21.35	18.46	33.65	**48.35**
	$\alpha = 0.10$	34.73	33.25	34.02	35.93	33.48	37.52	**52.67**
	$\alpha = 100.00$	45.25	44.87	45.32	44.67	45.68	53.63	**54.12**

Table 7 Comparison of computation, communication time, and number of rounds for model convergence for different methods.

Dataset	Method	T_{train} of each round (s)	T_{comm} of each round (s)	Number of rounds to convergence
CIFAR10, IID	FedAvg	108.6	2.40	125
	SMPC	178.3	4.00	196
	PPFLEC	120.2	10.40	143
	ELTFL	110.7	2.01	100
CIFAR100, IID	FedAvg	147.5	2.50	220
	SMPC	220.3	4.20	245
	PPFLEC	159.6	10.60	235
	ELTFL	107.9	2.03	110

Table 8 Total time required for convergence of different methods within different clusters. Edge cluster 1 is a cluster consisting of edge server 1 and all clients communicating with it, and edge cluster 2 is a cluster consisting of edge server 2 and all clients communicating with it.

Edge cluster	Method	Total time (s)
1	ELTFL	12 092.3
	FedAvg	32 062.5
	SMPC	53 460.0
	PPFLEC	39 218.4
2	ELTFL	12 387.2
	FedAvg	34 350.0
	SMPC	55 002.5
	PPFLEC	39 997.0

parameters.

Similarly, on the CIFAR100 dataset, ELTFL requires far fewer rounds to converge than the other three schemes. In terms of training time per round, ELTFL is lower than FedAvg and PPFLEC, and much lower than SMPC. In terms of communication time per round, ELTFL is also the least. In conclusion, ELTFL has low computational and communication overheads and fast convergence.

To evaluate the training efficiency across different devices, we calculate the total convergence time, which includes both training time and communication time, for two edge clusters with various data distributions on CIFAR10. Table 8 presents these results, showing that ELTFL incurs lower time costs than the other schemes across diverse device clusters and datasets. Specifically, on the CIFAR10 dataset, ELTFL exhibits a slightly lower time cost compared to FedAvg, and a significantly lower time cost compared to SMPC and PPFLEC. On the CIFAR100 dataset, ELTFL's time cost is much lower than that of the other three schemes. Furthermore, ELTFL demonstrates the minimal variation in time overhead across devices, remaining within a 300-s range. In comparison, time overhead of FedAvg varies by over 800 s, SMPC by over 1700 s, and PPFLEC by over 700 s.

6.5 Privacy protection analysis

In this section, we investigate these four schemes by employing deep leakage from gradients (DLG) attacks[44] to evaluate the possibility of reconstructing the original image from the transmitted data, potentially compromising the privacy of the original data. DLG attacks are particularly effective against traditional FL gradient transmission, which can restore datasets by simulating the gradient of model. We randomly select an image from the CIFAR100 dataset. DLG attacks generate an image at random, allowing DLG attacks to generate a random image and subsequently attempt to reconstruct the original image by simulating the gradients in each training round.

The presence of Fig. 6 indicates a successful DLG attack. Additionally, the loss value of the simulated gradient starts at 10 000.0, indicating complete image recovery. Conversely, if Fig. 7 is observed, it signifies a failed DLG attack, with a loss value of 83 786.1. After 270 simulation iterations, the loss value decreases to 83 746.8, demonstrating that the original image cannot be recovered. We conduct tests on 6400 images from the CIFAR10 dataset and obtain the attack results depicted in Fig. 8.

In Fig. 8, the ASR is shown, where the blue portion represents attacks between terminal devices and edge servers, while the red portion represents attacks between edge servers and the cloud. The ASR of the DLG attack between edge servers and terminal devices is 0%. ELTFL uses iterative masking at the edge, which hides data and prevents gradient reconstruction, protecting privacy during learning. In addition, since

Fig. 6 Success of gradient leak attack.

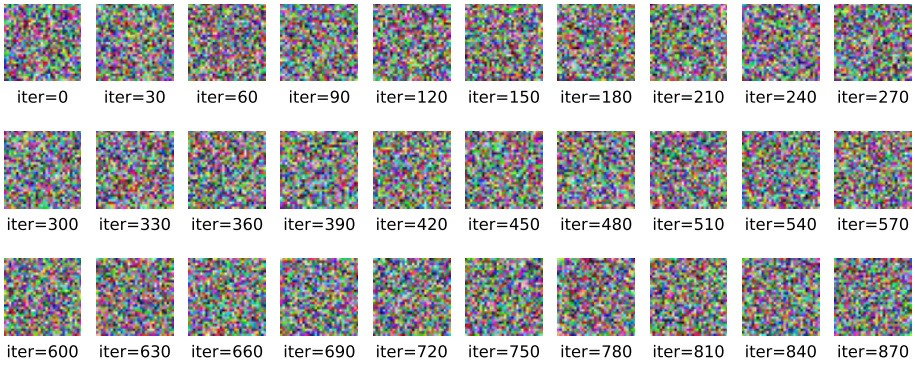

Fig. 7 Failure of gradient leak attack.

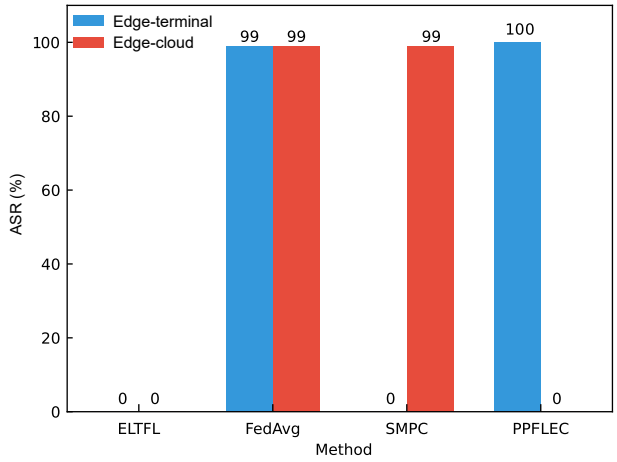

Fig. 8 Attack success rate diagram for four schemes.

the parameters of the global model are transmitted between edge servers and the cloud, the attacker is unable to restore the original image by gradient computation.

In comparison, FedAvg exhibits a high ASR exceeding 99% between the edge cloud and the edge, indicating that FL without encryption or masking risks significant data leakage. In the case of attacking SMPC, the attack success rate is 0% due to the presence of double masks between the edges. However, since no mask is added between edge servers and cloud, the image can be restored. It is important to note that the image data obtained by the attacker are based on the aggregated gradient and may not necessarily reflect the original data.

With PPFLEC, the edge data are not adequately protected and are directly aggregated at edge servers, making it susceptible to leakage. However, due to the inclusion of masks and hash functions between edge servers and the cloud, the data cannot be restored. In conclusion, ELTFL's effective integration of intelligent obfuscation and secure transmission proves highly suitable for privacy-preserving training in cloud-edge-terminal scenarios, demonstrating a robust balance of privacy and efficiency over competing methods.

7 Conclusion and Future Work

This paper presents a trusted heterogeneous FL scheme for edge scenarios, which has several notable strengths. First, it allows heterogeneous models from different end devices to contribute to the global model without requiring these devices to adopt uniform models, thereby maximizing data utilization across diverse devices. Second, the integrated distillation approach helps to reduce the burden on resource-constrained devices, allowing them to offload computationally intensive tasks to edge servers. Third, by incorporating a lightweight iterative masking mechanism, ELTFL ensures robust protection against privacy attacks, including gradient leakage, enhancing both privacy and security in FL.

Despite its advantages, ELTFL has some limitations. One of the key challenges is that the effectiveness of KD largely depends on the quality and diversity of the small datasets used at the edge, which might not fully represent the data distribution of all participating devices. Furthermore, communication efficiency between the edge and cloud could still be improved, especially when scaling to large numbers of devices and datasets.

In future research, we aim to investigate more efficient and adaptive KD techniques that can handle a wider range of heterogeneous models while further reducing communication costs. In addition, we plan to expand the scope of experiments to include larger and more diverse datasets, as well as real-world edge computing applications, to make ELTFL more robust in edge scenarios.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2022YFB3304303), the National Natural Science Foundation of China (Nos. 62271128 and U2333207), the Chengdu Key R&D Support Plan "Unveiling and Leading" Project (No. 2022-JB00-00013-GX), the Sichuan Science and Technology Plan "Unveiling and Leading" Project (No. 2023YFG0374), and the Key R&D Projects of Sichuan Provincial Science and Technology Plan (Nos. 2022ZDZX004, 2023YFG0029, and 2023YFG0150).

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[2] W. Liu, X. Xu, J. Wu, and J. Jiang, Federated meta reinforcement learning for personalized tasks, *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 911–926, 2024.

[3] N. A. Jalali and H. Chen, Federated learning security and privacy-preserving algorithm and experiments research under Internet of Things critical infrastructure, *Tsinghua Science and Technology*, vol. 29, no. 2, pp. 400–414, 2024.

[4] T. Wu, W. Dou, F. Wu, S. Tang, C. Hu, and J. Chen, A deployment optimization scheme over multimedia big data for large-scale media streaming application, *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 12, no. 5s, p. 73, 2016.

[5] D. Li, J. Lai, R. Wang, X. Li, P. Vijayakumar, B. B. Gupta, and W. Alhalabi, Ubiquitous intelligent federated learning privacy-preserving scheme under edge computing, *Future Gener. Comput. Syst.*, vol. 144, pp. 205–218, 2023.

[6] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, Heterogeneous federated learning: State-of-the-art and research challenges, *ACM Comput. Surveys*, vol. 56, no. 3, p. 79, 2023.

[7] W. Yuan, L. Qu, L. Cui, Y. Tong, X. Zhou, and H. Yin, HeteFedRec: Federated recommender systems with model heterogeneity, arXiv preprint arXiv: 2307.12810, 2023.

[8] G. Ye, T. Chen, Q. V. Hung Nguyen, and H. Yin, Heterogeneous decentralised machine unlearning with seed model distillation, *CAAI Trans. Intell. Technol.*, vol. 9, no. 3, pp. 608–619, 2024.

[9] G. Ye, T. Chen, Y. Li, L. Cui, Q. V. H. Nguyen, and H. Yin, Heterogeneous collaborative learning for personalized healthcare analytics via messenger distillation, *IEEE J. Biomed. Health Inform.*, vol. 27, no. 11, pp. 5249–5259, 2023.

[10] M. Imran, H. Yin, T. Chen, Z. Huang, X. Zhang, and K. Zheng, DDHH: A decentralized deep learning framework for large-scale heterogeneous networks, in *Proc. 2021 IEEE 37th Int. Conf. Data Engineering* (*ICDE*), Chania, Greece, 2021, pp. 2033–2038.

[11] E. Shang, H. Liu, J. Zhang, R. Zhao, and J. Du, Ensemble knowledge distillation for federated semi-supervised image classification, *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 112–123, 2025.

[12] D. Li and J. Wang, FedMD: Heterogenous federated learning via model distillation, arXiv preprint arXiv: 1910.03581, 2019.

[13] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, Ensemble distillation for robust model fusion in federated learning, in *Proc. 34th Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2020, pp. 2351–2363.

[14] L. Qi, C. Hu, X. Zhang, M. R. Khosravi, S. Sharma, S. Pang, and T. Wang, Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment, *IEEE Trans. Industr. Inform.*, vol. 17, no. 6, pp. 4159–4167, 2021.

[15] W. Dou, W. Tang, X. Wu, L. Qi, X. Xu, X. Zhang, and C. Hu, An insurance theory based optimal cyber-insurance contract against moral hazard, *Inf. Sci.*, vol. 527, pp. 576–589, 2020.

[16] R. Wang, J. Lai, X. Li, D. He, and M. K. Khan, RPIFL: Reliable and privacy-preserving federated learning for the Internet of Things, *J. Netw. Comput. Appl*., vol. 221, p. 103768, 2024.

[17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, Practical secure aggregation for privacy-preserving machine learning, in *Proc. 2017 ACM SIGSAC Conf. Computer and Communications Security*, Dallas, TX, USA, 2017, pp. 1175–1191.

[18] K. Wei, J. Li, C. Ma, M. Ding, W. Chen, J. Wu, M. Tao, and H. V. Poor, Personalized federated learning with differential privacy and convergence guarantee, *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4488–4503, 2023.

[19] G. Hinton, O. Vinyals, and J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv: 1503.02531, 2015.

[20] D. Jiang, C. Shan, and Z. Zhang, Federated learning algorithm based on knowledge distillation, in *Proc. 2020 Int. Conf. Artificial Intelligence and Computer Engineering* (*ICAICE*), Beijing, China, 2020, pp. 163–167.

[21] W. Yuan, C. Yang, L. Qu, Q. V. H. Nguyen, J. Li, and H. Yin, Hide your model: A parameter transmission-free federated recommender system, arXiv preprint arXiv: 2311.14968, 2024.

[22] H. Yin, L. Qu, T. Chen, W. Yuan, R. Zheng, J. Long, X. Xia, Y. Shi, and C. Zhang, On-device recommender systems: A comprehensive survey, arXiv preprint arXiv: 2401.11441, 2024.

[23] F. Sattler, T. Korjakow, R. Rischke, and W. Samek, FedAUX: Leveraging unlabeled auxiliary data in federated learning, *IEEE Trans. Neural Network. Learn. Syst.*, vol. 34, no. 9, pp. 5531–5543, 2023.

[24] J. Wang, X. Yang, S. Cui, L. Che, L. Lyu, D. Xu, and F. Ma, Towards personalized federated learning via heterogeneous model reassembly, in *Proc. 37th Int. Conf. Neural Information Processing Systems*, New Orleans, LA, USA, 2023, pp. 29515–29531.
[25] L. Yi, H. Yu, G. Wang, and X. Liu, FedLoRA: Model-heterogeneous personalized federated learning with LoRA tuning, arXiv preprint arXiv: 2310.13283, 2023.
[26] Z. Zhu, J. Hong, and J. Zhou, Data-free knowledge distillation for heterogeneous federated learning, in *Proc. 38th Int. Conf. Machine Learning*, Online, 2021, pp. 12878–12889.
[27] L. Zhang, L. Shen, L. Ding, D. Tao, and L. Duan, Fine-tuning global model via data-free knowledge distillation for non-IID federated learning, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition* (*CVPR*), New Orleans, LA, USA, 2022, pp. 10164–10173.
[28] X. Gong, A. Sharma, S. Karanam, Z. Wu, T. Chen, D. Doermann, and A. Innanje, Preserving privacy in federated learning with ensemble cross-domain knowledge distillation, arXiv preprint arXiv: 2209.04599, 2022.
[29] S. Han, S. Park, F. Wu, S. Kim, C. Wu, X. Xie, and M. Cha, FedX: Unsupervised federated learning with cross knowledge distillation, arXiv preprint arXiv: 2207.09158, 2022.
[30] Z. Chen, P. Tian, W. Liao, X. Chen, G. Xu, and W. Yu, Resource-aware knowledge distillation for federated learning, *IEEE Trans. Emerg. Top. Comput.*, vol. 11, no. 3, pp. 706–719, 2023.
[31] Y. Qiao, A. Adhikary, K. T. Kim, C. Zhang, and C. S. Hong, Knowledge distillation assisted robust federated learning: Towards edge intelligence, in *Proc. ICC 2024-IEEE Int. Conf. Communications*, Denver, CO, USA, 2024, pp. 843–848.
[32] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, Time aware distributed service recommendation with privacy preservation, *Inf. Sci.*, vol. 480, pp. 354–364, 2019.
[33] F. Fei, S. Li, H. Dai, C. Hu, W. Dou, and Q. Ni, A *k*-anonymity based schema for location privacy preservation, *IEEE Trans. Sustain. Comput*., vol. 4, no. 2, pp. 156–167, 2019.
[34] S. Kumari, M. Karuppiah, A. K. Das, X. Li, F. Wu, and N. Kumar, A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers, *J. Supercomput.*, vol. 74, no. 12, pp. 6428–6453, 2018.
[35] F. Benhamouda, A. Degwekar, Y. Ishai, and T. Rabin, On the local leakage resilience of linear secret sharing schemes, *J. Cryptol.*, vol. 34, no. 2, p. 10, 2011.
[36] R. Wang, J. Lai, Z. Zhang, X. Li, P. Vijayakumar, and M. Karuppiah, Privacy-preserving federated learning for Internet of Medical Things under edge computing, *IEEE J. Biomed. Health Inform.*, vol. 27, no. 2, pp. 854–865, 2023.
[37] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K. Y. Lam, Local differential privacy-based federated learning for Internet of Things, *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, 2021.
[38] Y. Wang, Y. Tian, X. Yin, and X. Hei, A trusted recommendation scheme for privacy protection based on federated learning, *CCF Trans. Netw.*, vol. 3, no. 3, pp. 218–228, 2020.
[39] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, FedED: Federated learning via ensemble distillation for medical relation extraction, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing*, Online, 2020, pp. 2118–2128.
[40] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, A robust ECC-based provable secure authentication protocol with privacy preserving for industrial Internet of Things, *IEEE Trans. Industr. Inform.*, vol. 14, no. 8, pp. 3599–3609, 2018.
[41] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues, Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications, *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4755–4763, 2019.
[42] A. Yazdinejad, A. Dehghantanha, H. Karimipour, G. Srivastava, and R. M. Parizi, A robust privacy-preserving federated learning model against model poisoning attacks, *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 6693–6708, 2024.
[43] G. Luo, N. Chen, J. He, B. Jin, Z. Zhang, and Y. Li, Privacy-preserving clustering federated learning for non-IID data, *Future Gener. Comput. Syst.*, vol. 154, pp. 384–395, 2024.
[44] H. Yang, M. Ge, D. Xue, K. Xiang, H. Li, and R. Lu, Gradient leakage attacks in federated learning: Research frontiers, taxonomy, and future directions, *IEEE Network*, vol. 38, no. 2, pp. 247–254, 2024.

Chuanwen Luo received the PhD degree from School of Information, Renmin University of China, Beijing, China, in 2020. He is currently an assistant professor at School of Information Science and Technology, Beijing Forestry University, China. He was a visiting scholar at Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA, in 2019. His research interests include various topics in the application of wireless networks, ad hoc and sensor networks, and algorithm design and analysis.

Jinshan Lai received the BS degree from University of Electronic Science and Technology of China, Chengdu, China, in 2021. He is currently pursuing the MS degree at University of Electronic Science and Technology of China, China. He is a China Computer Federation (CCF) student member. His research interests include federated learning and aggregation security, network and information security, and blockchain.

Fengli Zhang received the PhD degree from University of Electronic Science and Technology of China, Chengdu, China, in 2007. She is currently a professor at University of Electronic Science and Technology of China, China. Her research interests include database system and security, mobile data management, and network security. As the project leader or chief researcher, she has completed a number of national key scientific and technological research projects during the "Sixth Five Year Plan", "Seventh Five Year Plan", "Eighth Five Year Plan", and "Ninth Five Year Plan", and has successively participated in, organized, designed, and implemented a number of application projects, which have won the first prize and the third prize of Sichuan Science and Technology Progress Award respectively. She has participated in several projects of Professor Ouri Wolfson in mobile object database, mobile computing, temporal and spatial database, etc., won one invention, and published 5 papers in cooperation. Since 2002, as a project and technology, she has been responsible for the research and implementation of many projects such as the National 863 Program, the Natural Science Foundation of China, the Electronic Development Fund, the National Network Security Center Fund, and the Sichuan Provincial Fund. She has published more than 60 articles and 3 books in magazines and international conferences at home and abroad.

Ruijin Wang received the PhD degree from University of Electronic Science and Technology of China, Chengdu, China, in 2013. He is currently an associate professor at University of Electronic Science and Technology of China, China. He is also a senior member of CCF, member of the Asia Pacific Artificial Intelligence Association (AAIA), secretary general of the Chengdu Branch of the International Computer Society, visiting scholar at Northwestern University, Chicago, IL, USA, and senior overseas scholar from Sichuan Province. His research interests include cloud edge intelligent computing, data processing, artificial intelligence, information security, and blockchain. He is also an expert from the Ministry of Science and Technology, Ministry of Industry and Information Technology, National Natural Science and Technology Commission, Provincial Department of Science and Technology, and Department of Economy and Information Technology. He has published over 40 academic papers, 4 monographs/textbooks, 17 invention patents, and participated in the development of 3 international and industry standards in important journals and conferences such as *Software Journal* and *IEEE Journal of Biomedical and Health Informatics*.