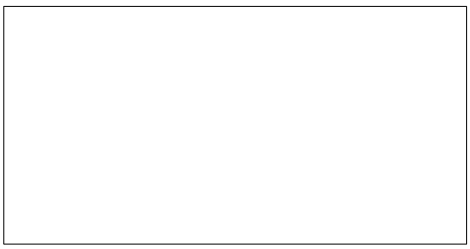


International Journal of Information Technology
https://doi.org/10.1007/s41870-025-03010-0

ORIGINAL RESEARCH

FEDGE: FEDerated learning at the EDGE on space platforms using deep neural network architectures

Rashmi Siddalingappa[1] · Deepa S[2] · Priya Stella Mary I[2] · Kalpana P[2] · Lakshmi B A[3]

Received: 7 September 2025 / Accepted: 3 December 2025

Abstract

We introduce FEDGE: FEDerated Learning at the EDGE, a framework designed for efficient AI deployment in resource-constrained satellite constellations. FEDGE integrates federated learning with edge computing to address communication overhead and latency challenges in distributed space environments. The framework features a novel edge-enhanced ground station protocol that dynamically schedules model aggregation based on satellite-provided metadata, combined with local stochastic gradient descent training at satellite edge devices and gradient compression via quantization. Experimental validation on MNIST and EuroSAT datasets demonstrates the practical viability of the approach. On MNIST, FEDGE achieved 94.33% training accuracy with 0.21 loss and 90.05% test accuracy with 0.24 loss. On EuroSAT, the framework reached 93.47% training accuracy with 0.18 loss and 91.51% test accuracy with 0.21 loss. Gradient quantization reduces data exchange by up to 14× with approximately 4% impact on test loss. These results validate FEDGE as a communication-efficient solution for decentralized AI deployment in satellite systems, enabling autonomous spacecraft intelligence and addressing the unique constraints of space-based computing platforms.

Keywords Federated Learning, Edge Computing, Machine Learning, Internet-of-Things, Deep Neural Network Architecture, Stochastic Gradient Descent

1 Introduction

Federated Learning (FL) marks a paradigm shift in machine learning (ML), enabling decentralized model training that mitigates the privacy and communication limitations of centralized systems [1, 2]. Unlike traditional ML, where data is aggregated in cloud servers, FL trains models directly on distributed edge devices, sharing only model updates instead of raw data. This approach enhances data privacy, reduces communication overhead, and is ideal for bandwidth-constrained and latency-sensitive environments [3]. Edge Computing (EC) complements FL by extending cloud capabilities closer to data sources, reducing latency and bandwidth use while improving reliability [4]. EC enables real-time processing for IoT applications in healthcare, smart manufacturing, and UAVs [5]. However, managing massive IoT data under strict performance and availability constraints remains challenging. Space platforms, such as satellites and spacecraft, further amplify these challenges due to limited bandwidth and latency-critical operations. These systems must autonomously perform tasks like object detection, localization, and beamforming, which conventional cloud architectures cannot efficiently support. To address this, we propose **FEDGE**—*FEDerated Learning at the EDGE on Space Platforms Using DNN Architectures*—a framework combining FL and EC to achieve real-time, communication-efficient learning in space environments.

✉ Rashmi Siddalingappa
siddalingapparashmi@gmail.com

Deepa S
sdeepa369@gmail.com

Priya Stella Mary I
priya.stella@christuniversity.in

Kalpana P
kalpana.p@christuniversity.in

Lakshmi B A
lakshmishashidhar12@gmail.com

[1] Department of Computer and Data Science, York St John University, London E14 2BA, UK

[2] Department of Computer Science, Christ University, Bangalore, Karnataka 560073, India

[3] UST Global, Bangalore, India

Published online: 19 December 2025

While future integration with AI-specific hardware such as ASICs is envisioned, this study remains hardware-agnostic and focuses on developing lightweight, communication-efficient DNN architectures suitable for low-power, resource-constrained systems.

2 Background study

Table 1 summarizes methods, datasets, and limitations, forming the foundation for FEDGE. While prior algorithms like FedAvg assume IID clients or need careful tuning, few studies jointly consider (i) intermittent client availability,

Table 1 Summary of related federated learning and edge-AI works. This table highlights the main problem, core method, datasets / experimental setup, key findings, and limitations

Refs.	Problem/focus	Core method	Datasets/setup	Key findings and limitations
[6]	Federated Averaging (FedAvg) baseline	FedAvg (local SGD + averaging)	MNIST, EuroSAT (in their evaluation)	Minimized communication via local updates; comparable accuracy to centralized training on MNIST and EuroSAT with fewer rounds. Limitation: assumes IID client data; sensitive to hyperparameters
[7]	Impact of non-IID data on FedAvg	Adaptive averaging, learning rate adjustments	CIFAR-10, ImageNet (experiments on non-IID splits)	Improved accuracy under non-IID via adaptive strategies; requires careful tuning; may not fully resolve severe heterogeneity
[8]	Practical system-level FL considerations	System framework: secure aggregation, client availability	Large-scale mobile deployments (system design)	Addresses scalability, privacy (secure aggregation); focuses on systems, not directly on model accuracy. Limitation: system complexity and integration costs
[9]	On-device language modeling	FedAvg with RNNs for next-word prediction	Mobile keyboard dataset (proprietary)	Comparable accuracy to centralized models; communication-efficient. Limitation: RNN compute cost on-device; privacy preserved but compute expensive
[10]	Non-IID robustness	FedDyn: dynamic regularization for federated optimization	CIFAR variants and synthetic non-IID tests	Outperforms FedAvg in unbalanced/non-IID settings; needs tuning for regularizer strength
[11]	Communication reduction techniques	Quantization and sparsification algorithms	Benchmarks (cite specifics)	Similar accuracy to FedAvg with lower communication; parameter settings crucial for trade-off
[12]	Robustness to client heterogeneity	FedProx: adds a proximal term to stabilize local updates	MNIST, FEMNIST, Sent140, The Complete Works of William Shakespeare	Improves convergence in highly non-IID data; effective for heterogeneous devices; requires tuning of proximal parameter
[13]	Asynchronous staleness-aware FL	FedAsync, FedBuff: asynchronous updates, buffer-based aggregation	CIFAR-10, speech tasks	Enables FL under intermittent connectivity (e.g., satellites); improves efficiency but accuracy degrades under high staleness
[14]	Gradient compression methods	QSGD (quantized gradients), Top-k sparsification	CIFAR-10, ImageNet	Achieves up to $10\times$ communication reduction with minor accuracy loss; adds complexity with error compensation
[15]	Metadata-driven scheduling	Importance sampling / metadata-aware client selection	FEMNIST, Shakespeare dataset	Prioritizes clients with high-value updates, reducing training time; requires metadata collection overhead
[16]	Privacy and secure aggregation	Hybrid differential privacy + secure aggregation frameworks	Healthcare, financial FL	Improves privacy guarantees with low communication overhead; trade-off in model accuracy due to DP noise
[17]	Edge AI in constrained environments	Survey of edge inference/training techniques for IoT/satellite AI	Broad survey across edge/satellite/IoT	Highlights challenges of model compression, lightweight inference, and deployment in constrained environments

(ii) metadata-driven aggregation scheduling, and (iii) quantization/compression for constrained satellite links. FEDGE explicitly addresses this space, offering metadata-aware aggregation and edge-side gradient quantization to enhance algorithmic robustness under real-world edge-space constraints.

2.1 Objectives of FEDGE

The objective of FEDGE is to establish a robust and efficient federated learning (FL) framework for edge-based AI applications. To minimize communication overhead from frequent parameter exchanges in conventional FL (e.g., FedAvg), FEDGE applies gradient compression with quantization at edge devices, reducing cloud-edge dependency and improving network efficiency. To handle data heterogeneity under non-IID distributions, FEDGE employs dynamic scheduling at the central server using edge-provided metadata (e.g., training accuracy, data statistics) to adjust aggregation strategies. For robustness and trustworthiness, it integrates secure aggregation protocols, supporting deployment in sensitive environments. In resource-constrained settings, FEDGE adopts lightweight models and efficient training algorithms to minimize computational load. The core innovation lies in an edge-enhanced ground station (GS) protocol that dynamically schedules model aggregation based on satellite metadata (e.g., training accuracy, round index), reducing communication costs and accommodating intermittent connectivity. Local training is performed on satellite edge devices using stochastic gradient descent (SGD). In spacecraft applications, FEDGE enables real-time image processing for object recognition and tracking, reducing reliance on ground control. For satellites, it supports beamforming optimization through agile beam adjustment, providing high throughput in large-array systems. While long-term goals include ASIC-based AI hardware deployment, this study remains hardware-agnostic, focusing on a foundational, ASIC-compatible FL framework. The proposed DNNs address critical computational tasks such as robotic control loops, intelligent data filtering, and high-bandwidth satellite communication, with potential extension to AI hardware performance verification. The key application scenarios of FEDGE include: (1) Multi-Legged Robotic Locomotion: DNNs manage complex feedback loops and MMA operations for agile navigation in dynamic environments. (2) Spacecraft Intelligence: Onboard DNNs enable real-time sensor data interpretation (object recognition, localization, tracking, segmentation), supporting autonomous decision-making. (3) Satellite Beamforming: DNNs control phased antenna arrays for agile, high-throughput beamforming, deriving phase and amplitude parameters more efficiently than traditional algorithms.

3 Methodology

The FEDGE framework integrates data preparation, federated learning, and edge computing, as summarized in Algorithm 1, covering the end-to-end process from data collection to model deployment. It begins with data preprocessing, followed by federated training with intermittent connectivity simulation, dynamic aggregation, and final deployment. Algorithm 2 (Ground Station procedure) handles model aggregation in Step 7, Algorithm 3 (Satellite Edge Computing) performs local training in Step 6, and Algorithm 4 (Stopping Criteria) determines training termination in Step 9.

FEDGE enables collaborative training between satellites (clients) and the ground station (GS), ensuring data privacy and minimizing communication costs. The GS maintains a global model (w_i) and coordinates synchronization using metadata, including training accuracy and round index. Satellites train locally with stochastic gradient descent (SGD), compute gradients, and send updates to the GS for aggregation. The GS procedure initializes the global model w_0 and round index $i = 0$. Step 1: Satellites upload gradients (g_k), round indices ($i_{g,k}$), and metadata (meta_k) to the GS buffer B_i, along with staleness values $s_k = i_g - i_{g,k}$. Step 2: If the scheduler condition $a_i = 1$ is met, the GS updates the global model and increments the round counter. Step 3: The updated model is broadcast to connected satellites. The satellite procedure starts by receiving w_i and i_g from the GS. Local data D_k are preprocessed (normalization and augmentation), and the local model is initialized with the global model. Each satellite performs E SGD steps with learning rate η, compresses the trained model via quantization, computes the gradient g_k (difference between trained and initial models), and transmits it with metadata to the GS. Stopping criteria are handled as described in Algorithm 4.

3.1 Edge computing

As computing advances, services are shifting from centralized cloud servers to distributed edge devices such as smartphones, wearables, and IoT sensors. Massive data generation at the edge makes cloud-based processing inefficient due to latency, bandwidth limits, and security risks. Edge computing overcomes these challenges through localized processing, reducing cloud dependency and improving response time, security, and bandwidth efficiency [18]. Unlike traditional cloud systems, where edge devices only consume data, edge computing enables them to act as both data producers and processors, leveraging distributed computing to enhance performance, reduce network congestion, and improve scalability. This architecture supports seamless integration with 5 G and AI-driven analytics. The latency of

Algorithm 1 Complete FEDGE methodology workflow

Input: Distributed datasets $\{D_k\}$, initial model w_0
Output: Final model w_{final}
Step 1: Data collection and initialization
Collect raw data from satellite edge devices (imagery, sensor readings).
Initialize global model w_0 at the Ground Station (GS).
Set hyperparameters: learning rate η, local epochs E, compression factor, number of clients N.
Step 2: Data preprocessing at edge devices
for each client $k \in \{1, \ldots, N\}$ **do**
    Preprocess local dataset D_k (normalization, augmentation, resizing to model input size)
    Store preprocessed data locally
end for
Step 3: Simulate non-IID distributions
Generate client partitions via Dirichlet sampling: $\pi^{(i)} \sim \text{Dirichlet}(\beta\theta)$
Assign samples to each client according to $\pi^{(i)}$
Step 4: Federated training initialization
Broadcast initial model w_0 to all satellites
Set round counter $r \leftarrow 0$ and maximum rounds $R_{\max}$
while $r < R_{\max}$ **and** stopping criteria not met **do**
    $r \leftarrow r + 1$
    Step 5: Simulate intermittent connectivity
    Generate binary availability mask (Bernoulli drop probability p)
    Determine active client set C_r
    Step 6: Local training at satellites (see Alg. 2)
    for each active client $k \in C_r$ **do**
        Perform local training on D_k (SGD for E epochs)
        Compress local updates and upload $(g_k, i_{g,k}, \text{meta}_k)$ to GS
    end for
    Step 7: Model aggregation at GS (see Alg. 1)
    GS receives updates, applies scheduler, aggregates and (if triggered) updates w
    Broadcast updated (w, i_g) to clients
    Step 8: Model evaluation
    Evaluate global model on validation/test data and log metrics (loss, accuracy, precision, recall, F1)
    Step 9: Check stopping criteria
    if any stopping condition is satisfied **then**
        break
    end if
end while
Step 10: Final model deployment
$w_{\text{final}} \leftarrow w$
Deploy w_{final} to satellites for inference
Step 11: Performance analysis
Generate ROC/PR curves and analyze communication efficiency and computational complexity
Compare FEDGE with baseline FL methods
return w_{final}

conventional cloud-based systems is characterized as shown in Eqs. (1) and (2).

$$D = D_c + D_t + D_p \tag{1}$$

where: D_c is computational delay in the cloud, D_t is transmission delay, D_p is processing delay at the edge. In edge computing, local processing reduces delay:

$$D_{\text{edge}} = D_p + D_e \tag{2}$$

where D_e is local edge processing time.

3.2 Federated learning (FL)

Federated learning is a distributed ML paradigm that trains models on many devices without sharing the data. FL is crucial for privacy-sensitive applications like space communications, healthcare and finance and so on [19]. Instead of transmitting entire datasets to a central server, FL enables local training on edge devices, such as smartphones and

Algorithm 2 Edge-enhanced ground station procedure

Input: Initial model w_0
Initialization: $i = 0$, $i_g = 0$, $B_0 = \emptyset$, $R_0 = \emptyset$
repeat
  Step 1: Receive Updates
  for each $k \in C_i$ (satellite connected to GS) **do**
    Receive $(g_k, i_{g,k}, \text{meta}_k)$
    $B_i \leftarrow B_i \cup \{(g_k, s_k)\}$, where $s_k = i_g - i_{g,k}$
    $R_i \leftarrow R_i \cup \{k\}$
  end for
  Step 2: Scheduler Decision
  if $a_i = 1$ **then**
    Perform model update: $w_{i+1} \leftarrow \text{ServerUpdate}(w_i, B_i)$
    $i_g \leftarrow i_g + 1$
    $B_{i+1} \leftarrow \emptyset$, $R_{i+1} \leftarrow \emptyset$
  end if
  Step 3: Broadcast Updates
  Transmit (w_{i+1}, i_g) to satellites in C_i
  $i \leftarrow i + 1$
until Stopping criterion is met

Algorithm 3 Satellite edge computing and federated learning procedure

Input: Global model w_0, local dataset D_k
repeat
  Step 1: Receive Updates
  Receive (w_i, i_g) from GS
  Step 2: Edge Processing
  Preprocess raw data from D_k (e.g., normalization, augmentation)
  Step 3: Local Training
  $w_0^k \leftarrow w_i$
  for $j = 0$ to $E - 1$ **do**
    $w_{j+1}^k \leftarrow w_j^k - \eta \nabla f(w_j^k, X_j^k)$
  end for
  Step 4: Gradient Compression and Upload
  $w_E^k \leftarrow \text{Compress}(w_E^k)$
  $g_k \leftarrow w_E^k - w_0^k$
  Transmit $(g_k, i_g, k, \text{meta}_k)$ to GS
until Stopping criterion is met

Algorithm 4 Stopping criteria for FEDGE framework

Criterion 1: Fixed Number of Global Rounds Stop after a predefined number N.
Criterion 2: Convergence of Global Model Stop if $\Delta = |L(w_{i+1}) - L(w_i)| \leq \epsilon$.
Criterion 3: Target Accuracy Achieved Stop if $\text{Accuracy}_{\text{global}} \geq T_{\text{accuracy}}$.
Criterion 4: Communication Budget Exhausted Stop if $C_{\text{total}} \geq B_{\text{budget}}$.
Criterion 5: Energy Constraints Stop if $E_{\text{total}} \geq E_{\max}$.
Criterion 6: No Significant Model Updates Stop if $\|g_k\| \leq \tau$, $\forall k \in C$.
Criterion 7: Time Constraints Stop if $t_{\text{elapsed}} \geq T_{\max}$.

IoT sensors, and periodically aggregates model updates on a central or edge server. The global model aggregation is shown in Eqs. (3) and (4):

$$W_{t+1} = \sum_{i=1}^{C_n} \frac{n_i}{C_n} W_t^i \tag{3}$$

where: C_n is the total clients, n_i represents data points at client i, W_t^i is the local model at time t. Each client optimizes its local model using:

$$L(W) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(W, x_j, y_j) \tag{4}$$

where $\ell(W, x_j, y_j)$ is the sample loss (x_j, y_j).

FL follows three main steps: (1) Globle model initialization: The initial model is sent to all devices. (2) Localized training: Clients train on their private data. (3) Model update and aggregation: The server gathers updates and refines the model.

3.3 Deep neural networks (DNNs)

DL employs Artificial Neural Networks (ANNs) with several hidden layers to understand hierarchical structures. A typical DNN neuron processes input features using weighted summation followed by an activation function (Eq. 5):

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{5}$$

where: x_i represents input features, w_i denotes weights, b is the bias term, $f(\cdot)$ is an activation function (e.g., ReLU, Sigmoid, Softmax). Training involves minimizing a loss function L using backpropagation and gradient descent (Eq. 6):

$$W = W - \gamma \frac{\partial L}{\partial W} \tag{6}$$

where γ is the learning rate. Common architectures include Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) [20]. Traditional deep learning (DL) models require centralized data in cloud servers for training, posing challenges including high communication costs from transmitting massive datasets and data privacy concerns due to legal and ethical constraints on sensitive information transfer. FL addresses these limitations through distributed DNN training at edge/remote nodes [3], sharing only model parameters (gradients) rather than raw data. Table 2 presents the architecture of the DNN model, including input shape, layer configurations (e.g. Conv2D, MaxPooling2D), dense layers, and optimization settings.

Table 2 DNN model parameters

Parameter	Value
Input shape	(32, 32, 3)
Number of convolutional layers	2
Conv2D filters (Layer 1)	32
Conv2D kernel size (Layer 1)	(3, 3)
Conv2D activation (Layer 1)	ReLU
MaxPooling2D pool size (Layer 1)	(2, 2)
Conv2D filters (Layer 2)	64
Conv2D kernel size (Layer 2)	(3, 3)
Conv2D activation (Layer 2)	ReLU
MaxPooling2D pool size (Layer 2)	(2, 2)
Dense layer size	10
Dense layer activation	Softmax
Optimizer	SGD
Learning rate	0.01
Loss function	Sparse categorical crossentropy
Metrics	Accuracy

3.4 Metrics

We conducted experiments using two benchmark datasets: MNIST and EuroSAT. (1) MNIST: The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0–9), with 60,000 images for training and 10,000 for testing. Each image is 28x28 pixels in size. Due to its relatively small size and simplicity, MNIST is considered a great testbed for validating the basic functionality of machine learning algorithms [21]. (2) EuroSAT: The EuroSAT dataset contains 60,000 color images divided into 10 classes [22]. There are 50,000 training images and 10,000 testing images, with each image being 32×32 pixels in size. EuroSAT is more challenging than MNIST as it contains color channels and more complex object variations, thus, this will provide an in-depth understanding of how the FEDGE algorithm would handle real-world image data

We use a loss function to estimate the deviation between the predicted and true class label. The FEDGE model employs the sparse categorical cross-entropy loss function for multi-class classification problems [20]. Minimizing cross-entropy encourages the model to assign high probabilities to the correct classes and low probabilities to the incorrect classes. For an individual data point, the sparse categorical cross-entropy loss is calculated as follows (refer Eq. (7)):

$$L = -\log(p) \tag{7}$$

where p is the predicted probability of the correct class.

Stochastic Gradient Descent (SGD) is an optimization algorithm that is used in the training phase so that the loss function is minimized by the model. To do so, the model tunes parameters such as weights and bias [23]. In a traditional gradient descent algorithm, the weights are computed for the entire dataset, but in SGD, these gradients are calculated for a single data point at each iteration, thus making it extremely useful for large datasets. The mathematical expression is as follows (refer Eq. (8)):

$$\theta_{t+1} = \theta_t - \eta \nabla Q(\theta_t; x^{(i)}, y^{(i)}) \tag{8}$$

where θ represents the model parameters, η is the learning rate and $\nabla Q(\theta_t; x^{(i)}, y^{(i)})$ is the loss function gradient evaluated in the training example i^{th}.

We use precision, recall and F1 score metrics to assess the quality of classification performed by FEDGE algorithm [24]. Precision: Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. Recall (Sensitivity): Recall, also known as sensitivity or true positive rate, represents the proportion of correctly identified positive samples out of all actual

positive samples. F1-score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. Higher scores in all of these metrics indicate a well-performing classifier, while disparities between classes can highlight classification imbalances. The precision (refer Eq. 9), recall (refer Eq. 10), and F1-score (refer Eq. 11) are calculated using the following formulas:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

where: TP = True Positives (correctly predicted positive instances) ; FP = False Positives (incorrectly predicted positive instances) ; FN = False Negatives (incorrectly predicted negative instances)

4 Data simulation and client partitioning

In federated learning (FL), data is typically distributed across multiple clients in a non-identical and unbalanced manner, known as non-IID (non–independent and identically distributed) data. To accurately evaluate the robustness and scalability of the proposed FEDGE framework, it is essential to emulate this heterogeneity in controlled experimental settings. This section describes the methodology adopted to simulate realistic data distributions and partition datasets across satellite clients. Following the data simulation, a client partitioning strategy is implemented to assign specific subsets of the dataset to individual satellite nodes. Each client thus holds a localized dataset that reflects practical challenges such as data bias, limited sample size, and uneven class representation.

4.1 Simulating label-skewed/non-IID distributions via Dirichlet sampling

Dirichlet sampling is a widely used approach for generating heterogeneous label proportions across clients. Let $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_K\}$ denote the global label set, with $K = 10$ for MNIST and EuroSAT. The global class distribution $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]$ represents the fraction of samples per class. For each client j, local label proportions $\boldsymbol{\phi}^{(j)}$ are drawn from a Dirichlet distribution:

$$\boldsymbol{\phi}^{(j)} \sim \text{Dir}(\gamma \boldsymbol{\lambda}), \quad \forall j \in \{1, \ldots, N\} \tag{12}$$

The concentration parameter γ controls the similarity between local and global distributions. A high γ (e.g., 10^6) produces nearly IID partitions, while a low γ (e.g., 0.1) yields highly skewed, label-sparse datasets. As shown in Fig. 1, smaller γ values cause clients to hold samples from only a few classes, whereas larger values ensure balanced data.

For experiments, the MNIST dataset and EuroSAT dataset are partitioned using this mechanism. We employ the RGB subset of EuroSAT (3-band images) to align with standard vision benchmarks, leveraging its natural variability in terrain, lighting, and seasonal features to evaluate the robustness of FEDGE under heterogeneous, real-world conditions.

4.2 Client-specific data generation using feature-space clustering

To simulate heterogeneous client data distributions, we employ feature-space clustering based on pretrained deep neural embeddings. Images from datasets such as MNIST and EuroSAT are passed through a pretrained ResNet18 to extract high-dimensional feature vectors, which are then

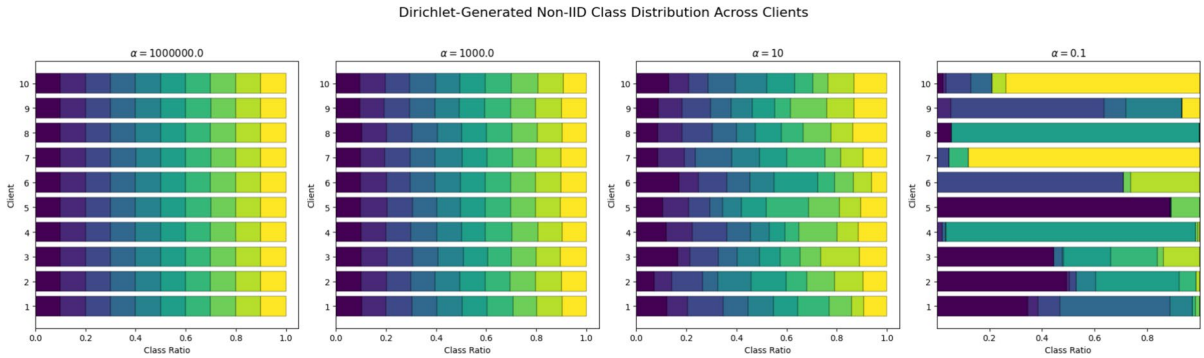


Fig. 1 Class distribution across $N = 10$ clients generated using Dirichlet sampling. High concentration ($\alpha = 10^6$) yields balanced partitions, while low concentration ($\alpha = 0.1$) produces highly skewed, label-sparse datasets, demonstrating increasing non-IIDness

projected to two dimensions using t-SNE for visualization of semantic groupings (Fig. 2). Let $\mathcal{C}$ denote the set of clusters in this feature space, with each cluster $c \in \mathcal{C}$ representing semantically similar samples.

We define a global cluster proportion vector $\boldsymbol{\theta}$ and sample a client-specific allocation vector $\boldsymbol{\pi}^{(i)}$ from a Dirichlet distribution:

$$\boldsymbol{\pi}^{(i)} \sim \text{Dirichlet}(\beta\boldsymbol{\theta}), \quad \forall i \in \{1, 2, \ldots, N\} \tag{13}$$

The concentration parameter β controls heterogeneity — smaller values yield more skewed (non-IID) client datasets, while larger values create near-uniform distributions. Each client receives data sampled according to $\boldsymbol{\pi}^{(i)}$, ensuring controlled variability in local feature distributions. This cluster-driven allocation mimics real-world personalization among edge devices or satellites, where local data differs due to geography or sensor specialization, thus providing a realistic evaluation of FEDGE under distributional heterogeneity.

4.3 Simulating edge-space conditions: intermittent connectivity and staleness

To capture the challenges of federated learning in edge and space-based environments, we simulate intermittent client availability caused by connectivity disruptions, limited bandwidth, or out-of-range conditions (e.g., satellite clients). In each communication round, a binary availability mask is generated using a Bernoulli distribution with a predefined drop probability p (e.g., 30%). A value of 1 represents an active client, while 0 indicates an unavailable or stale one. This mask is applied before model aggregation to emulate asynchronous participation. Each round, clients are independently included with probability $1 - p$, effectively modeling real-world conditions where clients may be offline, delayed, or out of range, as in satellite and remote edge networks. This setup reproduces temporal staleness and dynamic participation typical of real federated systems.

5 Results and discussions

Figure 3 illustrates the generalization capability of FEDGE, showing test loss and accuracy across ten clients over ten communication rounds. The test loss (left plot) decreases from 0.3515 in Round 1 to 0.2630 by Round 3 and stabilizes afterwards. Test accuracy (right plot) rises from 0.8987 to 0.92–0.93. Despite some client-specific variations, the upward trend confirms effective global model generalization.

Testing on the EuroSAT dataset, which presents complex aerial and satellite imagery, shows that FEDGE achieves a test accuracy of approximately 91% with a test loss of 0.6 (Fig. 4), confirming strong generalization in challenging scenarios.

Figure 5 shows ROC and Precision-Recall (PR) curves for the digit classification model. The ROC plots True Positive Rate versus False Positive Rate (AUC-ROC indicates class separability), and the PR curve reflects positive identification performance on imbalanced datasets. For the ten-class EuroSAT dataset, performance was evaluated over ten rounds using a one-vs-rest strategy with one-hot labels and softmax outputs. Curves are shown for three representative classes (0, 3, 7) across early (Round 1), mid (Round 5), and

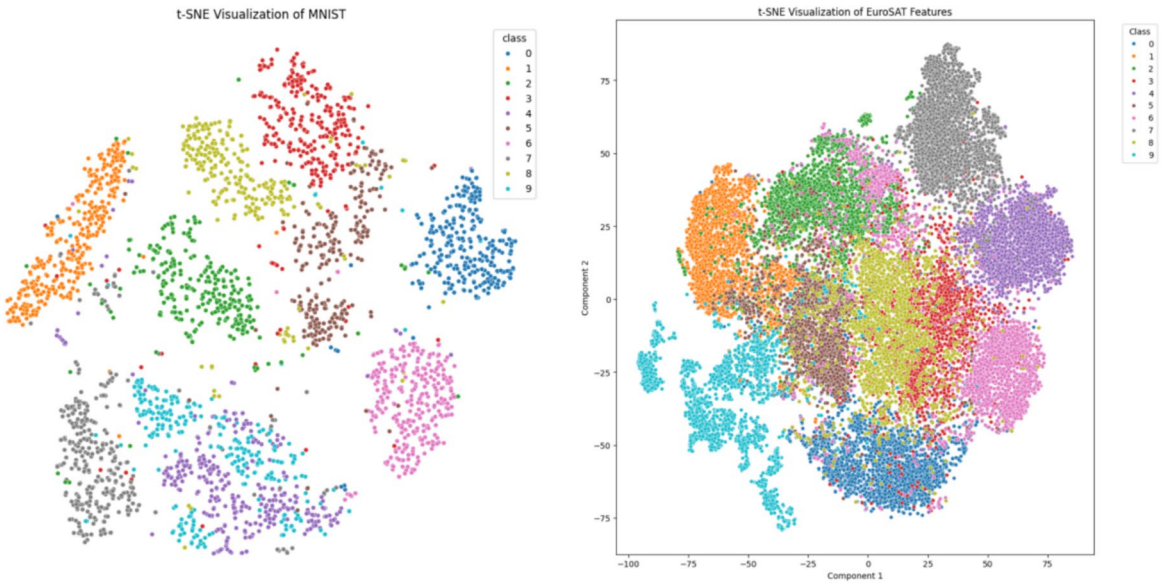


Fig. 2 t-SNE visualization of feature clusters from pretrained embeddings. Each cluster groups semantically related samples used for client-specific data allocation via Dirichlet sampling, where β controls the degree of non-IIDness

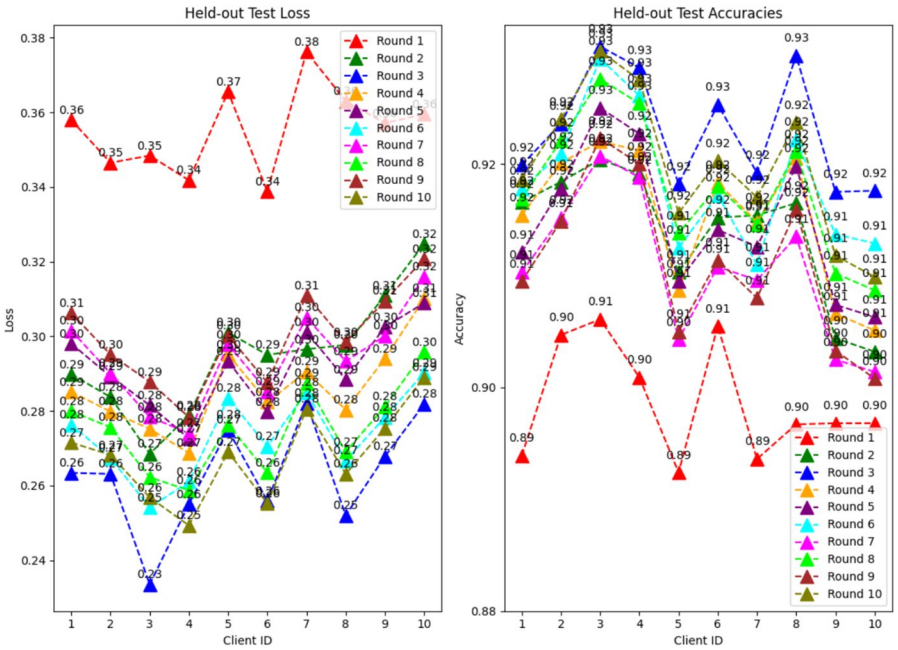


Fig. 3 Test loss and accuracy across clients on MNIST. Left: test loss (lower is better). Right: test accuracy (higher is better)

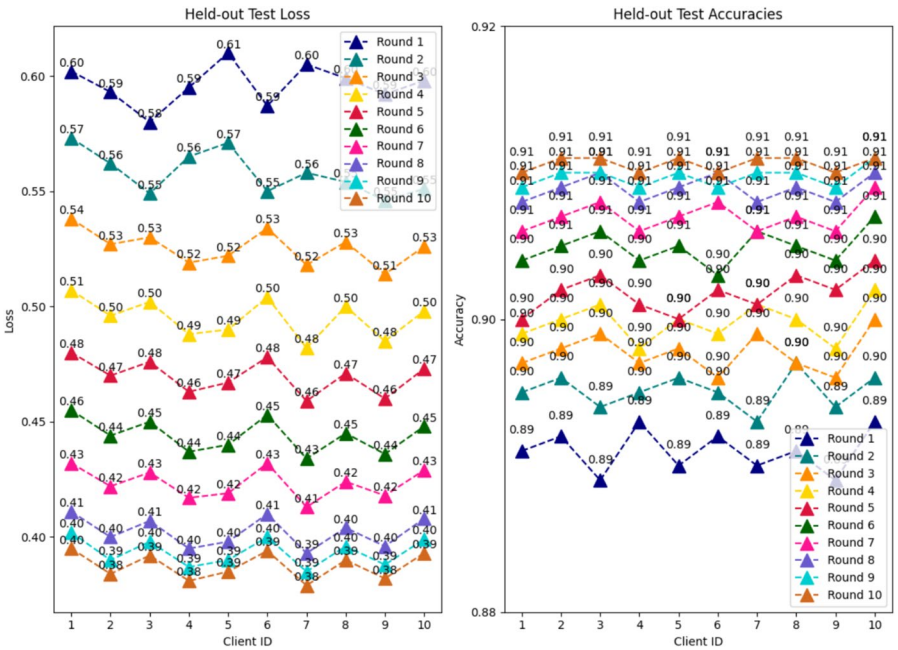


Fig. 4 Test loss and accuracy across clients on EuroSAT. Left: test loss (lower is better). Right: test accuracy (higher is better)

late (Round 10) rounds. Increasing areas under both curves indicate improved discrimination and calibration, with later rounds showing higher TPR and precision at lower FPR and higher recall.

Figure 6 illustrates FEDGE performance with and without gradient quantization on EuroSAT. Quantization reduces gradient precision for transmission from edge devices to the central server, using uniform or non-uniform methods, significantly lowering network latency [25]. The left subplot shows global test accuracy over rounds for quantized (blue) versus full-precision (red) gradients, and the right subplot shows corresponding loss. Quantization reduces data exchange by $\sim 14\times$ (170,552 $\rightarrow$ 12,182 bytes) with only a $\sim 5\%$ accuracy drop and a slight loss increase (~ 0.2), achieving a practical trade-off for real-time deep learning in resource-constrained edge environments . The performance of the proposed FEDGE is compared with the state-of-the-art works and tabulated in (Table 3).

5.1 Algorithm complexity analysis

The computational time and memory requirements of the FEDGE algorithm are analyzed using asymptotic notations. Time complexity, expressed in Big O notation, indicates the

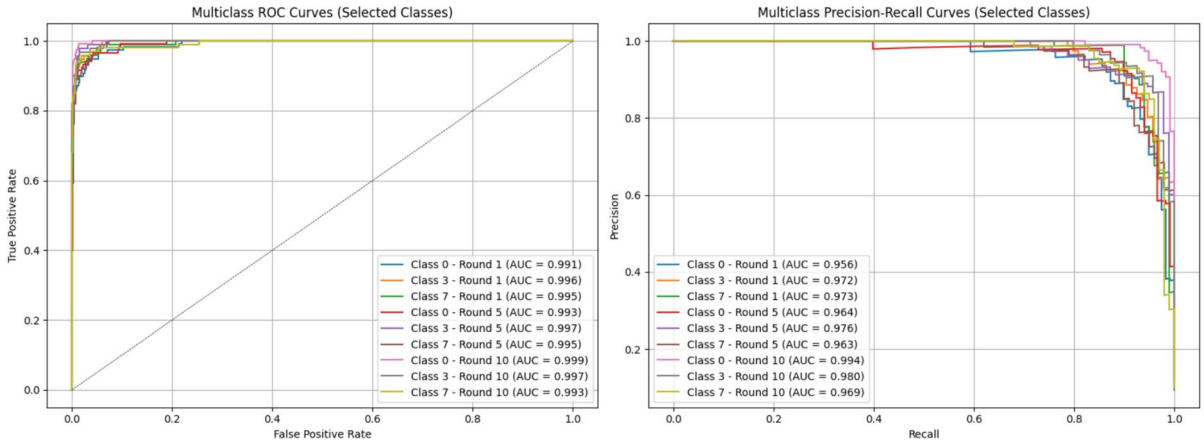
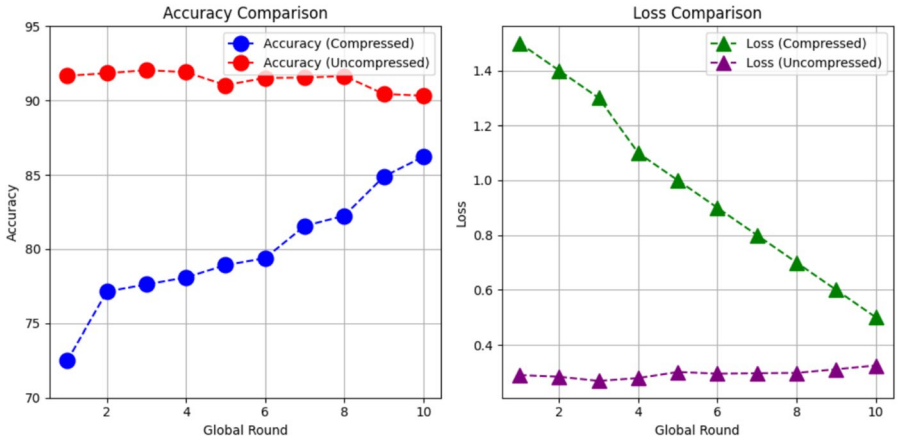


Fig. 5 ROC and precision-recall curves for EuroSAT dataset

Fig. 6 Gradient quantization for compressing data



algorithm's execution time as a function of input size (e.g., $O(n)$ for linear, $O(n^2)$ for quadratic growth), while space complexity reflects the memory required to store inputs and intermediate data [28].

5.1.1 Global model synchronization algorithm

The time complexity is computed in three steps. First, the Ground Station (GS) receives updates from n satellites, which takes $O(n)$. Second, the GS checks the condition $a_i = 1$ to decide whether to update the model, taking $O(1)$. Third, the GS broadcasts the updated global model to all satellites, again $O(n)$. Thus, a single synchronization round has (refer Eq. (14))

$$T_{\text{sync}} = O(n) + O(1) + O(n) = O(n), \tag{14}$$

showing that synchronization time scales linearly with the number of satellites.

The space complexity includes storing gradients and metadata from all satellites, $O(n \cdot m)$, and maintaining the global model, $O(k)$. Since $k \ll n \cdot m$, the total space complexity simplifies to (refer Eq. (15))

$$S_{\text{sync}} = O(n \cdot m) + O(k) \approx O(n \cdot m). \tag{15}$$

5.1.2 Edge-based FL algorithm

For the Edge-based FL algorithm, the time complexity consists of receiving the global model from the GS, $O(k)$, preprocessing the local dataset of size d_k, $O(d_k)$, training the local model with E epochs, $O(E \cdot d_k)$, and uploading gradients, $O(m)$. Therefore, the overall time complexity is (refer Eq. 16)

$$T_{\text{sat}} = O(k) + O(d_k) + O(E \cdot d_k) + O(m) = O(E \cdot d_k), \tag{16}$$

with E epochs and local dataset size d_k.

The space complexity includes storing the global model $O(k)$, the local dataset $O(d_k)$, and the gradients $O(m)$. As d_k typically dominates, the total space complexity simplifies to (refer Eq. (17))

$$S_{\text{sat}} = O(k) + O(d_k) + O(m) \approx O(d_k). \tag{17}$$

Table 3 Performance comparison of FEDGE with state-of-the-art federated learning frameworks

Method	Dataset	Accuracy (%)	Loss	Key features
FedProx [12]	FEMNIST	78–80	1.1–1.3	Handles system heterogeneity
FedAsync [13]	CIFAR-10	78–80	–	Asynchronous aggregation
Optimal Client Sampling [15]	CIFAR-10	92–95 (val.)	0.15–0.25	Optimized client selection strategy
Hybrid privacy-preserving FL [16]	MNIST	∼90	–	Privacy preservation with differential privacy
	Nursery (UCI)	∼80 (F1)		
FedNova [26]	CIFAR-10	82–85	0.45–0.55	Normalized averaging for heterogeneous local updates
SCAFFOLD [27]	FEMNIST	81–83	0.8–1.0	Variance reduction via control variates
FEDGE (Ours)	MNIST	94.33 (train)	0.21 (train)	Edge-enhanced GS protocol gradient compression (14 × reduction) satellite metadata-driven scheduling resource-constrained deployment
		90.05 (test)	0.24 (test)	
	EuroSAT	93.47 (train)	0.18 (train)	
		91.51 (test)	0.21 (test)	

5.2 Discussions

This study introduces FEDGE, a federated learning framework for real-time DNN deployment in edge environments with limited connectivity, such as satellite networks. The architecture leverages on-device processing and gradient compression to reduce communication overhead while maintaining high accuracy, demonstrating scalability, resilience to disconnections, and suitability for satellite-based federated learning applications.

Challenges and future scope: (i) Enhancing the scheduler in Algorithm 1 is a key area for improvement. Dynamically adjusting the a_i condition based on network status, satellite resources, and model performance could optimize efficiency and communication costs. Addressing non-IID data across satellites is also essential; techniques such as standardization, normalization, or domain adaptation in edge processing (Step 2 of Algorithm 2) could improve performance. (ii) Model staleness and synchronization remain challenges. While asynchronous updates help maintain training despite interruptions, they may lead to inconsistent models. Adaptive learning rates, dynamic aggregation, and staleness-aware gradient methods could improve stability and convergence. (iii) Resource-constrained optimization is critical. Although ASIC integration is acknowledged, this study focuses on simulations. Future work should explore lightweight architectures, gradient sparsification, compression, and efficient training for feasible deployment on satellites and edge devices. (iv) Security and privacy require stronger guarantees. Beyond inherent FL protections, mechanisms such as differential privacy or homomorphic encryption should be incorporated. Real-world deployment will also need to handle communication delays, intermittent connectivity, and dynamic topologies. (v) Finally, FEDGE could support advanced training paradigms, including real-time fine-tuning of pre-trained LLMs on space-generated data, enabling applications such as autonomous anomaly detection, predictive maintenance, and intelligent resource allocation. FEDGE could also foster collaborative research across distributed spacecraft datasets, accelerating discovery and providing new insights into space environments.

6 Conclusion

This research proposes FEDGE, a federated learning architecture for efficient and robust DNN deployment in resource-constrained edge environments, particularly satellite networks, with potential AI ASIC integration for enhanced performance. Key enhancements include on-satellite data pre-processing and gradient compression, reducing computational load on the ground station and optimizing bandwidth usage. Gradient compression techniques, such as sparsification or quantization, minimize communication overhead, enabling more frequent model updates in bandwidth-limited scenarios. Metadata sharing and a dynamic scheduler allow satellites to transmit information (e.g., training accuracy, data statistics) to the ground station, guiding aggregation decisions and prioritizing high-quality updates, thereby improving convergence speed and resource efficiency. FEDGE also supports scalability by handling distributed preprocessing and training at the edge, enabling robust federated learning across complex, distributed environments. This design aligns with the trend of leveraging satellite-edge computing for in-orbit ML training on massive, distributed datasets. Addressing the challenges outlined in the discussion can further enhance FEDGE's performance, robustness, and security, paving the way for applications in satellite observation systems, LiDAR scanning, autonomous vehicle networks, and embedded systems.

Data availability The datasets used in this study are 1) the MNIST database of handwritten data and 2) EuroSAT, labeled tiny images,

are freely available online at https://archive.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits [21], and https://github.com/phelber/eurosat [22]. The trained model developed in this study can be shared upon reasonable request. Interested researchers may contact the corresponding author via email for access to the model and related resources.

Declarations

Conflict of interest The authors declare no conflict of interest for the present research study.

References

1. Sanjay C, Jahnavi K, Karanth S et al (2024) A secured deep learning based smart home automation system. Int J Inf Technol 16(8):5239–5245
2. Kagi S, M A, Kousalya CG et al (2025) Federated deep reinforcement learning (fdrl) framework using pelican optimization (po) to achieve sustainable energy in IoT-integrated wireless networks. Int J Inf Technol https://doi.org/10.1007/s41870-025-02752-1 (received: 14 April 2025; Accepted: 21 Sep 2025; Published: 24 Oct 2025)
3. Bonawitz K, Eichner H, Grieskamp W, Harchaoui Z, Krieger H, Nardi I, Rahman R, Tran K, Wegner K, Simpson K (2019) Towards federated learning at scale: system design. In: Proceeding of 2019 ACM SIGSAC Conf Cloud Comput Sec Workshop, pp 1–16
4. Srivastava V, Lamba V, Mathada VS, Bulla C, Gupta N, Veeramanikandan P et al (2025) An IoT-based framework employing fuzzy logic and federated learning for decentralized decision-making. Int J Inf Tech 1–7
5. Chouhan B, Pai R, Pandey B (2025) Edge computing based emulator design for low-latency IoT health monitoring system. Int J Inf Tech 1–11
6. Sun T, Li D, Wang B (2022) Decentralized federated averaging. IEEE Trans Pattern Anal Mach Intell 45(4):4289–4301
7. Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V (2018) Federated learning with non-iid data. arXiv:1806.00582
8. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proceeding of ACM SIGSAC Conference on Computer Communications Section 2017, pp 1175–1191
9. Hard A, Rao K, Mathews R, Ramaswamy S, Beaufays F, Augenstein S, Eichner H, Kiddon C, Ramage D (2018) Federated learning for mobile keyboard prediction. arXiv:1811.03604
10. Jin C, Chen X, Gu Y, Li Q (2023) Feddyn: a dynamic and efficient federated distillation approach on recommender system. In: 2022 IEEE 28th The International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 786–793
11. Ghosh A, Song Q, Gupta P, Khudanpur S (2020) Communication-efficient federated learning with quantization and sparsification. arXiv:2003.13218
12. Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
13. Xie C, Koyejo S, Gupta I (2019) Asynchronous federated optimization. arXiv:1903.03934
14. Alistarh D, Grubic D, Li J, Tomioka R, Vojnovic M (2017) Qsgd: communication-efficient sgd via gradient quantization. In: Advances in neural information processing systems, pp 1709–1720
15. Chen W, Horvath S, Richtarik P (2020) Optimal client sampling for federated learning. arXiv:2010.13723
16. Truex S, Baracaldo N, Anwar A, Steinke T, Ludwig H, Zhang R, Zhou Y (2019) A hybrid approach to privacy-preserving federated learning. In: Proceeding of 12th ACM workshop Artificial Intelligence Security, pp 1–11
17. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J (2019) Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc IEEE 107(8):1738–1762
18. El-Sayed H, Sankar S, Prasad M, Puthal D, Gupta A, Mohanty M, Lin C-T (2017) Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing environment. IEEE Access 6:1706–1717
19. Xu J, Glicksberg BS, Su C, Walker P, Bian J, Wang F (2021) Federated learning for healthcare informatics. J Healthcare Inf Res 5:1–19
20. Goodfellow I (2016) Deep learning
21. LeCun Y, Cortes C, Burges CJ (2010) Mnist handwritten digit database. ATT Labs. Available: http://yann.lecun.com/exdb/mnist 2
22. Helber P, Bischke B, Dengel A, Borth D(2019) Eurosat: a novel dataset and deep learning benchmark for land use and land cover classification. IEEE J Sel Top Appl Earth Obs Remote Sens
23. Sun R-Y (2020) Optimization for deep learning: an overview. J Oper Res Soc China 8(2):249–294
24. Powers DM (2011) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. J Mach Learn Technol 2(1):37–63
25. Gray RM, Neuhoff DL (1998) Quantization. IEEE Trans Inf Theory 44(6):2325–2383
26. Wang J, Liu Q, Liang H, Joshi G, Poor HV (2020) Tackling the objective inconsistency problem in heterogeneous federated optimization. Adv Neural Inf Proc Syst 33:7611–7623
27. Karimireddy SP, Kale S, Mohri M, Reddi S, Stich S, Suresh AT (2020) Scaffold: stochastic controlled averaging for federated learning. In: The International Conference on Machine Learning. PMLR, pp 5132–5143
28. Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to algorithms. MIT press