LightFed: An Efficient and Secure Federated Edge Learning System on Model Splitting

Jialin Guo, Jie Wu, *Fellow, IEEE*, Anfeng Liu, Neal N. Xiong*, *Senior Member, IEEE*

Abstract—With the integration of Artificial Intelligence (AI) and Internet of Things (IoT), the Federated Edge Learning (FEL), a promising computing framework is developing. However, there are still unsolved issues on communication efficiency and data security due to the huge models and unreliable transmission links. To address these issues, this paper proposes a novel federated edge learning system, called LightFed, where the edge nodes upload only vital partial local models, and successfully achieve lightweight communication and model aggregation. First, a novel model aggregation method Model Splitting and Splicing (MSS) and a Selective Parameter Transmission (SPT) scheme are proposed. By detecting the updating gradients of local parameters and filtering significant parameters, selective rotated transmission and efficient aggregation of local models are achieved. Second, a Training Filling Model (TFM) is proposed to infer the total data distribution of edge nodes, and train a filling model to mitigate the unbalanced training data without violating the data privacy of individual users. Moreover, a blockchain-powered confusion transmission mechanism is proposed for defending the attacks from external adversaries and protecting the model information. Finally, extensive experimental results demonstrate that our LightFed significantly outperforms the existing FEL systems in terms of communication efficiency and privacy security.

Index Terms—Federated Edge Learning, Communication Efficiency, Privacy Protection, Deep Neural Network

1 Introduction

With the advancement of microprocessors and 5G communication technologies, the storing and processing of massive data shift from the network center to the edge [1], and the Edge Computing (EC) paradigm becomes extensively utilized [2]. Meanwhile, the rapid development of Artificial Intelligence (AI) provides a solid foundation for mining the potential value of edge data information in depth [3], [4]. Therefore, Federated Edge Learning (FEL), a novel computing framework for AI fusion EC, was developing [5], [6]. Unlike other distributed computing frameworks, Edge Nodes (EN) in FEL never directly exchange source data, but instead a Global Server (GS) orchestrates ENs and aggregates the parameters of local AI models. This mode preserves node data privacy and allows for secure data sharing [7].

Although FEL solves the challenges of large computation volume and high storage capacity requirements for devices in traditional DL while also protecting the data privacy to some extent [5], [7], FEL still has many issues with communication, training efficiency and higher-level security assurance [8].

First, although nodes in basic FEL never communicate source data, the ENs should still upload local model parameters to the GS periodically [7]. However, Deep Neural Networks (DNN) are now growing in size in tandem with the complexity of training tasks, and the model parameter amount is extremely high. For instance, the classical network structure UNet for analyzing medical images might include tens of millions of parameters [9]. The communication efficiency between ENs and the GS will certainly be reduced if such a huge model is transmitted in each model aggregation. To address this issue, Wu et al. [10] and Wang et al. [11] adaptively modify the model aggregation frequency to reduce the communication amount. However, the redundant model aggregation method FedAvg [7] is still kept in their schemes, which is not conductive to the communication efficiency. Therefore, a novel lightweight model aggregation technique should be suggested, which will function better with the communication strategies used in the previous studies.

Second, unlike centralized machine learning, the data distribution of ENs may not be uniform and compatible with the expectation of GS [12], resulting in a drop in training efficiency and the accuracy of global model. To compensate the impact of non-independent-identical distribution (non-IID) on FEL system, the Astraea framework introduces mediators into the FEL architecture to achieve the detection and adjustment for imbalanced client data [13]. In addition, FedHome implements the generation of approximately balanced dataset by a lightweight GCAE encoder [14]. The above methods indeed never leak the data of ENs, but it reveals the EN data distribution to the mediators or GS during data adjustment, which is still the privacy in a wide sense. The user profile is likely to be detected by the GS or external adversaries based on the data distribution [15]. Therefore, it is valuable and challenging to design a data adjustment strategy that does not reveal the data distribution of ENs.

Despite the fact that FEL protects the source data, some attacks, such as inverse attack [16], [17], nevertheless exploit the local model to deduce the data features of ENs backwards.

- *Jialin Guo and Anfeng Liu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China. E-mail: {guojialin@csu.edu.cn, afengliu@mail.csu.edu.cn}.*
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA. E-mail: jiewu@temple.edu.*
- *Neal N. Xiong is with the Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX 79830, USA. E-mail: xiongnaixue@gmail.com.*

*Manuscript received ***; revised ***; accepted ***.*
*Date of publication ***; date of current version ***.*
(Corresponding author: Neal N. Xiong.)

The GS can directly acquire the local models of ENs, while external adversaries may obtain model parameters by tapping the wireless channel, and ENs may infer the data distribution of other nodes by the inverse calculation of aggregation after receiving the global model [16]. Furthermore, this type of attack is more effective in the case of data imbalance. Some works have applied differential privacy methods [18], [19], but they introduce noise while altering the local models, lowering the purity of models and maybe reducing the efficacy of training. As a result, we need to build a new general model encryption transmission scheme to secure the user privacy at a higher level without compromising model accuracy.

To address the aforementioned issues with FEL, we present a novel communication-efficient and secure FEL system LightFed. LightFed provides a lightweight aggregation method by reducing the parameter amount uploaded by ENs, and encrypts the parameter by quick hash transformations. Our primary contributions are as follows:

- We first propose a lightweight model aggregation method Model Splitting and Splicing (MSS) and a Selective Parameter Transmission Protocol (SPT) that differ from classic FedAvg. In model aggregation, SPT enables automatic detection for model parameters and filters out parameter vectors with large gradients. The ENs should only rotatedly upload partial vectors to the GS, and the GS can achieve efficient aggregation of local parameters.
- A novel data adjustment method Training Filling Model (TFM) is suggested. TFM finds the minor classes in the training data by the global test result, fills the data gap and balances the dataset by training a filling model. It never reveals the data distribution of ENs and GS, thoroughly ensuring the user privacy.
- Based on SPT, a blockchain-powered parameter confusion transmission mechanism is proposed for the security of model information. The GS issues hash keys in the DAG blockchain for ENs. ENs can quickly shuffle the uploading parameters by the unique hash key to confuse external eavesdroppers, accomplishing the lightweight model encryption.

The rest of this paper is organized as follows. Section 2 reviews the related work. The system model and research motivation are presented in Section 3. Section 4 introduces our novel FEL system LightFed. Then, Section 5 provides the experimental results and an engineering application of LightFed. Finally, conclusion and future work are given in Section 6.

2 Related Work

Machine Learning (ML), as an emerging technique that digs deeply into the inherent rules and representation of data, has produced excellent results in processing types of data such as images [20], natural language [21], and sound [22]. Based on these results, ML can provide accurate prediction or decision for both civilian and industrial systems [23]. However, facing complex data processing tasks, ML models must execute computation with massive sample data, which undoubtedly poses a challenge to the storage capacity and computing power of devices. Moreover, network failure and hardware failure are becoming more frequent as storage capacity of centralized nodes scales up [24]. To address these issues, distributed ML is proposed, which trains models in parallel across several computing nodes, reducing the load on central server and the training time. Petuum [25], GraphLab [26] and Parameter Server [27] are the classical distributed ML platforms.

To further guarantee the data privacy of computing nodes, Google first proposed the concept of Federated Learning (FL) based on the theory of distributed ML [28]. If the computing nodes are common edge nodes, it is called Federated Edge Learning (FEL) [5]. FEL advocates recruiting resource-free edge nodes to achieve joint training of ML models. Each edge node holds private training data and never exchanges data directly during training, but shares updated model parameters. A global server (GS) aggregates the local models by weightedly averaging the model parameters from edge nodes to obtain a global model with better performance, which is the classical FedAvg algorithm [7]. The privacy-preserving and scalability of FEL have led to its rapid adoption in various fields, such as healthcare [29], industrial control [30], traffic monitoring [31].

FEL lowers the application barrier for distributed ML, however owing to the intrinsic restrictions in FEL, there are still some issues with communication efficiency and security. In addition to the aforementioned strategies to reduce communication frequency, Konecny et al. [32] employed model compression approaches to reduce the size of local models to improve the uplink communication. Smith et al. [33] investigated that multi-task federated learning can naturally deal with the problem of large communication volume in conventional federated learning, and suggested the novel approach MOCHA to improve the communication. However, varying form the above studies, LightFed innovates the aggregation method by splitting local models and considers different update patterns of parameters. Second, with the development of attack techniques, Melis et al. and Hitaj et al. proposed novel attack methods to obtain data features by stealing model parameters [16], [17], and higher-level security defenses are yet to be proposed. Lu et al. [34] enhanced the security performance of federation learning by introducing global blockchain and local DAGs. Dai et al. [35] proposed a fusion AI and blockchain distributed resource sharing platform, improving the security of resource allocation at edge nodes. However, considering the access overhead of models in the blocks in previous research, LightFed provides a lightweight blockchain empowered encryption method integrated to the model transfer, where the blocks are only used to store hash keys with small volume.

3 System Model and Research Motivation

3.1 System Model

The LightFed proposed in this paper enables secure and fast federated model training by recruiting heterogeneous ENs, widely used in various distributed ML scenarios, including UAV-assisted rescue [36], weather data prediction [2], traffic flow monitoring [31], etc. The basic components of LightFed are shown in Fig. 1, primarily including a global server and edge nodes. (1) Global Server (GS): There are some devices equipped with high-performance servers, such as UAVs, small base stations, RSUs, etc. They act as the initiators and controllers of federated learning tasks, and orchestrates the training of nodes and aggregates local models. Meanwhile, some GSs hold private data, e.g., UAVs obtain image data through aerial photography, and can also participate in model

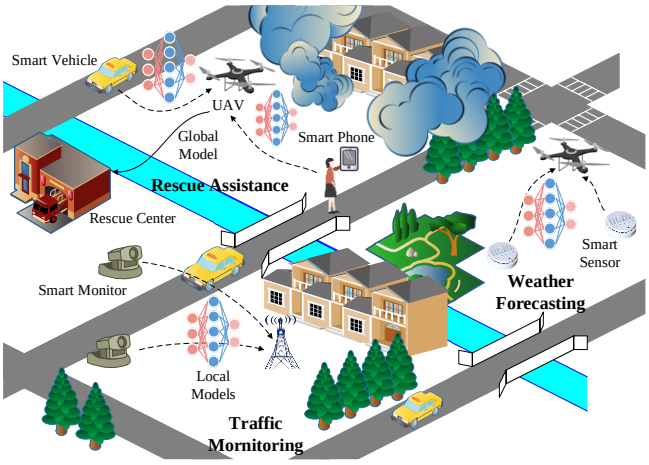


Fig. 1 System model and applications

Table 1
Main symbols and their meanings

Symbol	Meaning
$\boldsymbol{N}_i$	The edge node set in i-th aggregation
$\boldsymbol{w}_i$	The global model after i-th aggregation
$\boldsymbol{w}_i^a$	The local model of n_a after i-th aggregation
$\mathbb{D}_a$	The local training set of n_a
$\mathcal{C}$	The communication efficiency indicator
$\mathcal{S}$	The data security indicator
acc_g^i	The global test accuracy after i-th aggregation
$\mathcal{B}$	The data unbalance indicator
$\mathcal{S}_g^i, \mathcal{S}_a^i, \mathcal{S}_e^i$	The defending indicators of GS, d_a and external adversary
m_r^i	The parameter redundancy in i-th aggregation
d_r	The decay ratio for parameter redundancy
m_a^i	The transmission amount of n_a in i-th aggregation
$\mathcal{N}_v^l$	The vector number in l-th layer of model
ξ_b, ξ_u	The thresholds for LBP and SUP
$\boldsymbol{w}_i^{fill}$	The filling model in i-th aggregation
ρ_c^i	The filling ratio for class c in i-th aggregation
F_{unit}^i	The unit size of filling data in i-th aggregation

training as trainers. (2) Edge Nodes (EN): a huge number of heterogeneous mobile smart devices, such as smart vehicles on the road, cellphones of pedestrians, smart cameras and sensors, etc., are in charge of receiving global model from the GS and updating local models. To indicate the available EN set, we define $\boldsymbol{N} = \{n_1, n_2, \dots n_i, \dots\}$. These ENs have limited computational resources and the capacity to collect data from various locations and motion trajectories. Therefore, they may obtain unique data to assist in improving the generalization capability of models.

3.2 Basic FEL Model

We introduce basic federated learning using an image recognition training task. By federated learning, ENs can use their image data to assist GS in training image recognition models. First, the GS determines the model type and size, such as the classical Convolutional Neural Network (CNN) or Support Vector Machine (SVM), etc. [37], and initializes the global model parameter as $\boldsymbol{w}_0$, which is sent to the ENs that have verified their identities. EN n_a starts the local parameter update once it receives the global model. First n_a needs to define the loss function $l_{n_a}(\boldsymbol{w}_i^{n_a})$, as shown in Equation (1). $\mathbb{D}_a$ denotes the private dataset of n_a, $\boldsymbol{w}_i^{n_a}$ is the local model parameter updated by n_a in the i-th round, and $\boldsymbol{x}_j$ and y_j are the j-th image sample in $\mathbb{D}_a$ and its corresponding label, respectively. $f(\boldsymbol{w}_i^{n_a}, \boldsymbol{x}_j, y_j)$ then characterizes the difference between the predicted label of sample $\boldsymbol{x}_j$ and its true label y_j according to the model parameter $\boldsymbol{w}_i^{n_a}$. The cross-entropy loss function and the logarithmic loss function, for example, are commonly employed [37].

$$l_{n_a}(\boldsymbol{w}_i^{n_a}) = \frac{1}{|\mathbb{D}_a|}\sum_{j=1}^{|\mathbb{D}_a|} f(\boldsymbol{w}_i^{n_a}, \boldsymbol{x}_j, y_j). \quad (1)$$

After several rounds of local parameter update, n_a and other ENs selected by GS upload the local parameters to GS for the first model aggregation to obtain the global model $\boldsymbol{w}_1$. The goal of model aggregation is to find the optimal model parameters $\boldsymbol{w}^*$ with the minimum average prediction loss over all ENs, i.e., maximizing the generalization ability of model, as shown in Equation (2). After the GS aggregates the local models, the new global model is issued to ENs again to start rounds of local parameter update.

$$\boldsymbol{w}^* = \min_{\boldsymbol{w}} \frac{1}{|\boldsymbol{N}|}\sum_{a=1}^{|\boldsymbol{N}|} l_{n_a}(\boldsymbol{w}_i^{n_a}). \quad (2)$$

3.3 Problem Statement

Basic FEL, as previously noted, makes use of distributed computing resources to solve the limitations of traditional ML, but it still faces many problems to implement a secure and efficient federal edge system. In this subsection, we state our goals and define the corresponding performance metrics.

- Communication Efficiency. As metioned above, the decentralization of data and computation in FEL leads to a higher communication cost. Therefore, the first goal of LightFed is to achieve a lightweight and efficient model communication strategy without degrading the model accuracy. We define $\mathcal{C}$ as a metric of communication efficiency, where I is the total number of model aggregations in the training task, and acc_g^i is the accuracy of the global model on the global test set after the i-th aggregation. $|\boldsymbol{W}_i|$ denotes the total number of parameters transmitted during the i-th model aggregation.

$$\mathcal{C} = \frac{1}{I}\sum_{i=1}^{I} \frac{acc_g^i}{|\boldsymbol{W}_i|}. \quad (3)$$

- Data Security. Due to the emergence of inversion attacks, the goal of privacy protection rises from source data security to model security. We primarily investigate three types of attackers: GS, ENs, and external adversaries. There is a premise that the first two types of attackers are "honest and curious" [38]: they follow basic communication protocols and learning patterns, but they exploit the data received to get as much information as possible.

We define $\mathcal{S}$ to denote the security of model transmission, as shown in Equation (4). $\mathcal{S}_g^i$, $\mathcal{S}_a^i$ and $\mathcal{S}_e^i$ denote the performance indicators of defending the attacks from GS, ENs and external adversaries, respectively, and ϑ_g, ϑ_d and ϑ_e are their weight coefficients. Equation (5) illustrates the calculation of $\mathcal{S}_g^i$, where $\mathbb{F}_{det}^i$ is the data features detected by GS at the i-th model aggregation, and $\mathbb{F}_{tar}$ is the true data features of the attacked target. $\|\cdot\|_2$ is used to calculate the difference between $\mathbb{F}_{det}^i$ and $\mathbb{F}_{tar}$. $\mathcal{S}_a^i$ and $\mathcal{S}_e^i$ are calculated in the similar way.

$$\mathcal{S} = \frac{1}{I}\sum_{i=1}^{I}\left(\vartheta_g \mathcal{S}_g^i + \frac{\vartheta_d}{|\boldsymbol{N}_i|}\sum_{a=1}^{|\boldsymbol{N}_i|}\mathcal{S}_a^i + \vartheta_e \mathcal{S}_e^i\right). \quad (4)$$

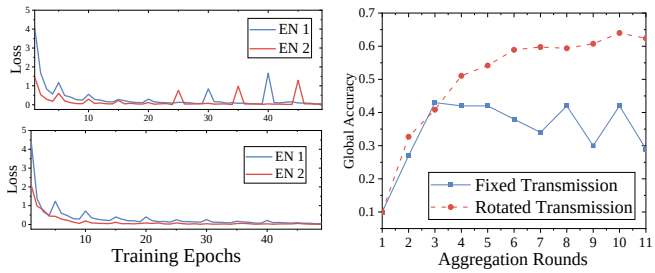


Fig. 2 The comparison of fixed transmission and rotated transmission on loss and global accuracy

$$\mathcal{S}_g^i = \left\| \mathbb{F}_{det}^i - \mathbb{F}_{tar} \right\|_2. \tag{5}$$

- Data Balance. The non independent identical distribution (non-IID) of EN data significantly affects the convergence speed and accuracy of FEL [13]. Moreover, to ensure the data privacy, ENs refuse to disclose their data distribution to the GS for the data adjustment. Under this premise, designing a mechanism for GS to snoop the training data profile and adaptively adjust the data distribution without requesting for EN privacy is a key issue.

We define $\mathcal{B}$ as a metric of the unbalance of training data distribution, as shown in Equation (6). C is the total number of data classes, $\sum_{n_a \in \boldsymbol{N}} |\mathbb{D}_a|$ is the amount of data used to update the model before the i-th aggregation, and $\sum_{n_a \in \boldsymbol{N}} |\mathbb{D}_a^j|$ is the number of samples in class j. The main symbols and their meanings in this paper are represented in Table 1.

$$\mathcal{B} = \frac{1}{I} \sum_{i=1}^{I} \sqrt{\frac{\sum_{j=1}^{C} \left(\frac{\sum_{n_a \in N_i} |\mathbb{D}_a|}{C} - \sum_{n_a \in N_i} |\mathbb{D}_a^j| \right)^2}{C}}. \tag{6}$$

3.4 Research Motivation

(1) Motivation of Model Splitting and Splicing (MSS)

To reduce model transmission, we first explore the efficacy of model splitting and splicing (MSS) for aggregation. In other words, each EN is responsible for transmitting partial parameters, e.g., one of the layers in the model, instead of the complete model. The GS receives distinct model segments uploaded by ENs and aggregates the models in a spliced manner. Note that this aggregation approach presupposes that ENs and the GS apply neural networks with same structure and size.

We first simulate two ENs for model splicing experiments. The EN datasets are created by sampling on the sorted MNIST dataset using a truncated Gaussian distribution. Therefore, the samples in each local dataset are unbalanced, meeting the non-IID in FEL assumption. The ENs upload model segments for splicing after every 5 epochs of local update. We first fix the layers transmitted by ENs. For example, in a two-layer model, EN 1 and EN 2 always uploads the first and the second layer respectively. The curves of training loss under fixed transmission are shown in the upper part of Fig. 2(a). The loss of both ENs decreases significantly during the early training phase, but after 20 epochs, the loss of both ENs begins to alternatively surge when the model is spliced. The blue curve in Fig. 2(b) shows the global accuracy of fixed transmission also decreasing in the later phase. This is because the fixed transmission makes the parameters of different layers updating only on the local training set of one particular EN, without touching other datasets. The excessive bias of the local model cannot be mitigated, resulting in a decrease in global accuracy. To solve the problem, we try to introduce a rotated transmission strategy, in which EN 1 and EN 2 alternately upload the model parameters, and the loss curves are given in the lower part of Fig. 2(a). Compared to fixed transmission, rotated transmission allows each layer of the model updating alternately on the two local datasets, resulting in improved model homogeneity and accuracy. To summarize, MSS aggregation is viable, but the rotated transmission mechanism must be applied to avoid the excessive local bias of parameters.

(2) Motivation of Selective Paramter Transmission (SPT)

In the training of neural networks, the update strength and the contribution to model convergence are not equivalent for each layer and even each parameter [12]. Therefore, we further speculate that ENs even have no need to transmit the complete layers by ignoring the parameters with little contribution. To explore the update of parameters in FEL, we performed the following simulation experiments to obtain Fig. 3.

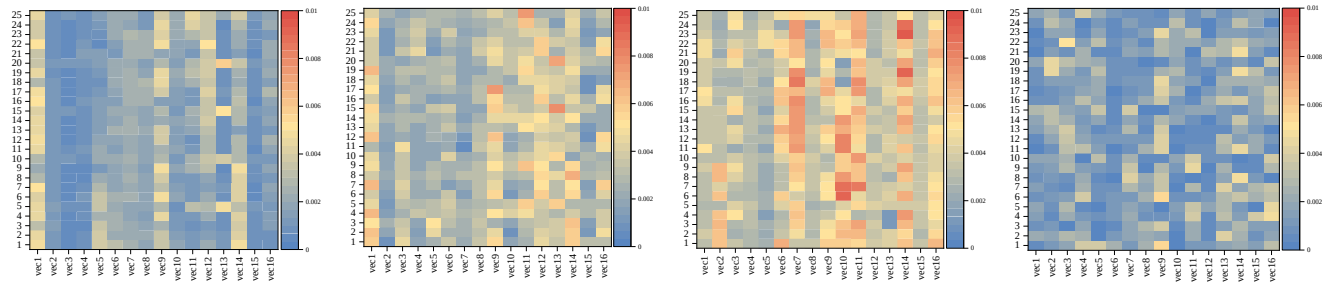
The first convolution layer of model is partitioned into 16 vectors in sequence. In Fig. 3(a), we observed some vectors, such as vector 1 and vector 9, have substantial variations of update across ENs, while vector 2 and vector 3, have lesser difference. (b) shows the parameter difference after 10 epochs of local update. Due to the different data distributions on the ENs, the same initial model trained on the respective datasets leads to a larger bias in distinct directions as the training epochs increase. Then we shrink the standard deviation of the data sampling distribution of the ENs, i.e., the σ in the Gaussian distribution, to make the data distribution of the ENs more inhomogeneous. After that ENs update locally by 10 epochs, and (c) was obtained. Comparing (c) and (b), the gap between the model parameters of ENs is much larger, which demonstrates the impact of data inhomogeneity on the model update direction. (d) shows the standard deviation of parameters between the local model of one EN and the global model after two rounds of aggregation with FedAvg. Compared to the separate training in (c), the overall difference in parameter update is reduced, and the model aggregation mitigates the undesirable bias of local models.

In summary, the update strength of the model parameters on ENs varies with the data distribution and aggregation. For the local parameters less different from the global model or with high similarity across ENs, repeated transmission is less necessary than other parameters with obvious update. Therefore, to further reduce the communication amount, we need to design a selective parameter transmission protocol based on MSS, and then ENs can adaptively select significant parameters for transmission.

4 Our Proposed LightFed

4.1 Overview of LightFed

The key components and workflow of LightFed are depicted in Fig. 4. First, the GS initializes the model parameters and selects EN group to establish connections, and then broadcasts the model to ENs. Meanwhile, according to the concept of MSS, the GS splits the model into smaller parameter blocks,



(a) (b) (c) (d)

Fig. 3 The heatmaps of the standard deviation of the parameters of the first convolution layer. (a) Standard deviation of the parameters of ENs, σ=10000, t_{loc}=5. (b) Standard deviation of the parameters of ENs, σ=10000, t_{loc}=10. (c) Standard deviation of the parameters of ENs, σ=1000, t_{loc}=10. (d) Standard deviation of the parameters of EN and GS applying FedAvg, σ=1000, t_{agg}=2.

which are composed of several parameter vectors. For each parameter block, each EN is assigned to transmit partial parameter vectors after local update. Then the GS collects the vectors uploaded by ENs and spliced them into complete parameter blocks to achieve the model aggregation. To solve the data imbalance, the GS analyzes the data distribution after aggregating model and samples filling data from its dataset to train the filling model. Moreover, to prevent external adversaries from stealing model information, the GS sets up blockchain-powered hash keys for ENs to encrypt data. The following subsections go through the specifics of LightFed.

4.2 Details of Model Splitting and Splicing (MSS)

(1) Model splitting. We assume that the GS and ENs apply the model with same structure. There are L layers in the model and $\mathcal{N}_v^l$ parameter vectors in the l-th layer. Therefore, the GS can split the total $\sum_{l=1}^{L}\mathcal{N}_v^l$ parameter vectors into b parameter blocks, each of which contains $\frac{\sum_{l=1}^{L}\mathcal{N}_v^l}{b}$ vectors. Assume that the GS selects EN group $\boldsymbol{N}_j$ at the j-th round of aggregation, and each EN needs to transmit partial vectors in each parameter block.

(2) Parameter assignment and rotated transmission. The GS employs a rotation counter to regulate the parameters ENs uploading in model aggregation, as seen in the lower right corner of Fig. 4. The rotation counter follows the concept of rotated transmission, i.e., the j-th EN in the i-th round of aggregation must upload the parameters transferred by the j+1-th EN in the $i-1$-th round. For example, in Fig. 4 the GS selects 4 ENs to train the model, and in the 1st round of aggregation, the rotation counter assigns EN 2 to transmit vector 5 to vector 12 in each parameter block, while in the 2nd round EN 2 needs to upload vector 1 to vector 8, which are the parameters uploaded by EN 1 in the last aggregation. In this way, each EN can upload its local model to GS in batches after at most $|\boldsymbol{N}_j|$ rounds of aggregation without replacing ENs. It is noted that every two or more adjacent ENs transmit parameters with overlapping parts. For example, in the 1st aggregation, both EN 1 and EN 2 upload vector 5 to vector 8, which is called parameter redundancy. When aggregating models, the GS weightedly averages the redundant vectors provided by EN 1 and EN 2. The purposes are mainly the following. 1) Provide a backup for some lost parameters due to undesirable communication conditions. 2) Achieve the average aggregation of partial parameters, which helps to further eliminate excessive local bias. 3) Calculate the difference between local models and select the parameter vectors with larger update strength to focus on in the next round of aggregation. This will be described detailedly in (3).

The parameter redundancy amount m_r is adaptively adjusted by GS. In the early training phase, the update directions of local models are inconsistent and the global model has not reached convergence, so it is more likely a large number of parameter vectors are excessively biased, as Fig. 3(c). Therefore, we need larger m_r for mitigating local bias and detecting the vectors with large update difference. However, in the late phase, as the model is about to converge the update difference between local models becomes smaller, and m_r can be lowered to further reduce the parameter transmission and shorten the training period. To implement the adaptive parameter redundancy, we define the calculation of m_r which is related to the convergency of model, as shown in Equation (7). i is the aggregation round, and Δacc_g^i is the difference between the global accuracy of i-th round and that of the last round. ϑ_{acc} and ϑ_i are the weighting factors of Δacc_g^i and i, respectively. λ_r is the correlation coefficient, and d_r is the decay exponent, which takes values from -1 to 0. The vector number to be transmitted by each EN m_a^i is shown in Equation (8), where $|\mathbb{D}_k|$ is the data amount of the k-th EN. For cases where new ENs join the training or there are ENs outages causing the EN number changes, the GS can quickly adjust m_a^i by Equation (9) and simply send the starting position and length of the new transmitting parameter segments to ENs.

$$m_r^i = \left\lceil \lambda_r \left(\vartheta_{acc}\Delta acc_g^i + \vartheta_i i\right)^{d_r} \right\rceil. \tag{7}$$

$$m_a^i = \left\lceil \frac{|\mathbb{D}_a|}{\sum_{k=1}^{|\boldsymbol{N}_i|}|\mathbb{D}_k|}\left(\frac{\sum_{l=1}^{L}\mathcal{N}_v^l}{b} + |\boldsymbol{N}_i| m_r^i\right) \right\rceil. \tag{8}$$

(3) Selective parameter transmission. Based on the deviation map of local models in Section 3.4, we classify the model parameter vectors into the following three types. The GS and ENs will detect the type of vectors and implement different transmission and aggregation strategies.

- Large Bias Parameters (LBP). LBP are the parameter vectors significantly updating compared to the previous global model and have large difference on local models, i.e., the vectors detected by parameter redundancy in (2). We specify two thresholds ξ_b and ξ_u for determining the types of parameter vectors. Thus, the parameter vectors satisfying Equation (9) are LBP in the i-th round of aggregation. $\sigma_b(\boldsymbol{v}_i^a)$ denotes the variance of the vector $\boldsymbol{v}_i$ in multiple local models, where $\boldsymbol{V}_i$ is the vector list of $\boldsymbol{v}_i$ collected by GS, and $\bar{\boldsymbol{V}}_i$ is the average value of $\boldsymbol{V}_i$. $\sigma_u(\boldsymbol{v}_i^a)$ denotes the difference of $\boldsymbol{v}_i$ in the model of n_a, and the corresponding vector $\boldsymbol{v}_i^g$ in the global model.

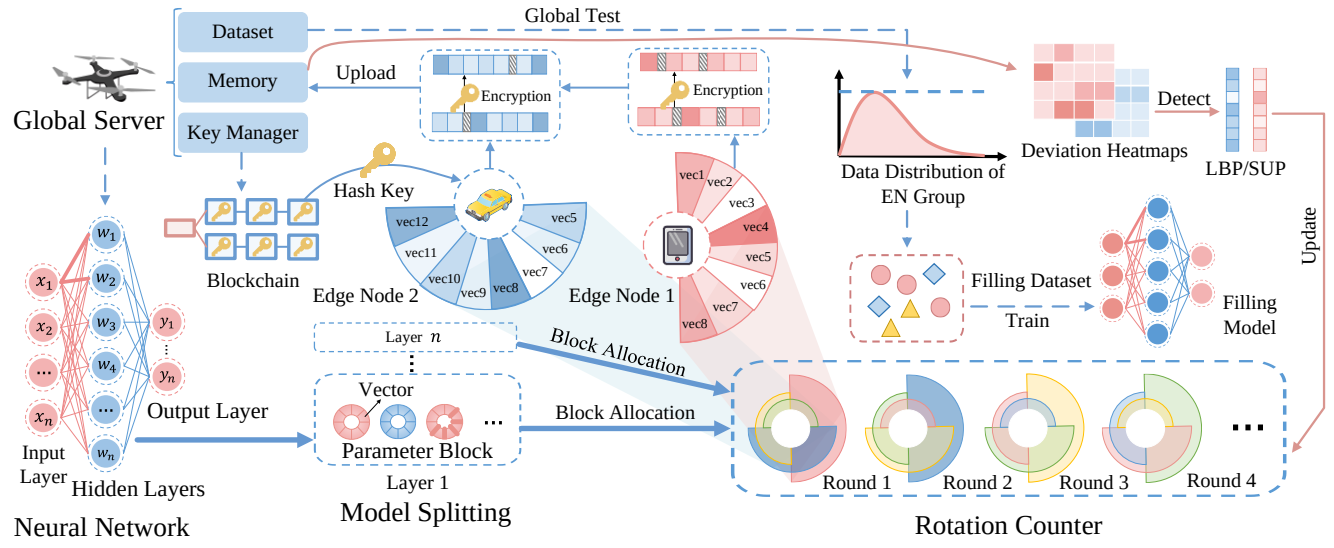


Fig. 4 The main components and workflow of LightFed

$$\boldsymbol{v}_i \in \text{LBP}, \sigma_b(\boldsymbol{v}_i^a) > \xi_b \wedge \sigma_u(\boldsymbol{v}_i^a) > \xi_u. \tag{9}$$

$$\sigma_b(\boldsymbol{v}_i^a) = \frac{1}{|\boldsymbol{V}_i|}\sum_{j=1}^{|\boldsymbol{V}_i|}\left\|\boldsymbol{v}_i^j - \overline{\boldsymbol{V}}_i\right\|_2. \tag{10}$$

$$\sigma_u(\boldsymbol{v}_i^a) = \left\|\boldsymbol{v}_i^a - \boldsymbol{v}_i^g\right\|_2. \tag{11}$$

Algorithm 1 SPT Protocol

Input: $\boldsymbol{D}$, I_{agg}
Output: $\boldsymbol{w}_{I_{agg}+1}$

GS initializes the global model $\boldsymbol{w}_0$ and LBP list
for $i = 0 \ldots I_{agg}$:
  GS selects $\boldsymbol{N}_i$ and sends $\boldsymbol{w}_i$
  GS computes m_r^i by Equation (7)
  for n in $\boldsymbol{N}_i$:
    GS computes m_a^i by Equation (8)
  end for
  GS updates rotation counter by m_a^i and LBP list
  GS issues vector index list to $\boldsymbol{N}_i$
  for n in $\boldsymbol{N}_i$:
    n updates $\boldsymbol{w}_i^n$ for several epochs
    n creates vector list based on vector index list
    n detects SUP and replaces SUP by placeholders
    n uploads vector list
  end for
  GS receives vector lists from $\boldsymbol{N}_i$ and detects LBP
  GS updates LBP list
  GS splices the vector lists to $\boldsymbol{w}_{i+1}$
end for
return $\boldsymbol{w}_{I_{agg}+1}$

For LBP, the GS weightedly averages them to ensure the correction of local bias and the generalization of model. In addition, the GS marks these LBP and adds their indexes to the parameter transmitting list, commanding more ENs to additionally upload these parameter vectors in next aggregation than just the ENs in their turn for transmitting these vectors. This allows the GS to enhance the aggregation of these vectors in the next round.

- Large Update Parameters (LUP). LUP are the parameter vectors significantly updating compared to the previous global model but less different across local models, i.e., $\sigma_b(\boldsymbol{v}_i^a) \leq \xi_b \wedge \sigma_u(\boldsymbol{v}_i^a) > \xi_u$. This indicates that the directions of LUP on local models are consistent, so for LUP, the GS requires locally updated vectors for direct model splicing, but have no need to collect more LUP from other ENs for average aggregation.
- Small Update Parameters (SUP). SUP are the parameter vectors that are not significantly updated compared to the previous global model and differs little across local models, i.e., $\sigma_b(\boldsymbol{v}_i^a) \leq \xi_b \wedge \sigma_u(\boldsymbol{v}_i^a) \leq \xi_u$. Therefore, it is unnecessary to upload SUP when aggregating models. SUP can be detected by the ENs themselves. If one SUP is found in the list of uploading parameter vectors, a very short placeholder is used to replace it instead, telling the GS that this vector is hardly updated. For example, in Fig. 4 vector 3 and vector 6 of EN 1 and vector 7 of EN 2 are SUP, which are replaced with placeholders before the upload to further reduce the transmission cost.

It is noted that the types of parameter vectors are not constant and will change with the model training. The GS and ENs should periodically update the types of parameter vectors so that the transmission strategy can be adapted to the training situation. Moreover, the LBP is detected by GS based on the parameter vectors collected from ENs, while SUP is determined by each EN itself. It is likely that the vector $\boldsymbol{v}_i^a$ in the local model of n_a is updated to a small extent and defined as SUP, but the vector at the same position $\boldsymbol{v}_i^b$ in the local model of n_b is updated significantly and is of type LUP or LBP. Finally, Algorithm 1 illustrates the main steps of SPT protocol.

4.3 Training Filling Model (TFM) Mechanism

Because the EN data is non-IID, without the data distribution of each EN, GS may select ENs all lacking training samples of a specific class when the aggregating model, leading a drop in the accuracy of global model. Therefore, to mitigate the impact of non-IID data on model performance, the nodes in FL need to adjust the training data [13].

On the left side of Fig. 5, the data adjustment approach used in prior research is illustrated. To begin, each EN provides the GS its data distribution information. The GS then compares the received information to its own test data distribution and informs ENs on data adjustment strategies. The strategies can identify major and minor classes, i.e., classes with excessive and undersized samples, and direct ENs to

downsample major classes and perform data augmentation on minor classes [13]. This guarantees that the distribution of the local training data is consistent with the test data of GS. This approach, however, never considers the data privacy security of ENs or its applicability in realistic scenarios. ENs should first submit the original data distribution to the GS to update the training data, however the honest-and-curious GS may infer some personal traits of ENs based on the data distribution. Obviously, the private information of ENs is exposed. Second, in practice, some dishonest ENs may be reluctant to make changes to their private data and instead continue training models with the original data after obtaining the adjustment strategies, which is hard to detect in time for GS. As a result, such dishonest ENs may have an impact quietly on the global model performance in the long term.

To preserve the data privacy of ENs while enhancing the robustness of data adjustment approach, the scheme of training filling model proposed is called Training Filling Model (TFM), as illustrated on the right side of Fig. 5. The EN group uploads local model fragments to the GS first, followed by model aggregation by the GS. The global model is obtained after aggregation, and GS tests it on its dataset. It is worth noting that the test is run by data label classes, which not only tests the global accuracy, but also shows the performance of the global model on each class statistically. It is reasonable to speculate that if samples of a class are substantially missing from the training data, then the trained model will perform poorly on the test data of this class. As a result, by comparing the test accuracy of the global model on each class, the classes with accuracy below the mean are identified as minor classes. Then the GS samples data of minor classes from its training set to form a filling dataset, which is then used to train a filling model. However, there are some GSs that are not equipped with enough training data on them, which is the reason why they have to release federated learning tasks. For this situation, we provide a targeted recruitment strategy. GS will only select part of the EN group as trainers, while other ENs are also available for rich training data. Therefore, GS can recruit them as targeted trainers for training the filling model. Targeted trainers need to train with data meeting GS requirements, i.e., data from minor classes detected by GS through testing, and have no need to use all the training data as ordinary trainers. In this way, GS can divide the required filling dataset into multiple portions and release the tasks to avaliable ENs, and declare the reward for targeted training. Then the eligible ENs can accept the tasks and sample data from their own training datasets and add certain noise (data from other classes) to them, thus migrating the model overfitting and shielding their data distribution from the outside. After GS receives these trained filling models, it first applies its own test data to validate the effect of the models. Once the effect of the filling models is verified, the GS can aggregate them with the global model. By aggregating the filling model and global model, the data gap in ENs can be bridged. In addition, if the data gap in a given class is too wide for the GS and targeted ENs to fill with all of their local data, they can utilize data augmentation techniques to transform the current data and generate new training examples.

As indicated in Equation (12), We define $acc_g^i(c)$ as the test accuracy of the aggregation model in the i-th round for class c on the global test set, and ρ_c^i as the filling ratio of class c in the i-th aggregation round. Equation (13) illustrates the

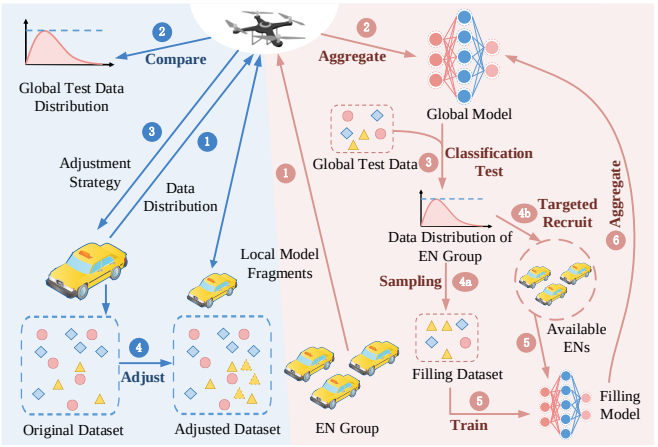


Fig. 5 The comparison between the filling model method and the data adjustment method in previous work

calculation of the filling data amount of class c, where F_{unit}^i is the filling data unit size defined by the GS. When the filling model is trained, the GS performs a weighted average of the global model and the filling model to obtain the final global model $\boldsymbol{w}_{i+1}$.

$$\rho_c^i = max\left(0, \frac{1}{C}\sum_{j=1}^{C} acc_g^i(j) - acc_g^i(c)\right). \tag{12}$$

$$F_c^i = \rho_c^i F_{unit}^i. \tag{13}$$

$$\boldsymbol{w}_{i+1} = \frac{\sum_{k=1}^{|\boldsymbol{N}_i|} |\mathbb{D}_k| \boldsymbol{w}_i^{agg} + \sum_{c=1}^{C} F_c^i \boldsymbol{w}_i^{fill}}{\sum_{k=1}^{|\boldsymbol{N}_i|} |\mathbb{D}_k| + \sum_{c=1}^{C} F_c^i}. \tag{14}$$

This approach of training filling model offers the following advantages over the data adjustment approach in prior work. First, the initiative of data gap detection and filling is moved from ENs to the GS, which prevents the misleading data adjustment behaviors of dishonest ENs and enhances the practicality and robustness of approach. Second, instead of directly touching the data distribution of ENs, the GS infers the overall data distribution of EN group from the aggregated global model and fills the data gaps by model aggregation, which is more thorough and ensures the data privacy of ENs. Algorithm 2 depicts the main steps of TFM mechanism.

4.4 Lightweight Defense Scheme for Multi-Attackers

Although FEL does not exchange the source data of ENs, the model still retains the relative information, as stated in the Problem Statement. By stealing the complete local model, inversion attack etc. can deduce the data distribution of users [17]. In FEL, GS, ENs, and external adversaries may all constitute a danger to data privacy [16]. Following that, we will develop the appropriate lightweight defenses for various attackers.

- GS. The concept of local model splitting and rotated transmission makes the GS get segments of local models and the complete local models are not available, which ensures data security to some extent. In Section 4.2, the parameter redundancy m_r^i controls the transmission amount m_a^i for each EN by Equation (9). However, when m_r^i is excessively large, ENs need to transmit the complete local models, and the parameter transmission strategy will degenerate to basic FEL. Therefore, to avoid GS from acquiring too large local model segments, we need to constrain m_a^i, i.e., $m_a^i \leq \left(1 - \gamma_g^i\right)\frac{\sum_{l=1}^{L} \mathcal{N}_v^l}{b}$,

Algorithm 2 TFM Mechanism

Input: F_{unit}^{i}
Output: $\boldsymbol{w}_{i+1}$
GS receives the vector lists and aggregates → $\boldsymbol{w}_i^{agg}$
for $c = 1 \ldots C$:
  GS tests the accuracy of $\boldsymbol{w}_i^{agg}$ on class c
  GS appends $acc_g^i(c)$ to $\boldsymbol{Acc}_g^i$
end for
GS compute the mean of $\boldsymbol{Acc}_g^i \rightarrow acc_{mean}^i$
for $c = 1 \ldots C$:
  GS computes F_c^i by Equation (12) and (13)
end for
if GS has enough training data:
  GS samples and forms the filling dataset $\mathbb{D}_{fill}$
  GS trains filling model $\boldsymbol{w}_i^{fill}$ using $\mathbb{D}_{fill}$
end if
else:
  GS recruits targeted ENs $\boldsymbol{N}_i^{tar}$
  $\boldsymbol{N}_i^{tar}$ train $\boldsymbol{W}_i^{fill} = \left\{\boldsymbol{w}_{i,1}^{fill}, \ldots, \boldsymbol{w}_{i,|\boldsymbol{N}_i^{tar}|}^{fill}\right\}$
end else
GS computes $\boldsymbol{w}_{i+1}$ by Equation (14)
return $\boldsymbol{w}_{i+1}$

where γ_g^i is the defense coefficient against GS, which can be determined by GS and EN group after game negotiation. The larger γ_g^i is, the smaller m_a^i is and the better the defense effect is. From this, we can obtain m_r^i after subjecting to the defense constraint as:

$$\frac{|\mathbb{D}_a|}{\sum_{k=1}^{|\boldsymbol{N}_i|}|\mathbb{D}_k|}\left(\frac{\sum_{l=1}^{L}\mathcal{N}_v^l}{b}+|\boldsymbol{N}_i|m_r^i\right)<\left(1-\gamma_g^i\right)\frac{\sum_{l=1}^{L}\mathcal{N}_v^l}{b},$$

i.e., $m_r^i < \frac{\left[\left(1-\gamma_g^i\right)\sum_{l=1}^{L}\mathcal{N}_v^l\right]\sum_{k=1}^{|\boldsymbol{N}_i|}|\mathbb{D}_k| - \sum_{l=1}^{L}\mathcal{N}_v^l|\mathbb{D}_a|}{b|\mathbb{D}_a||\boldsymbol{N}_i|}$.

- ENs. In basic FEL, the curious EN may infer the data distribution of other ENs by computing the difference between the global model and its own local model, i.e., the inverse calculation of FedAvg, especially when the EN number is small [16]. However, in our LightFed, MSS has changed the model aggregation method to "averaging-splicing", so the global model is no longer a simple weighted average of all local models, and the above inference method fails. Secondly, in Section 4.3, the training and aggregation of the filling model disturbs the original data distribution of ENs, thus resisting the prying of malicious ENs into the data privacy of other ENs. Due to TFM, the GS has removed the apparent data imbalance of EN data by aggregating filling model before issuing the new global model. Meanwhile, the original data distribution features of ENs are also hidden. Therefore, when a malicious EN gets the new global model, it can no longer accurately infer the data distribution of other ENs by inverse FedAvg.
- External Adversaries. Malicious external attackers may eavesdrop on the channel during the model training process to get the model segments uploaded by ENs. According to the law of parameter rotation transmission, the external adversary can retain the state of eavesdropping for multiple aggregations, and collects distinct local model segments of ENs and splices them into a complete model, thus spying on the user privacy. Moreover, the external adversary either listens to the downlink, and the steals and directly applies the global model issued by GS. Therefore, for malicious external adversaries, we must achieve a higher defense goal, namely, to prevent external adversary from getting any valid model segment.

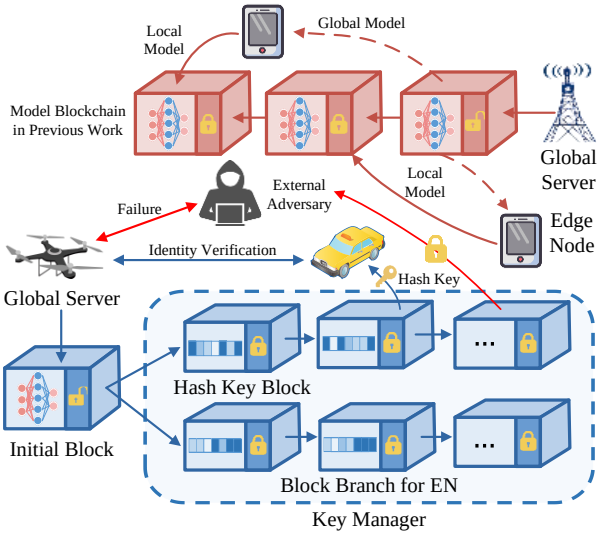


Fig. 6 The comparison of blockchain-powered parameter confusion transmission mechanism and model blockchain in previous work.

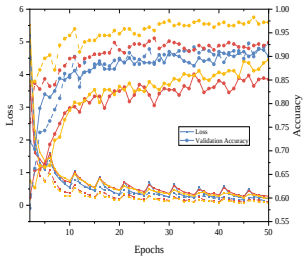
The blockchain-powered parameter confusion transmission mechanism is proposed for defending external adversary, as shown in Fig. 6. Before attending the federal training, GS applies to the authorities for opening a DAG blockchain and sends a random initial global model to the initial block. Any user can access the initial block because it has no specified permissions. Second, the ENs chosen by GS read and validate the model in the initial block before the local training. The blockchain then breaks into several branches, each of which corresponds to one EN. The hash key block on the branch is used to hold the hash key function published by GS, which is unique to each EN. The job of hash key function $\mathcal{H}$ is to mess up the sequence of the parameter vectors sent on the link between ENs and GS, i.e., parameter confusion transmission, thus achieving the encrypted communication. For instance, for EN n_a, it will transform the uploading vector list $\mathbb{V}_a^i$, i.e., the random mapping of $\mathbb{V}_a^i$ to itself, to obtain $\mathcal{H}_a\left(\mathbb{V}_a^i\right)$. The GS can inversely infer to recover $\mathbb{V}_a^i$ based on $\mathcal{H}_a$ after receiving the confusing data stream, while other users without $\mathcal{H}_a$ cannot get the correct order of the model fragments even if they eavesdrop on $\mathcal{H}_a\left(\mathbb{V}_a^i\right)$. To ensure the security of the hash key, the permission of each hash key block is set by GS when publishing it, and only authenticated ENs can get the private key from GS and open the corresponding hash key blocks to read their own hash keys, as shown in Fig. 6.

5 Performance Analysis

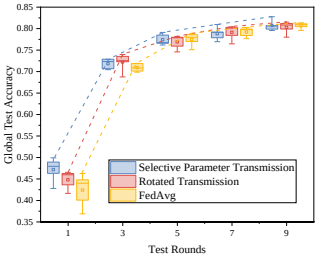
5.1 Experiment Setup

In this section, we implement the algorithms above using the Keras framework and verify the performance of the novel FEL system on the generic datasets MNIST, Fashion MNIST and CIFAR-10, including model convergence, model accuracy, communication efficiency and data security.

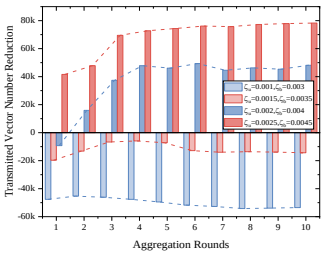
To simulate the non-IID of the local data of ENs, we apply a truncated Gaussian distribution on the datasets after sorting by labels, sampling with different means and standard devia-



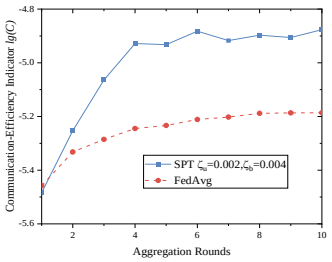
(a) The loss and validation accuracy of ENs applying LightFed



(b) The comparison of SPT and FedAvg on the test accuracy of global model



(c) The transmitted vector number reduction under different thresholds



(d) The comparison of SPT and FedAvg on the communication efficiency indicator $lg(\mathcal{C})$

Fig. 7 The performance on model accuracy and communication efficiency [7]

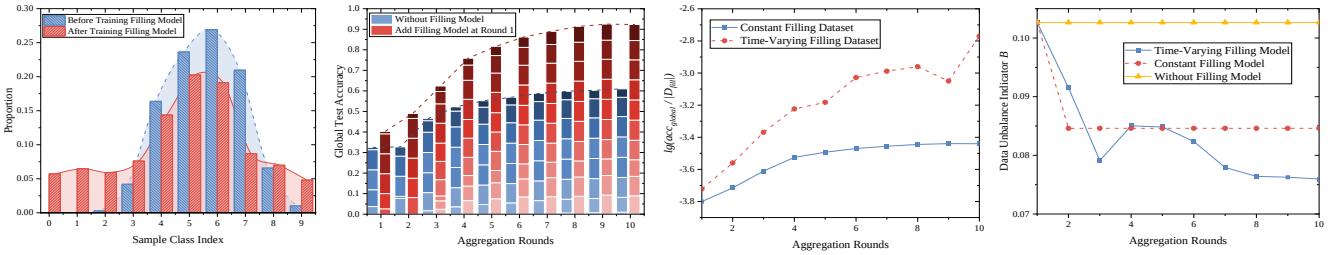
Table 2 The performance comparison between SPT and FedAvg on various model structures, datasets and EN group scales

	MNIST				Fashion MNIST				CIFAR-10			
	SPT		FedAvg		SPT		FedAvg		SPT		FedAvg	
	$\overline{acc_{test}}$	acc_g	$\overline{acc_{test}}$	acc_g	$\overline{acc_{test}}$	acc_g	$\overline{acc_{test}}$	acc_g	$\overline{acc_{test}}$	acc_g	$\overline{acc_{test}}$	acc_g
FCN	51.43%	81.78%	46.68%	82.54%	35.58%	57.56%	37.92%	58.28%	-	-	-	-
CNN	59.58%	94.55%	62.66%	94.98%	47.18%	**80.77%**	47.11%	76.03%	43.13%	51.16%	42.73%	52.38%
AlexNet	53.19%	**94.88%**	55.41%	93.11%	39.84%	**77.10%**	43.66%	75.80%	39.44%	**52.38%**	40.12%	50.42%
DenseNet	54.77%	93.44%	53.52%	94.95%	39.48%	73.18%	40.55%	74.44%	40.66%	52.17%	41.49%	52.56%
ResNet	47.51%	**79.90%**	46.54%	75.25%	36.13%	**71.70%**	37.24%	70.91%	37.56%	43.83%	37.00%	44.95%
	EN Number= 5				EN Number= 7				EN Number= 9			
	SPT		FedAvg		SPT		FedAvg		SPT		FedAvg	
	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g	$\overline{\lvert \boldsymbol{W} \rvert}$	acc_g
FCN	69.16M	81.78%	219.36M	82.54%	101.41M	82.26%	307.11M	85.97%	173.23M	82.72%	394.86M	86.64%
CNN	25.68M	94.55%	92.34M	94.98%	49.58M	**94.94%**	129.28M	94.20%	90.22M	**94.92%**	166.21M	94.84%
AlexNet	33.30M	**94.88%**	73.69M	93.11%	53.04M	94.95%	103.17M	95.08%	82.36M	95.98%	132.64M	96.39%
DenseNet	26.62M	93.44%	85.20M	94.95%	48.66M	93.72%	119.28M	94.19%	93.44M	94.02%	153.37M	95.14%
ResNet	63.63M	**79.90%**	220.86M	75.25%	104.52M	**87.87%**	309.21M	82.18%	139.02M	**88.01%**	397.55M	84.79%

tions to form the local dataset for each EN. Next, we apply LightFed to various network structures such as basic CNN, AlexNet and DenseNet for training image recognition tasks.

5.2 Performance on Accuracy and Communication

Fig. 7 depicts the performance of LightFed on model accuracy and communication. We simulated 6 ENs with different local data volume and data distribution for 10 rounds of model aggregation. Fig. 7(a) illustrates the loss and validation accuracy of each EN, which both all converge after 50 local epochs. Second, we discover that the training loss of ENs improves significantly in the first epoch after the model aggregation, but the loss decreases to a lower value as the local update progresses. This is because the GS takes fragments of multiple local models and splices them into a global model, and the data features of different ENs are fused. As a result, the newly aggregated global model is less adaptive to local training data than the previous local model. The global test accuracy of SPT, rotated transmission and FedAvg are compared in Fig. 7(b). SPT outperforms the rotated transmission because it selectively transmits parameters, marks parameters with high update strength and focuses on their aggregation: SPT converges faster in the early stage of model training and has same peak accuracy with the other two schemes, resulting in efficient model aggregation. Fig. 7(c) depicts the transmission reduction under various thresholds applying SPT. When ξ_b and ξ_u are larger, ENs detect more SUP unnecessary to be transmitted as the model aggregation rounds increase, and the number of LBP continues to decrease, resulting in a significant reduction in the parameter number uploaded by ENs, lowering the communication overhead between ENs and GS. When ξ_b and ξ_u are small, however, the number of SUP decreases while additional LBP to be transmitted increases, which increases parameter aggregation redundancy, and the total number of parameterstransmitted even increases significantly compared to the original scheme. Therefore, in practice, determining the proper ξ_b and ξ_u based on the gradient of the model parameters is critical for the reduction of redundant parameters transmitted without compromising the model test accuracy. Fig. 7(d) compares the performance of SPT and FedAvg on the communication efficiency index $\mathcal{C}$. We convert $\mathcal{C}$ to $\lg(\mathcal{C})$ for comparison since the magnitude of $\mathcal{C}$ is too tiny. Compared with FedAvg, SPT significantly improves on the metric $\lg(\mathcal{C})$ due to the significant transmission reduction and the undiminished model accuracy. To further demonstrate the communication efficiency and scalability of LightFed, we applied models such as AlexNet, DenseNet to implement federated training on MNIST, Fashaion MNIST and CIFAR-10 to obtain Table 2. From Table 2, first we see that the global model accuracy acc_g has a larger improvement than $\overline{acc_{test}}$, the average accuracy of local models on the global test set. Second, compared with FedAvg, LightFed significantly reduces the parameter transfer $\overline{|\boldsymbol{W}|}$ without compromising the model effectiveness, and is able to adapt to changes in EN group scale, demonstrating that LightFed can be applied flexibly to different model structures and training tasks.



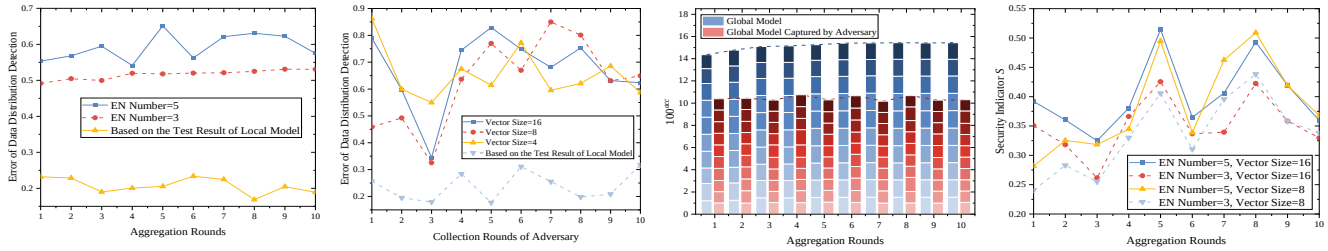
(a) The proportion of samples in various classes before and after training filling model

(b) The comparison on the global test accuracy before and after training filling model

(c) The comparison on the extra training cost of constant and time-varying filling dataset

(d) The comparison on the data balance indicator $\mathcal{B}$ of constant and time-varying filling model

Fig. 8 The performance of filling model mechanism



(a) The error of data distribution detection of GS

(b) The error of data distribution detection of external adversary

(c) The comparison on the global test accuracy of global model and the model captured by adversary

(d) The security indicator $\mathcal{S}$ under different EN number and vector size

Fig. 9 The performance of data security protection mechanism

5.3 Performance of TFM

To demonstrate the effectiveness of the filling model for filling the data gap, we intentionally scaled down the EN number to 3, so that their total local training data distribution as shown in the blue curve in Fig. 8(a) with obvious data gaps: class 0-2 and class 9 have almost no training samples. However, after the GS tests the accuracy of the global model on all classes of data, as well as sampling from its own data to get the filling dataset based on the test accuracy, the superimposed data distribution of GS and ENs is shown as the red curve in Fig. 8(a). Comparing the two data distribution curves, the data distribution is more balanced after adding the filling dataset, and the previous data gaps are filled by the GS data. Then, Fig. 8(b) illustrates the boosting effect of filling model on the global test accuracy. Each bar is formed by stacking the test accuracy on different classes, and the tiny bars with different shades indicate different classes in the dataset. Before aggregating the filling model, there are almost no light blue tiny bars at the bottom of the blue bar and the height of dark blue tiny bar at the top is small, indicating that the model without filling model has poor performance on class 0-2 and class 9 due to the data gaps. Therefore, the overall test accuracy is lower than 0.6. However, after aggregating the filling model, it is obvious that the model has a good effect on the classes with terrible test accuracy before, and the global test accuracy continues to improve, with the peak exceeding 0.9, even better than that of Fig. 7(b). The comparison above fully illustrates the superiority of the filling model in improving the model accuracy.

The filling dataset for training the filling model above is sampled by GS during the first model aggregation round, and will not be changed during the subsequent training process. However, as the local model is continuously updated, the test accuracy of the global model after each aggregation varies, so the filling dataset capacity and the proportion of each class should be adjusted. Therefore, we propose a time-varying filling dataset strategy based on the constant filling dataset, i.e., the GS resamples the filling data according to the latest test results after each model aggregation. The test accuracy of all classes of data continues to increase with the model training progress and the filling model being fused, according to simulated experiments. As a result, in the latter stage of training, the GS unnecessarily supplies the large-capacity filling dataset as the earlier stage, which helps GS train the filling model with less overhead. Fig. 8(c) shows the efficiency of the constant and time-varying filling model, denoted by $\lg\left(\frac{acc_{global}}{|\mathbb{D}_{fill}|}\right)$, where $|\mathbb{D}_{fill}|$ is the capacity of the filling dataset. The efficiency of the time-varying filling model improves considerably with aggregation rounds and surpasses that of the constant filling model. Finally, Fig. 8(d) compares the performance of the three schemes on the data imbalance metric $\mathcal{B}$. The constant filling model only adjusts the data distribution in the first model aggregation, and then the training data remains unchanged, while the time-varying filling model adjusts adaptively based on the latest test results of the model, achieving better balanced training data and lower extra training cost.

5.4 Performance of Model Protection

The error of the EN data distribution detected by GS in LightFed is shown in Fig. 9(a). The yellow curve indicates the detection error of the data distribution of EN target inferred from the local test results, which is always lower than 0.25, while the detection error of GS according to the collected model fragments is higher than 0.5, and more EN participants bring the larger error. This indicates that discontinuous model fragments can hide the local data features of EN target to a certain extent and effectively resist the inversion attack of GS. Fig. 9(b) shows the detection error of the external adversary for

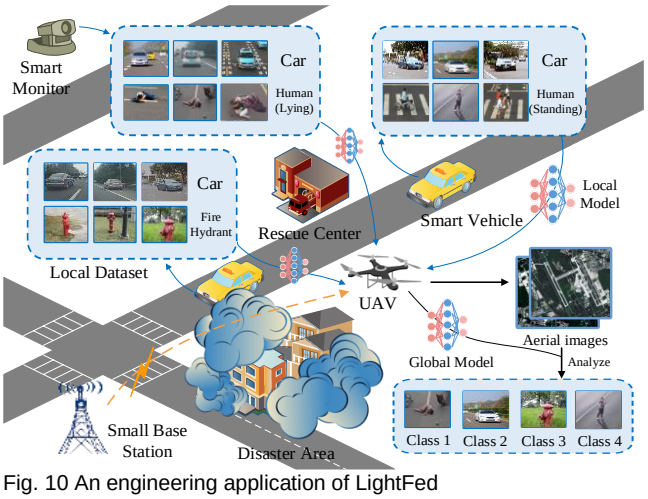


Fig. 10 An engineering application of LightFed

the data distribution of EN target under different vector sizes. Similarly, the detection accuracy of external adversary is significantly reduced and the detection results are very unstable compared to that based on the local test results of the EN target. This is due to the fact that the external adversary can only eavesdrop on the model fragments encrypted by random hash keys and is unable to recover them. The failure of model fragments scrambles the detection of external adversary. Fig. 9(c) shows the availability of the encrypted global model captured by external adversary. We use 100^{acc} as a comparison metric and see that the accuracy of global model captured by adversary is significantly lower, indicating that under blockchain-powered confusion transmission, the external adversary is unable to not only peek into the local data distribution of ENs but also effectively apply the trained global model. Finally, Fig. 9(d) shows the trend of the security indicator δ. As the EN number increases and the vector size reduces, δ is gradually improved. More EN participants reduce the data exposure of an individual EN, and smaller vector size increases the difficulty of recovering encrypted model fragments, which are both conductive to the defense against attackers.

5.5 Engineering Applications

Fig. 10 illustrates an engineering application of LightFed, called federated rescue system. Due to the high-risk and urgency of rescue tasks, UAV is of great advantage as a tiny and agile detection tool when rescuers cannot easily access the disaster area [36]. Applying LightFed, it is promising to utilize the free resource of smart edge nodes around the disaster area to achieve joint model training. The UAV first determines important sample classes, such as targets to be rescued (cars, humans, etc.) and available rescue resources (fire hydrants, etc.), and then establishes connections with surrounding edge nodes (smart vehicles, smart cameras, etc.). The recorders of vehicles and cameras store many images of vehicles, pedestrians in postures: standing normally, lying down due to traffic accidents, etc. According to the workflow of LightFed described above, smart vehicles and cameras can update local models and upload them using lightweight encryption and protocol, enabling the UAV to quickly aggregate and improve the accuracy of global model. By splitting aerial images and applying the global model, the UAV can detect and locate important targets in the captured area, and then send the detection information to the rescue center for guidance.

6 Conclusion and Future Work

This paper proposes a novel FEL system LightFed, aiming to achieve lightweight and secure AI model transmission and aggregation. First, the aggregation method MSS based on model splitting and the SPT scheme are introduced to reduce the data amount in the uplink of system. Second, TFM mechanism is designed to mitigate the local data imbalance while ensuring the data privacy of edge nodes. In addition, we propose a blockchain-powered confusion transmission strategy against multi-attackers to resist inversion attacks and secure the model with low extra cost. For the future in this paper, we plan to focus on the resource wastage caused by unbalanced computing power of ENs, and design aggregation strategies to further improve the system efficiency.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (No. 62072475, No. 61772554) and Independent Exploration and Innovation Project for Graduate Students of Central South University (2021zzts0750).

References

[1] Z. Ning et al., "Distributed and Dynamic Service Placement in Pervasive Edge Computing Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1277-1292, 2021.

[2] J. Yan, S. Bi, Y. J. Zhang and M. Tao, "Optimal Task Offloading and Resource Allocation in Mobile-Edge Computing with Inter-User Task Dependency," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 235-250, 2020.

[3] L. Huang, S. Bi and Y. J. A. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581-2593, 2020.

[4] T. Huang, W. Lin, W. Wu, L. He, K. Li and A. Y. Zomaya, "An Efficiency-Boosting Client Selection Scheme for Federated Learning With Fairness Guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1552-1564, 2021.

[5] X. Lin, J. Wu, J. Li, X. Zheng and G. Li, "Friend-as-Learner: Socially-Driven Trustworthy and Efficient Wireless Federated Edge Learning," *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2021.3074816.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Trans. Intell. Syst. Technol*, vol. 10, no. 2, 2019.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, pp. 1273-1282, 2017.

[8] L. U. Khan, W. Saad, Z. Han, E. Hossain and C. S. Hong, "Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1759-1799, 2021.

[9] X. Li, H. Chen, X. Qi, Q. Dou, C. -W. Fu and P. -A. Heng, "H-Dense UNet: Hybrid Densely Connected UNet for Liver and Tumor Segmentation From CT Volumes," *IEEE Transactions on Medical Imaging*, vol. 37, no. 12, pp. 2663-2674, 2018.

[10] W. Wu, L. He, W. Lin, R. Mao, "Accelerating Federated Learning Over Reliability-Agnostic Clients in Mobile Edge Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1539-1551, 2020

[11] S. Wang et al., "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205-1221, 2019.

[12] X. Wu, X. Yao and C. -L. Wang, "FedSCR: Structure-Based Communication Reduction for Federated Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1565-1577, 2021.

[13] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan and L. Liang, "Self-Balancing Federated Learning With Global Imbalanced Data in Mobile Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59-71, 2021.
[14] Q. Wu, X. Chen, Z. Zhou and J. Zhang, "FedHome: Cloud-Edge based Personalized Federated Learning for In-Home Health Monitoring," *IEEE Transactions on Mobile Computing*, doi: 10.1109/TMC.2020.3045266.
[15] Y. Qu et al., "Decentralized Privacy Using Blockchain-Enabled Federated Learning in Fog Computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5171-5183, 2020.
[16] B. Hitaj, G. Ateniese, and F. P-Cruz, "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Association for Computing Machinery*, New York, NY, USA, pp. 603–618, 2017.
[17] L. Melis, C. Song, E. De Cristofaro and V. Shmatikov, "Exploiting Unintended Feature Leakage in Collaborative Learning," *IEEE Symposium on Security and Privacy*, pp. 691-706, 2019.
[18] Y. Lu, X. Huang, Y. Dai, S. Maharjan and Y. Zhang, "Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134-2143, 2020.
[19] K. Wei et al., "Federated Learning With Differential Privacy: Algorithms and Performance Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454-3469, 2020.
[20] G. Wang, et al, "Deep learning for tomographic image reconstruction," *Nat Mach Intell*, vol. 2, pp. 737–748, 2020.
[21] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, C. Li, "Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey," *ACM Trans. Intell. Syst. Technol*, vol. 11, no. 3, 2020.
[22] Z. Mushtaq, S. Su, "Environmental sound classification using a regularized deep convolutional neural network with data augmentation," *Applied Acoustics*, vol. 167, 2020.
[23] Q. Zhang, C. Zhou, Y. Tian, N. Xiong, Y. Qin and B. Hu, "A Fuzzy Probability Bayesian Network Approach for Dynamic Cybersecurity Risk Assessment in Industrial Control Systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2497-2506, 2018.
[24] Y. Qu and N. Xiong, "RFH: A Resilient, Fault-Tolerant and High-Efficient Replication Algorithm for Distributed Cloud Storage," *2012 41st International Conference on Parallel Processing*, pp. 520-529, 2012.
[25] E. P. Xing et al., "Petuum: A New Platform for Distributed Machine Learning on Big Data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49-67, 2015.
[26] Y. Low, et al, "GraphLab: A New Framework For Parallel Machine Learning", *arXiv preprint*, arXiv: 1408.2041, 2014.
[27] M. Li et al, "Scaling distributed machine learning with the parameter server," *Proc. 11th USENIX Symp. Operating Syst. Des. Implementation*, pp. 583–598, 2014.
[28] A. Hard, et al, "Federated Learning for Mobile Keyboard Prediction," *arXiv preprint*, arXiv: 1811.03604, 2018.
[29] M. Hao, H. Li, G. Xu, Z. Liu, and Z. Chen, "Privacy-aware and Resource-saving Collaborative Learning for Healthcare in Cloud Computing," *Proceedings of the International Conference on Communications (ICC), Dublin, Ireland*, pp. 1–6, 2020.
[30] B. Liu, L. Wang, M. Liu, and C.-Z. Xu, "Federated Imitation Learning: A Novel Framework for Cloud Robotic Systems With Heterogeneous Sensor Data," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3509–3516, 2020.
[31] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato and S. Zhang, "Privacy-Preserving Traffic Flow Prediction: A Federated Learning Approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751-7763, 2020.
[32] J. Konečný, et al, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint*, arXiv:1610.05492, 2016.
[33] V. Smith, C. K. Chiang, M. Sanjabi, A. Talwalkar, "Federated multi-task learning," *arXiv preprint*, arXiv:1705.10467, 2017.
[34] Y. Lu, X. Huang, K. Zhang, S. Maharjan and Y. Zhang, "Blockchain Empowered Asynchronous Federated Learning for Secure Data Sharing in Internet of Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298-4311, 2020.
[35] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, 2019.
[36] W. Chen, Z. Su, Q. Xu, T. H. Luan, R. Li, "VFC-Based Cooperative UAV Computation Task Offloading for Post-disaster Rescue," *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Toronto, ON, Canada, pp. 228-236, 2020.
[37] X. Niu, C. Y. Suen, "A novel hybrid CNN–SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318-1325, 2012.
[38] S. Liu, J. Yu, C. Hu, M. Li, Y. Wang, "Traceable Multiauthority Attribute-Based Encryption with Outsourced Decryption and Hidden Policy for CIoT," *Wireless Communications and Mobile Computing*, pp. 1-16, 2021.

Jialin Guo is currently a student at the School of Computer Science and Engineering, Central South University, China. His research interests include edge computing and wireless sensor networks.
E-mail: guojialin@csu.edu.cn.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, network trust and security, distributed algorithms, and cloud computing.
E-mail: jiewu@temple.edu.

Anfeng Liu is a Professor with School of Computer Science and Engineering of Central South University, China. His major research interest is wireless sensor networks, Internet of Things, information security, edge computing and crowdsourcing.
E-mail: afengliu@mail.csu.edu.cn.

Neal N. Xiong is current an Associate Professor at Department of Computer Science and Mathematics, Sul Ross State University, Alpine, TX 79830, USA. He received his both PhD degrees in Wuhan University, and Japan Advanced Institute of Science and Technology, respectively. Before he attended Northeastern State University, he worked in Georgia State University, and Colorado Technical University (full professor about 5 years), Northeastern State University about 14 years. His research interests include Cloud Computing, Security and Dependability.
E-mail: xiongnaixue@gmail.com.