Lightweight Federated Learning for Terminal-Edge-Cloud with Two-Step Privacy Protection in Consumer Electronics

Chao Chen, Xiaolong Xu, Yifan Zhang, Muhammad Bilal, Haolong Xiang*

***Abstract*— With the rapid development of Internet of Things (IoT) technology, consumer electronics increasingly integrate edge computing to improve real-time decision-making in applications such as smart cities and intelligent transportation. The protection of user privacy has become a critical concern due to the vast amounts of sensitive data generated and processed in consumer electronics. Existing privacy-preserving methods primarily address two aspects: safeguarding sensitive user data during offloading from devices to edge servers, and protecting model parameters in cloud processing. However, these approaches often fail to comprehensively address privacy risks in terminal-edge-cloud collaboration, including challenges arising from data heterogeneity, risks of user privacy leakage, and parameter reverse inference attacks. To address these challenges, we propose a lightweight privacy-preserving federated learning algorithm with two-step differential privacy for consumer electronics, integrated into a terminal-edge-cloud architecture (TECDP). TECDP utilizes edge computing and differential privacy to reduce data transmission and perform local preprocessing and encryption, balancing privacy protection with data utility. Lightweight CNN models run on edge devices, while more complex models are deployed in the cloud for improved accuracy. We conducted extensive experiments on the MNIST and CIFAR datasets and evaluated the impact of varying privacy budgets and parameters on model performance. The results demonstrate that TECDP maintains high accuracy while reducing the risk of data leakage.**

***Index Terms*— Lightweight Artificial Intelligence (AI), Internet of Things (IoT), edge computing, federated learning, differential privacy, privacy protection.**

I. Introduction

In recent years, the integration of Artificial Intelligence (AI) into consumer electronics has accelerated with advancements in Machine Learning (ML) and the Internet of Things (IoT) [1]. Smart home appliances, wearable gadgets, and intelligent cameras are using AI more and more to make judgments in real time and give personalized services at the edge [2]. But it's still impossible to employ typical AI models on edge devices with few resources because they demand a lot of memory and processing power [3]. This has led to the creation of lightweight AI systems that use methods like model compression, pruning, and knowledge distillation to keep important features while still letting edge devices work well [4]. Lightweight AI solutions make the consumer electronics industry work better by letting edge devices process data in real time [5].

Data breaches [6], latency, and single points of failure [7] are some of the challenges that cloud-based AI processing faces. Federated Learning (FL) enables collaborative model training on edge devices while keeping raw data localized, thus mitigating some privacy risks [8]. FL still has problems, though, such as too much communication, data that isn't always the same, and the fact that enemies can attack it [9]. Edge computing solves these problems by moving computing tasks to the edge of the network. This makes cloud services work better and keeps data safer and more organized [10]. Edge computing can make consumer electronics applications safer and more private when used with FL [11]. However, sending all raw data from consumer devices to edge servers is also not safe because edge servers can potentially be attacked by untrusted third parties, which adds more privacy risks [12].

A lot of personal information, like biometric data and patterns of how individuals use their gadgets, is stored on consumer devices. This makes it hard to keep privacy safe in decentralized edge systems [13]. Existing privacy-preserving methods [14–17] primarily focus on data anonymization, encryption, and secure data-sharing protocols. Even with these steps, it is still hard to find a balance between data usefulness and privacy protection. When people share and use consumer data, it can put their privacy at jeopardy because it generally contains very private information [18].Traditional data protection methods frequently have a hard time finding the right balance between protecting privacy and using data efficiently. Using untrusted third-party service providers makes privacy leaks more likely [19].

This paper presents a lightweight privacy-preserving federated learning algorithm featuring two-step differential privacy for consumer electronics, incorporated within a terminal-edge-cloud architecture (TECDP), to mitigate the substantial privacy concerns related to consumer electronics data. This method is specifically designed for consumer electronics, with a focus on the application of lightweight AI. TECDP employs edge

Chao Chen and Yifan Zhang are with the School of Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: chaoc@nuist.edu.cn; yifanzhang20011231@gmail.com.

Xiaolong Xu and Haolong Xiang are with the School of Software and the Jiangsu Province Engineering Research Center of Advanced Computing and Intelligent Services, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: xlxu@nuist.edu.cn; hlxiang@nuist.edu.cn.

Muhammad Bilal is with the School of Computing and Communications, Lancaster University, United Kingdom. E-mail: m.bilal@ieee.org.

* Haolong Xiang is the corresponding author.

computing and differential privacy to speed up data transport and do local encryption and preprocessing. This finds a middle ground between keeping data private and making it helpful. Consumer devices like personal computers, sensors, and wearable electronics use local differential privacy (LDP). The level of privacy protection is set based on how private the data is and what the user wants. Lightweight Convolutional Neural Network (CNN) models work on edge devices, while more complicated models are put in the cloud to make them more accurate. In this way, not only can user privacy be protected during data transmission, but edge computing resources can also be effectively utilized, and the privacy of the system can be enhanced. The contributions of this paper can be concluded:

• We designed a new terminal-edge-cloud privacy protection architecture for consumer electronics that combines smart devices, edge servers, and cloud aggregation servers to protect user data in consumer electronics environments.

• We built a lightweight CNN model that uses depthwise separable convolutions and is specifically designed to work well on edge layers with limited resources. We make the model better for consumer electronics edge devices with low processing power by substituting ordinary convolutions with depthwise separable convolutions. This cuts down on the number of parameters and the amount of computation needed.

• Taking into account user preferences for privacy data, we installed a local differential privacy algorithm on consumer electronics devices. This lets consumers change the level of privacy protection to match their own needs. The updated model parameters are also protected on the edge servers.

• We analyzed the impact of privacy budget and parameters on model accuracy under different data distribution methods. The suggested solution protects privacy very well while still making sure that the model training is quick and correct, which lowers the chance of data leaks.

II. Related work

This section reviews existing literature in lightweight AI for IoT, edge computing in consumer electronics, federated learning, and integrative privacy protection methods.

A. Lightweight AI for IoT Applications

Recent improvements in lightweight AI frameworks have made it possible to use AI on IoT devices that don't have a lot of resources. This lets them digest data quickly and make decisions in real time. As IoT systems become more common in many fields, there is a greater need for AI solutions that work well within the restricted computing and energy resources of edge devices. These lightweight AI models try to make the computer work less hard without losing a lot of accuracy. Fouda et al. [20] developed a lightweight AI framework for the early identification of forest fires with drone-acquired data, integrating machine learning with convolutional neural networks (CNNs). Malibari [21] suggested a lightweight AI model enhanced by the Equilibrium optimizer (EO) for forecasting chronic diseases utilizing patient sensor data inside an IoT healthcare framework. To address traffic management in densely deployed IoT networks, Ateya et al. [22] developed a lightweight AI framework based on a CNN model to predict traffic, aiding in congestion avoidance.

Even though these methods indicate how lightweight AI could operate in different IoT circumstances by making devices smart, the fact that numerous consumer gadgets can handle sensitive data is a big privacy problem.

B. Edge Computing for Consumer Electronics

Edge computing can make smart cameras, wearables, and home automation systems work better by letting them handle data faster in the consumer electronics field. This cuts down on the amount of communication that needs to happen, which speeds up processing times and protects privacy by keeping sensitive data on the device, which lowers the risk of data breaches.

For instance, edge computing makes it possible to track video in real time and find vehicles in video surveillance and intelligent transportation systems. This solves problems like limited bandwidth and heavy processing needs. Wang et al. [23] introduced a multicamera multihypothesis tracking (MC-MHT) framework for real-time video tracking in edge-based Industrial Internet of Things (IIoT) systems. It reduces communication bottlenecks and enhances privacy by allocating the tracking work across edge cameras. Yang et al. [24] put forward the Edge-empowered Cooperative Multi-camera Sensing (ECoMS) system. The system uses edge computing to improve vehicle identification, tracking, and feature extraction in real time. This cuts down on the requirement for high-bandwidth data transfer and a lot of post-processing. Li et al. [25] also came up with Polly, a technology that lets you analyze footage from different cameras on edge devices. Polly lets cameras with overlapping fields of vision (FoVs) share inference results, which cuts down on unnecessary calculations and makes things run more smoothly. Even with these benefits, it is still hard to manage edge devices with limited resources and make sure that performance is always good in distributed networks.

C. Federated learning for Consumer Electronics

Federated Learning (FL) offers a distributed and collaborative AI model, enabling model training across multiple devices without necessitating the sharing of original data. This method of keeping data locally and updating models in a coordinated way is safer and more efficient for consumer electronics applications. For example, Song et al. [26] proposed FedBEVT, a federated learning-based method to enhance Bird's Eye View (BEV) perception in autonomous driving, addressing data heterogeneity. Similarly, Xie et al. [27] proposed a federated framework using semantic communication for real-time license plate recognition, aiming to improve traffic management while preserving privacy.

FL still has problems including high communication overhead, different data distribution, scalability, and the risk of privacy leaks through model modifications. TECDP builds upon FL's distributed paradigm by implementing a terminal-edge-cloud architecture specifically designed to mitigate these

issues. The addition of a two-step differential privacy mechanism at both the user device and the edge server gives better security to both raw data and model parameters. This makes it better for the sensitive and variable nature of consumer electronics data.

D. Integrative Methods for Privacy Protection

Recent research has begun exploring the synergistic effects of the aforementioned technologies to enhance data privacy and security. Fang et al. [28] put up a secure network coding (GS-SNC) technique based on Gold sequences. The goal was to improve safe and efficient data transmission in cloud-edge-terminal collaboration for artificial intelligence of things (AIoT).Wei et al. [29] created a lightweight, generic network intrusion detection system (NIDS) that works best on devices with limited resources to make IoT security better against botnet attacks. This technique uses 21 statistical features and then turns packet-length sequences into RGB images to find botnets using a lightweight CNN. Kim et al. [30] considered using the Gaussian mechanism to protect local differential privacy in federated learning models using stochastic gradient descent. By establishing suitable metrics for FL with LDP, they illustrated the balance between user privacy, global utility, and transmission rate.

Even though the approaches talked about are useful, many people don't see how federated learning, edge computing, and special privacy needs in consumer devices come together. Current research frequently delineates performance trade-offs or fails to comprehensively handle the extensive privacy requirements for data transfer and model parameter safeguarding in these contexts.

Our TECDP technique, on the other hand, is perfect for this case. It has a terminal-edge-cloud design and a two-step mechanism for differential privacy. This keeps both user data and model parameters safe while yet letting users set their own privacy settings. This method naturally deals with problems like data diversity and growth. TECDP is different from other solutions that only focus on things like secure transmission. It adds differential privacy straight into the federated learning process. This approach uses edge computing to lower the danger of data leaks and speed up processing. It is also very important to find a balance between protecting privacy and making data useful. Focusing solely on model accuracy without considering the security of sensitive data is inappropriate. In the same way, if the privacy protection is too high, it makes data less useful when sent to edge devices, which makes the system less successful. So, a good balance needs to be struck between protecting privacy and keeping the data useful for making good decisions at the edge.

III. Priliminaries

To effectively build our proposed architecture that strikes this crucial balance, we must first define the key technologies involved.

A. Edge Computing

Edge computing is a decentralized approach that processes data near its source, improving response times and reducing bandwidth usage. This technology is highly crucial for keeping data safe and private on consumer devices since it cuts down on the quantity of sensitive information that is sent [31]. Processing data locally on IoT devices and sensors lowers the chance of privacy breaches when data is sent. This method keeps personal information in a safe place, which solves privacy issues that come up with consumer gadgets.

B. Federated Learning

Federated learning (FL) is a decentralized machine learning method that lets several clients train a global model together while keeping their local training data private [32]. FL is different from standard centralized machine learning methods since it lets models learn from different and dispersed datasets without sending raw data. This makes the data safer and more private. The core of FL is the distributed optimization problem, typically represented as:

$$\min_{w} f(w) = \sum_{k=1}^{K} p_k f_k(w), \tag{1}$$

where w is the global model parameter, K is the number of participating clients, p_k is the weight of client k, usually the proportion of data at client k to the total data, $f_k(w)$ is the loss function at client k. The goal is to minimize the loss function.

Local updates and global aggregation are also quite important. In each iteration, each client k updates the model parameters w on local data, represented as:

$$w_k^{t+1} = w_k^t - \eta \nabla f_k(w_k^t), \tag{2}$$

where w_k^t is the model parameter at client k in the t-th round, η is the learning rate, $f_k(w_k^t)$ is the gradient at client k in the t-th round.

Then, the server aggregates the updates from all clients to generate the new global model:

$$w^{t+1} = \sum_{k=1}^{K} p_k w_k^{t+1}. \tag{3}$$

The aggregation server sends the new global model parameters to each client. Upon receiving the latest global model, each client uses it as the starting point for the subsequent training round. Through this distributed training method, federated learning can fully utilize the computational and data resources of each client while ensuring data privacy, thereby collaboratively training an efficient global model.

C. Differential Privacy

Differential privacy is a widely adopted privacy-preserving framework, it has been an actual standard for various data types in the privacy protection field over the past decade, owing to its robust privacy guarantees and minimal computational cost [33]. It introduces mathematical noise into the

data analysis process to ensure that the results of the analysis do not significantly depend on the presence or absence of any single data point. In other words, the impact of any single data point is minimized, thereby protecting individual privacy.

Differential privacy contains two important privacy protection definitions, namely ϵ-differential privacy and (ϵ, δ)-differential privacy [34].

The definition of ϵ-differential privacy is a mechanism M satisfies ϵ-differential privacy, if for any two adjacent data sets D and D' (D and D' only differ by one record), and any possible output $S \subseteq Range(M)$, both have:

$$Pr[M(D) \in S] \leq e^{\epsilon} Pr[M(D') \in S]. \tag{4}$$

The definition of (ϵ, δ)-differential privacy is a mechanism M satisfies (ϵ, δ)-differential privacy, and any possible outputs $S \subseteq Range(M)$ include:

$$Pr[M(D) \in S] \leq e^{\epsilon} Pr[M(D') \in S] + \delta. \tag{5}$$

Here, ϵ is the privacy budget, which controls the strictness of privacy protection: the smaller ϵ is, the stronger the privacy protection. δ allows for a small probability of failure, meaning the mechanism output may violate the ϵ-differential privacy condition under certain circumstances, but this probability is controlled by δ. A smaller δ provides stronger privacy protection. Together, ϵ and δ control the overall strictness of privacy protection.

Common noise mechanisms in differential privacy generally include the Exponential mechanism, Laplacian mechanism, Gaussian mechanism and Hybrid mechanism. The Laplacian mechanism is suitable for scenarios that meet ϵ-differential privacy requirements. The Gaussian mechanism is suitable for scenarios that meet the requirements of (ϵ, δ)-differential privacy. It is usually necessary to introduce a failure term with a small probability when protecting strong privacy. This paper uses a Gaussian mechanism to add noise to protect user data privacy.

In Federated Learning, differential privacy can be achieved by adding noise to the local model updates, represented as:

$$w_k^{t+1} = w_k^t - \eta \nabla f_k(w_k^t) + N(0, \sigma^2), \tag{6}$$

where $N(0, \sigma^2)$ is Gaussian noise with a mean of 0 and a variance of σ^2.

Table I summarizes the key symbols and their meanings in the aforementioned formulas. Understanding these symbols is essential for analyzing the underlying mechanisms of privacy-preserving techniques. The trade-off between privacy and data utility, together with the incorporation of DP into intricate machine learning algorithms, presents considerable problems.

TABLE I
SUMMARY OF KEY NOTATIONS

Math Symbols	Descriptions
w	Global Model Parameters
K	Number of Clients
p	Client Weights
$f(w)$	Client Loss Function
w_k^t	Model Parameters of Client k at Round t
η	Learning Rate
$\nabla f_k(w_k^t)$	Gradient
ϵ	Privacy Budget
δ	Failure Probability
D	Data Set

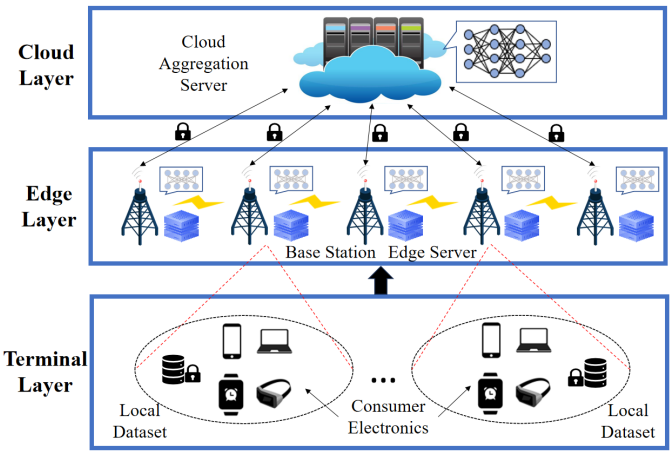


Fig. 1. A typical scenario of applying federated learning in consumer electronics environment.

IV. Privacy Protection Method Based on Terminal-Edge-Cloud Architecture

A. System Architecture

This paper proposes a system architecture focused on safeguarding user data privacy within a terminal-edge-cloud framework. There are three layers in the structure: the IoT device layer, the edge server layer, and the cloud aggregation server layer. Each layer has its own jobs and traits when it comes to processing data and protecting privacy. Hierarchical processing and user data protection make sure that the system is safe, reliable, and has low latency. Fig. 1 illustrates a typical terminal-edge-cloud architecture in a consumer electronics environment.

IoT Device Layer (Terminal Layer): This layer is made up of IoT devices including cameras, environmental sensors, and smart wearables that have raw user data. Before any data is transmitted, each IoT device applies LDP to the user's raw data. This is achieved by adding Gaussian noise directly on the device. Users can adjust a privacy level λ, influencing the noise scale, based on the sensitivity of their data. This crucial step ensures that the original sensitive data is perturbed at the source, minimizing privacy risks even before it leaves the user's device.

Edge Server Layer (Edge Layer): Edge servers are set up in base stations near IoT devices and have a lot of computational power. These servers put together the data they get and do the first steps of training or updating their own local models. The edge server adds another layer of differential privacy before sending model parameters to the cloud aggregation server. These local model parameters have Gaussian noise added to them. This process keeps the aggregated model information safe from any inference attacks. It also makes sure that contributions from single devices or small groups of

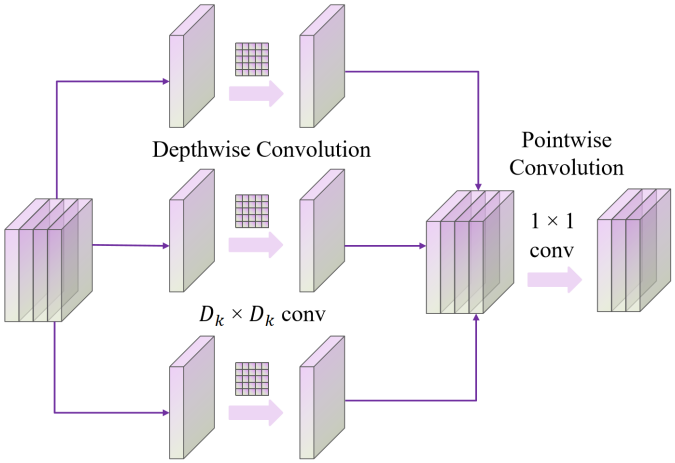


Fig. 2. Depthwise separable convolutions.

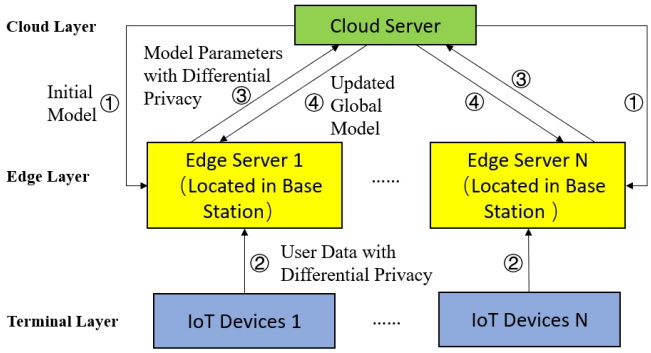


Fig. 3. The process of TECDP.

devices stay private while they are being sent to the cloud.

Cloud Aggregation Server Layer (Cloud Layer): The edge servers provide the privacy-protected model parameters to the cloud aggregation server. Then it changes the global model by putting these parameters together. This global model uses the data from all the devices that are part of it, but it does not have direct access to raw user data or unprotected model parameters from specific edge servers.

To achieve the desired lightness, we replace standard convolutional layers in our baseline edge CNN architecture with depthwise separable convolutions, as shown in Fig. 2. A standard $D_k \times D_k$ convolution operation processes spatial and cross-channel correlations simultaneously, the parameter cost is represented as:

$$P = N \times D_k \times D_k \times M, \tag{7}$$

where M is the number of input channels, N is the number of output channels, and D_k is the kernel size.

A depthwise separable convolution, on the other hand, splits the normal convolution into two independent, more efficient operations. The first step is a depthwise convolution, which uses one spatial filter on each of the M input channels. This operation only learns spatial features in each channel and gives an output of M channels in between. The parameter cost for this step can be expressed as:

$$P_{dw} = M \times D_k \times D_k, \tag{8}$$

Following the depthwise convolution, a pointwise convolution is applied. This step uses 1×1 convolutions to project the M channels from the depthwise step homens output channels. The pointwise convolution combines the channel-wise characteristics learnt in the depthwise step in a linear way, which helps control cross-channel correlations. Its parameter cost is given by:

$$P_{pw} = N \times 1 \times 1 \times M, \tag{9}$$

Thus,the total parameter cost for a depthwise separable convolution is the sum of the parameters from these two steps:

$$P_{ds} = (M \times D_k \times D_k) + (N \times M). \tag{10}$$

For instance, in the first convolutional block of our edge CNN, which initially processes $M = 1$ input channel to produce $N = 10$ output channels with a $D_k = 5$ kernel, a standard convolution would require 250 parameters. Using a depthwise separable convolution cuts the number of parameters down to 35, which is almost 86% less for this layer.

This architecture is meant to make use of the computing power of edge servers while using fewer cloud resources. This will fix latency problems and improve data privacy. The method reduces the risk of exposing raw data by processing it locally on IoT devices and edge servers before transmitting it to the cloud. This is critical for consumer electronics that care about privacy. The hierarchical design also strikes a balance between computational efficiency and privacy security by spreading workloads across several layers.

B. Implementation Steps

The proposed terminal-edge-cloud privacy protection architecture includes the following specific steps, as shown in Fig. 3:

• Set up global model parameters on the cloud server and provide each edge server its own set of initial model parameters.

• IoT devices pre-process user data. Before sending the user's raw data to the edge servers deployed in base stations, the devices apply differential privacy techniques, adding Gaussian noise to protect data privacy. They assign different privacy budgets based on data sensitivity, and then transmit the processed data to the edge servers. While ensuring that users' personal privacy data remains confidential, the overall utility of their data must also be guaranteed.

• Edge servers receive processed data from the IoT devices. Edge servers aggregate the data and perform initial model training and parameter updates. Differential privacy techniques are applied to the model parameters, adding Gaussian noise to ensure the privacy of the parameters during transmission and prevent inference of individual data points from model updates.

• The cloud server receives model parameters from the edge servers. It updates and optimizes the global model, generating and storing the final version. The updated model is then disseminated to each edge server for the subsequent round of training and refinement.

The Gaussian noise mechanism mentioned in this paper adds noise $N(0, \sigma^2)$ according to the following formula:

$$A(D) = f(D) + N(0, \sigma^2), \tag{11}$$

where $A(D)$ represents the query result with noise, and $f(D)$ denotes the original query result.

The incorporation of Gaussian noise in differential privacy introduces a trade-off between privacy and model accuracy. Higher noise intensity enhances privacy protection but degrades model convergence by distorting gradient updates. Conversely, lower noise levels improve convergence speed but reduce privacy protection. The variance σ^2 in the Gaussian mechanism is typically determined by the privacy parameters ϵ, δ, and the sensitivity $\triangle f$ of the query. Standard deviation σ is defined as:

$$\sigma = \frac{\sqrt{2ln(1.25/\delta)} \triangle f}{\epsilon}, \tag{12}$$

where the sensitivity $\triangle f$ is defined as the maximum change of the query result on any neighboring dataset. This equation demonstrates that increasing ϵ reduces noise variance, leading to improved convergence while sacrificing privacy.

$$\triangle f = max_{D,D'} \|f(D) - f(D')\|. \tag{13}$$

Additionally, we have analyzed the Gaussian noise mechanism within the framework of differential privacy. Although the Gaussian noise mechanism introduces a small probability parameter δ, which allows the privacy protection mechanism to fail with a very low probability, this design actually enhances the practicality of differential privacy. In pure ϵ-differential privacy, the output distributions of data must be almost identical in all cases, which can result in the addition of excessive noise when dealing with complex and high-dimensional data, thereby affecting the usability and accuracy of the data. In contrast, (ϵ, δ)-differential privacy allows for a very small failure probability δ, and in most cases, it can still provide protection levels close to ϵ-differential privacy, but with significantly reduced noise, thereby improving data usability. Especially for high-dimensional data and multiple query scenarios, the Gaussian mechanism, while satisfying (ϵ, δ)-differential privacy, is more effective than the pure ϵ-differential privacy mechanism. Therefore, although δ introduces a minimal risk of privacy leakage, this risk can be set to an extremely low level, almost negligible, while the overall effectiveness of privacy protection and data usability is significantly enhanced.

C. TECDP Algorithms

This workk presents two-step differential privacy algorithms designed to safeguard data privacy in edge servers and IoT devices. Algorithm 1 shows a way for IoT devices to use differential privacy, where users can set a privacy level λ to control how much noise there is. Based on user data and privacy settings set by the user, the algorithm figures out the noise scale σ. The noise scale has a direct effect on how much noise is introduced to each data point. A bigger σ protects privacy better but may make the data less useful. A smaller σ gives more accurate data but less privacy protection. Gaussian noise is introduced to each data point, which makes the data noisy and protects the privacy of the data.

Algorithm 1 Differential Privacy in IoT Device.

Input: User Privacy Data D, Privacy Budget ϵ, Failure Probability δ, Sensitivity $\triangle f$, User-defined Privacy Level λ
Output: Noisy Privacy Data D'
1.**Initialize** empty list D' for noisy data
2.$\sigma \leftarrow \frac{\sqrt{2ln(1.25/\delta)}\triangle f}{\lambda\epsilon}$ // σ is the noise scale
3.**for** each data point x in D **do**
4. $\eta \leftarrow N(0, \sigma^2)$ // η is the Gaussian noise
5. $x' \leftarrow x + \eta$ // Add noise
6. $D' \leftarrow x'$
7.**end for**
8.**return** D'

Algorithm 2 Differential Privacy in Edge Servers.

Input: Local Dataset D, Local Model M, Privacy Budget ϵ, Failure Probability δ, Sensitivity $\triangle f$, Global Model G
Output: Model parameters with added Gaussian noise w_G

1.**Initialize** Local model M, Global model G
2.$\sigma \leftarrow \frac{\sqrt{2ln(1.25/\delta)}\triangle f}{\epsilon}$ // σ is the noise scale
3.**for** each local epoch **do**
4. **for** each minibatch b in D **do**
5. $g \leftarrow \nabla M(D_b)$ // g is the gradient
6. $\eta \leftarrow N(0, \sigma^2)$ // η is the Gaussian noise
7. $g' \leftarrow g + \eta$ // Update gradient with noise
8. Update local model M with noisy gradient g'
9. **end for**
10.**end for**
11.Aggregate local model parameters to update global model G
12.**return** G, w_G

In Algorithm 2, local and global models are initialized, and the noise scale δ is calculated based on the privacy budget ϵ, failure probability δ, and sensitivity Δf. The privacy budget ϵ controls the overall privacy protection level, with smaller values of ϵ providing stronger privacy protection at the cost of increased noise. The sensitivity $\triangle f$ determines the maximum possible change in the output. This change is observed in the model gradient when a single data entry is modified. The approach calculates the gradient for each minibatch of data during each local training epoch, adds Gaussian noise to the gradient, and then uses the noisy gradient to update the local model. Finally, the parameters from the local models are combined to make the global model better. This adds Gaussian randomness to the model parameters. These two algorithms effectively achieve differential privacy protection by adding Gaussian noise, allowing for the protection of sensitive data

during training and enabling users to control the level of privacy protection according to their needs.

The selection of data distribution during the training of user privacy data substantially influences model efficacy and privacy safeguarding. We looked at both Independent and Identically Distributed (IID) and Non-Independent and Identically Distributed (NIID) data sets to deal with the fact that real-world data is not always the same. The differential privacy method described in this paper protects privacy during training by adding Gaussian noise. This allows for effective model training while keeping user data safe. A detailed explanation of IID and NIID distributions is provided in the experimental section.

V. Experimental Results and Performance Evaluation

A. Experimental Setup and Datasets

We conducted our experiments on a server equipped with a 12th Gen Intel(R) Core (TM) i7 processor and an NVIDIA GeForce RTX 4090 GPU, using Python 3.10 and the PyTorch library as development tools. This experiment considered a terminal-edge-cloud system architecture, including clients, edge servers, and a cloud aggregation server. There were 50 clients, 5 edge servers, and 1 cloud aggregation server in the edge-cloud system. We made sure that each edge server had the same number of clients, which kept the amount of data on each edge server the same.

We used the standard MNIST and CIFAR-10 datasets to test our strategy. These are standard choices in machine learning research. They are straightforward yet effective for evaluating a model's generalization capabilities. MNIST is well-known for handwritten digit recognition. CIFAR-10 is harder because it contains small color images from many different classes. We chose these specifically to simulate scenarios common in consumer electronics. MNIST represents simple Optical Character Recognition (OCR) tasks on resource-constrained devices like smartwatches or small sensors. CIFAR-10 covers more complex image classification and object recognition. This complexity is crucial for functions in smart cameras and smart home devices.

We used the MNIST and CIFAR-10 datasets to do our tests. MNIST has 60,000 training photos and 10,000 test images of 28x28 pixel grayscale handwritten digits. CIFAR-10 has 60,000 training images and 10,000 test images of 32x32 pixel colour images across 10 classes. In the edge layer, we employed a lightweight convolutional neural network (CNN) with two convolutional layers and two fully connected layers, chosen for its simplicity and efficiency in edge deployment. At the cloud layer, a more complex CNN with three convolutional blocks was used to enhance performance. Table II shows the experimental hyperparameters. Each client performed local computation using mini-batch Stochastic Gradient Descent (SGD) with a batch size of 20 and an initial learning rate of 0.01 for MNIST and 0.1 for CIFAR-10, decaying exponentially per epoch at rates of 0.995 and 0.992, respectively. The centralized training of this model achieved over 90 percents test accuracy. Table III presents key information about the MNIST and CIFAR-10 datasets, including image size and the number of training and test samples, providing a clearer comparison of their characteristics.

TABLE II
Training Hyperparameters

Hyperparameter	Notation	Value
Client Number	N_c	50
Edge Number	N_e	5
Batch Size	B	20
Learning Rate	η	0.01
Local Epoch	E	1
Epsilon	ϵ	20
Delta	δ	1e-5
Learning Rate Decay	γ	0.995
Communication Number	K	100

TABLE III
Dataset

Dataset	Image Size	Training Samples	Test Samples
MNIST	28×28	60000	10000
CIFAR-10	32×32	50000	10000

To enhance privacy protection, we included a differential privacy approach that uses Gaussian noise. We introduced Gaussian noise to the parameters depending on the privacy budget parameters ϵ and δ after each model update to make sure that client private data was not leaked during training. This mechanism protected privacy within a specific privacy budget by adding the right amount of noise to each model update. The standard deviation of Gaussian noise was changed on the fly based on the privacy settings and the global learning rate to find the ideal balance. The learning rate η set the size of the steps that were taken to change the model parameters during training. A higher learning rate meant that each update had a bigger step size, which could mean that more noise was needed to make sure privacy was protected. If the learning rate were lower, on the other hand, the step size would be smaller, which would make the noise less.

B. Data Distribution

In federated learning, data distribution significantly affects model training and privacy protection performance. This experiment included two data distribution methodologies: Independent and Identically Distributed (IID) and Non-Independent and Identically Distributed (NIID).

Independent and Identically Distributed (IID): In this case, data samples for each client are randomly taken from the whole dataset, making sure that the data is spread out evenly among all clients. This way of distributing helps with the speed and uniformity of model training. In edge computing settings, IID data distribution can make full use of edge devices' processing power. This cuts down on the uncertainties and extra communication costs that come with data heterogeneity during model training.

Non-Independent and Identically Distributed (NIID): The data samples for each client come from different data subsets, which suggests that the data is not evenly distributed

TABLE IV
ACCURACY UNDER DIFFERENT PRIVACY BUDGETS

Privacy Budgets	Mnist	Cifar-10
ϵ=10.0	0.3664	0.3595
ϵ=20.0	0.7322	0.7179
ϵ=30.0	0.7736	0.7623
ϵ=40.0	0.8112	0.8040
ϵ=50.0	0.8424	0.8356

out among all clients. This form of distribution is more like how data is actually distributed in the real world, but it also makes it harder to train the model. In edge computing environments, federated learning under NIID data distribution is more challenging but can better simulate the diversity and heterogeneity of real-world data. To address this, we used differential privacy and adaptive gradient descent strategies on edge devices to balance data utility and privacy protection.

Federated learning can better safeguard user privacy with diverse data dissemination methods when the edge-cloud architecture is used. Edge devices may effectively gather local model updates for IID data distribution. This cuts down on the number of times data needs to be sent and lowers the danger of data leaks. The edge-cloud design can make full use of the dispersed computing capability of edge devices for local data preprocessing and encryption, which improves privacy protection for NIID data delivery. Additionally, edge-cloud collaboration can effectively address network latency and bandwidth limitation issues, ensuring system robustness and efficiency, enabling the system to continue functioning normally even if some nodes fail.

C. Metrics

To fully evaluate how well the suggested solution protects user privacy and keeps data useful in the context of consumer electronics, this section will introduce and use two evaluation measures. These measures will assist us figure out how accurate the procedure is and how well it protects privacy.

We chose two important assessment metrics for this experiment: accuracy and privacy protection strength. Accuracy is an important way to judge how well the model predicts things overall. Our approach consistently achieves a high accuracy across various datasets and privacy budgets (ϵ). The detailed results are presented in Table III.

For privacy protection strength, we use the differential privacy budget (ϵ) to measure the degree of privacy protection. A smaller ϵ value indicates stronger privacy protection. This metric is used to evaluate the effectiveness of our method in safeguarding user data privacy. As the ϵ value decreases, the strength of privacy protection increases, but the model's accuracy slightly decreases. This indicates that while ensuring privacy, our method can still maintain satisfactory model performance.

D. Experimental Results and Analysis

Fig. 4 respectively illustrates the model accuracy under differential privacy settings when the privacy budget is set

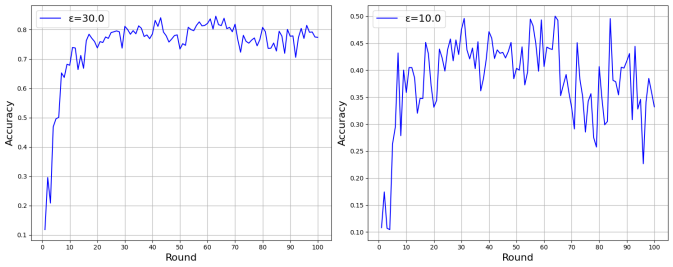


Fig. 4. Accuracy with privacy budget of 10.0 and 30.0.

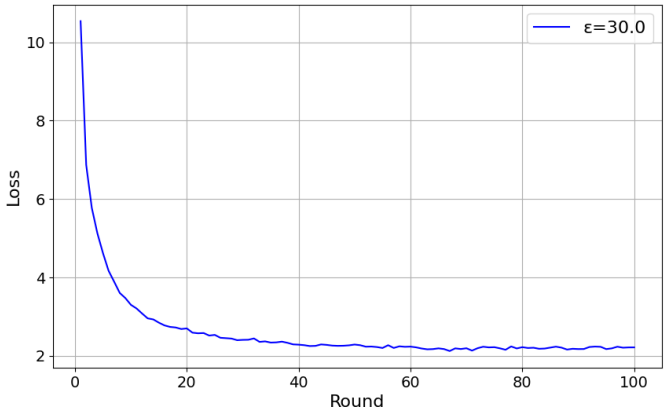
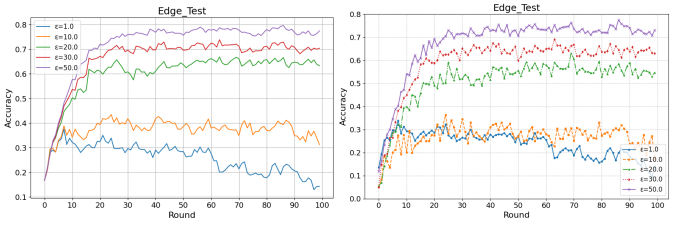


Fig. 5. Training loss over epochs.

to 10.0 and 30.0. With a privacy budget of 10.0, the level of protection for user private data is too high, which greatly lowers the model's accuracy and makes it useless in real-world situations. So, when the privacy budget is set to 30.0, the model gets more accurate. This shows that it is possible to safeguard privacy by adding noise while still making the model work well. When using data in real life, it's important to find a balance between protecting consumers' privacy and making the data useful.

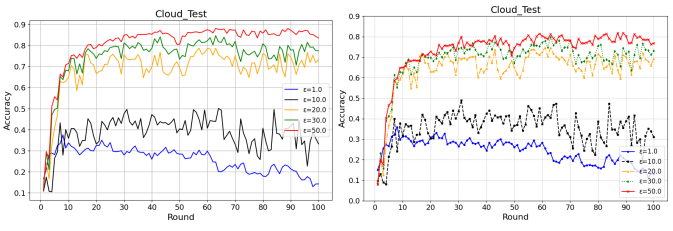
Fig. 5 shows how the model's loss values changed during training from the 1st to the 100th epoch. In the first 20 epochs, the loss values drop quickly, which means that the model is learning and optimizing quickly at this time. After the 20th epoch, the loss values gradually stabilize, signifying that the model is progressively achieving convergence.



(a) Edge Test Accuracy on MNIST (b) Edge Test Accuracy on CIFAR-10

Fig. 6. Accuracy Comparison under different epsilon values within the edge layer.

Fig. 6 and Fig. 7 analyze the performance of our method on the dataset using Gaussian noise for differential privacy protection, tested under both IID and NIID data distributions.



(a) Cloud Test Accuracy on MNIST (b) Cloud Test Accuracy on CIFAR-10

Fig. 7. Accuracy Comparison under different epsilon values within the cloud aggregation layer.

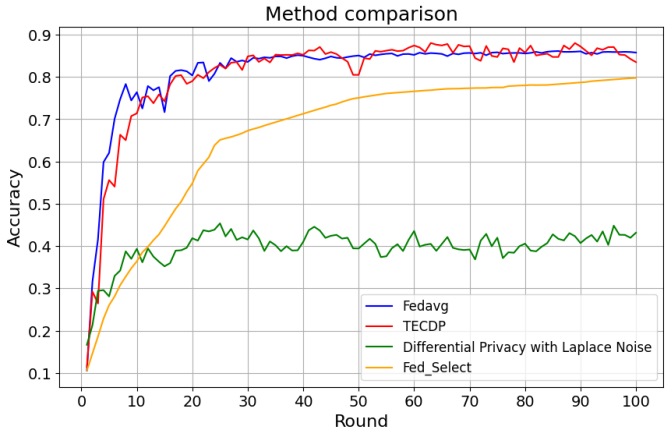


Fig. 8. Comparison of accuracy across different methods.

The figures show the accuracy under different privacy budgets (ϵ) set to 1.0, 10.0, 20.0, 30.0, and 50.0. As the privacy budget ϵ increases, the model accuracy gradually improves. This is because a higher ϵ value corresponds to less noise addition, thereby preserving more useful information. Fig. 6 shows the model test accuracy after privacy-protected data aggregation at the edge layer. This accuracy reflects the performance of the edge layer only. The model accuracy in this figure is based on the different privacy budgets applied at the user IoT devices. Fig. 7 illustrates the model test accuracy after the cloud aggregation layer aggregates the updated model parameters from the edge servers. The accuracy at the cloud aggregation layer is adjusted by changing the privacy budget during the transmission of parameters from the edge layer, while the privacy budget at the terminal layer remains unchanged. It can be observed that the accuracy at the cloud aggregation layer is relatively higher because it benefits from a more powerful model for classification. This demonstrates that our method can maintain high accuracy while satisfying user privacy protection needs.

In the experiment, we compared the accuracy performance of TECDP in model training with three different methods, which include the traditional Federated Averaging (FedAvg) algorithm, Fed_Select [35], and the differential privacy method with Laplace noise. Among these, Fed_Select is a new privacy-preserving federated learning method. FedAvg averages model updates from clients without differential privacy, serving as a baseline to assess privacy protection's impact on accuracy. Fed_Select optimizes client selection to improve convergence while maintaining privacy. Laplace Noise Differential Privacy adds Laplace noise to protect data privacy. TECDP leverages the edge layer to perform lightweight local training and intermediate aggregation. This architectural choice is intended to reduce communication overhead and lower the computational burden on user IoT devices. The experimental results show that TECDP performed the best.

Fig. 8 shows that TECDP, FedAvg, and Fed Select all had very high accuracy. The differential privacy approach with Laplace noise, on the other hand, had the lowest accuracy of the four methods. Further investigation shows that TECDP and FedAvg approaches have about the same level of accuracy, but TECDP is far better at protecting privacy. When we compared TECDP to Fed_Select, we observed that Fed_Select's convergence time was slower and its accuracy at the 100th round was not as good as TECDP's. We examined Gaussian noise with Laplace noise with the same privacy budget. We observed that the Laplace noise approach should ensure privacy in theory, but it doesn't work as well in practice since it isn't accurate enough. Our method keeps a high level of accuracy while also safeguarding user privacy by adding noise in the right way.

The experimental results show that TECDP works better than baseline methods because it strikes a good balance between protecting privacy and getting accurate models. The solution employs local differential privacy on IoT devices and further security on the edge server. The architecture significantly improves edge-cloud collaboration by reducing latency and resource utilization compared to FedAvg. TECDP works well with both IID and non-IID data distributions, which makes sure that convergence is steady. Also, utilizing Gaussian noise keeps excellent privacy protections with very little loss of accuracy. These things make TECDP better at privacy-preserving federated learning for consumer electronics.

In summary, our TECDP approach not only works well, but it also has some distinct benefits when it comes to protecting privacy. This is a good way to preserve privacy in situations when rigorous privacy protection is needed, like in the consumer electronics area, while also making sure that data is useful.

VI. Conclusion

This study presents an innovative way for enhancing privacy and security in consumer electronics through the integration of a terminal-edge-cloud federated learning architecture with two-step differential privacy. We added depthwise separable convolutions to address the need for a lightweight CNN at the edge layer. This method adds a multi-layered privacy protection system, which makes sure that data is safe while keeping the model's accuracy and efficiency high. Experiments performed on diverse datasets and varying data distributions illustrate its efficacy across varied privacy budget contexts. The findings suggest that the equilibrium between privacy safeguarding and data functionality is essential for the effective implementation of federated learning in consumer electronics. Future work will concentrate on enhancing the equilibrium between privacy and utility, while investigating supplementary applications across various consumer electronics.

VII. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant (No. 92267104 and 62372242), and in part by Jiangsu Provincial Major Project on Basic Research of Cutting-edge and Leading Technologies under Grant (No. BK20232032).

REFERENCES

[1] Haewon Byeon et al. “Lightweight AI and Blockchain Optimization for Enhancing Consumer Electronics Decision-Making”. In: *IEEE Transactions on Consumer Electronics* 71 (2025), pp. 6007–6015.

[2] Xiao-Yu Zhang et al. “Robust Lightweight UAV Inspection System for Consumer Electronics Applications in Smart Grids”. In: *IEEE Transactions on Consumer Electronics* 71 (2025), pp. 3118–3128.

[3] Fahad Masood et al. “AI-based wireless sensor IoT networks for energy-efficient consumer electronics using stochastic optimization”. In: *IEEE Transactions on Consumer Electronics* 70.4 (2024), pp. 6855–6862.

[4] Hou-I Liu et al. “Lightweight deep learning for resource-constrained environments: A survey”. In: *ACM Computing Surveys* 56.10 (2024), pp. 1–42.

[5] Jia-Hao Syu et al. “A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics”. In: *IEEE Transactions on Consumer Electronics* 69.4 (2023), pp. 1023–1034.

[6] Bowen Liu et al. “CEIO: A Cache-Efficient Network I/O Architecture for NIC-CPU Data Paths”. In: *Proceedings of the ACM SIGCOMM 2025 Conference*. 2025, pp. 381–394.

[7] Junxue Zhang et al. “Liteflow: towards high-performance adaptive neural networks for kernel datapath”. In: *Proceedings of the ACM SIGCOMM 2022 Conference*. 2022, pp. 414–427.

[8] Xinyang Huang et al. “Accelerating privacy-preserving machine learning with GeniBatch”. In: *Proceedings of the Nineteenth European Conference on Computer Systems*. 2024, pp. 489–504.

[9] Jingxue Chen et al. “When federated learning meets privacy-preserving computation”. In: *ACM Computing Surveys* 56.12 (2024), pp. 1–36.

[10] Megha Sharma, Abhinav Tomar, and Abhishek Hazra. “Edge computing for industry 5.0: Fundamental, applications and research challenges”. In: *IEEE Internet of Things Journal* 11.11 (2024), pp. 19070–19093.

[11] Mohammad Kamrul Hasan et al. “Federated learning for computational offloading and resource management of vehicular edge computing in 6G-V2X network”. In: *IEEE Transactions on Consumer Electronics* 70.1 (2024), pp. 3827–3847.

[12] Jiajun Wu et al. “Topology-aware federated learning in edge computing: A comprehensive survey”. In: *ACM Computing Surveys* 56.10 (2024), pp. 1–41.

[13] Xin Wang et al. “Privacy-preserving AI framework for 6G-enabled consumer electronics”. In: *IEEE Transactions on Consumer Electronics* 70.1 (2024), pp. 3940–3950.

[14] Izhar Ahmed Khan et al. “Fed-inforce-fusion: A federated reinforcement-based fusion model for security and privacy protection of IoMT networks against cyber-attacks”. In: *Information Fusion* 101 (2024), pp. 102002.

[15] Muhammad Bilal and Sangheon Pack. “Secure Distribution of Protected Content in Information-Centric Networking”. In: *IEEE Systems Journal* 14.2 (2020), pp. 1921–1932.

[16] Raihan Ur Rasool et al. “Security and privacy of internet of medical things: A contemporary review in the age of surveillance, botnets, and adversarial ML”. In: *Journal of Network and Computer Applications* 201 (2022), pp. 103332.

[17] Xiaoding Wang et al. “Federated learning-empowered disease diagnosis mechanism in the internet of medical things: From the privacy-preservation perspective”. In: *IEEE Transactions on Industrial Informatics* 19.7 (2022), pp. 7905–7913.

[18] Qingxin Lin et al. “Fed-PEMC: A privacy-enhanced federated deep learning algorithm for consumer electronics in mobile edge computing”. In: *IEEE Transactions on Consumer Electronics* 70.1 (2024), pp. 4073–4086.

[19] Elnaz Rabieinejad et al. “Two-level privacy-preserving framework: Federated learning for attack detection in the consumer internet of things”. In: *IEEE Transactions on Consumer Electronics* 70.1 (2024), pp. 4258–4265.

[20] Mostafa M Fouda et al. “A lightweight hierarchical AI model for UAV-enabled edge computing with forest-fire detection use-case”. In: *IEEE Network* 36.6 (2022), pp. 38–45.

[21] Areej A Malibari. “An efficient IoT-Artificial intelligence-based disease prediction using lightweight CNN in healthcare system”. In: *Measurement: Sensors* 26 (2023), pp. 100695.

[22] Abdelhamied A Ateya et al. “Lightweight deep learning-based model for traffic prediction in fog-enabled dense deployed iot networks”. In: *Journal of Electrical Engineering & Technology* 18.3 (2023), pp. 2275–2285.

[23] Shuai Wang et al. “Blockchain-empowered distributed multicamera multitarget tracking in edge computing”. In: *IEEE Transactions on Industrial Informatics* 20.1 (2023), pp. 369–379.

[24] Hao Frank Yang et al. “Cooperative multi-camera vehicle tracking and traffic surveillance with edge artificial intelligence and representation learning”. In: *Transportation research part C: emerging technologies* 148 (2023), pp. 103982.

[25] Jingzong Li et al. “Cross-camera inference on the constrained edge”. In: *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE. 2023, pp. 1–10.

[26] Rui Song et al. “FedBEVT: Federated learning bird’s eye view perception transformer in road traffic systems”. In: *IEEE Transactions on Intelligent Vehicles* 9.1 (2023), pp. 958–969.

[27] Renyou Xie et al. “Asynchronous federated learning for real-time multiple licence plate recognition through semantic communication”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[28] Weidong Fang, Chunsheng Zhu, and Wuxiong Zhang. “Toward secure and lightweight data transmission for cloud–edge–terminal collaboration in artificial intelligence of things”. In: *IEEE Internet of Things Journal* 11.1 (2023), pp. 105–113.

[29] Chongbo Wei, Gaogang Xie, and Zulong Diao. “A lightweight deep learning framework for botnet detecting at the IoT edge”. In: *Computers & Security* 129 (2023), pp. 103195.

[30] Muah Kim, Onur Günlü, and Rafael F Schaefer. “Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 2650–2654.

[31] Javier Mendez et al. “Edge intelligence: concepts, architectures, applications, and future directions”. In: *ACM Transactions on Embedded Computing Systems (TECS)* 21.5 (2022), pp. 1–41.

[32] Wenke Huang, Mang Ye, and Bo Du. “Learn from others and be yourself in heterogeneous federated learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10143–10153.

[33] Ying Zhao and Jinjun Chen. “A survey on differential privacy for unstructured data content”. In: *ACM Computing Surveys (CSUR)* 54.10s (2022), pp. 1–28.

[34] Mengmeng Yang et al. “Local differential privacy and its applications: A comprehensive survey”. In: *Computer Standards & Interfaces* (2023), pp. 103827.

[35] Akarsh K Nair, Jayakrushna Sahoo, and Ebin Deni Raj. “Privacy preserving Federated Learning framework for IoMT based big data analysis using edge computing”. In: *Computer Standards & Interfaces* 86 (2023), pp. 103720.