Arabian Journal for Science and Engineering
https://doi.org/10.1007/s13369-025-10533-y

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Optimizing Federated Learning for Medical Images Analysis Through Traffic Efficiency

Lamees Rababa[1] · Omar AlZoubi[1,2] · Muneer Bani Yassein[1]

Received: 15 August 2024 / Accepted: 27 July 2025

Abstract
Federated learning (FL) is an innovative approach to machine learning that emphasizes data privacy by allowing model training on decentralized data sources. This study focuses on the application of FL using medical data, specifically chest X-ray images. We address the issue of traffic congestion between the server and clients by implementing three compression techniques: Pruning, Clustering, Quantization, and a hybrid technique that integrates these three methods which we name it (QPC). Our results demonstrate that clustering is the most effective compression technique, achieving superior upload and download speeds while preserving accuracy and significantly reducing traffic between the central server and clients. The average upload speed, download speed, and accuracy achieved were approximately 29.71 GB/s, 367.86 GB/s, and 97.53%, respectively. This research advances the field of FL by improving scalability and efficiency, offering valuable insights for future research in Federated Learning and healthcare applications.

Keywords Federated learning · Convolutional neural network · Machine learning · Chest radiograph · Internet of Things

1 Introduction

Federated learning (FL), an emerging subject in modern machine learning paradigms, has become a critical framework for addressing privacy issues, data decentralization, and model collaboration. As the exponential rise of digital data continues to reshape the information landscape, classical machine learning models face significant challenges in balancing model performance with individual data privacy. Differentiated by its decentralized training approach, FL seeks to bridge these divergent objectives by enabling collaborative model training across remote devices without centralizing raw data. This paradigm shift results in a harmonious convergence of machine learning improvements and privacy preservation, ushering in a new era in the pursuit of intelligent algorithms within the complex fabric of modern data. This research supports a thorough exploration and understanding of federated learning (FL) through its integration with medical data, aiming to address a critical challenge: traffic congestion between the server and the client. This effort holds promise for an advanced future across various domains, particularly in healthcare.

Medical imaging, including high-resolution methods like computed tomography (CT) and magnetic resonance imaging (MRI), is one of the most important approaches for detecting and treating medical problems. Deep learning for medical image processing and analysis could become crucial for delivering contemporary, comprehensive healthcare. For example, convolutional neural networks (CNNs) have demonstrated remarkable performance in several medical image processing tasks, such as detection, classification, and segmentation [1]. These capabilities for data learning and modeling are often housed in cloud servers or data centers. However, the performance of deep learning models is greatly impacted by the unavailability and high cost of large-scale annotated medical image datasets, primarily due to the difficulty of transferring sensitive data that contain patient information. Federated learning (FL) has emerged as a technique for protecting sensitive data. It is a novel method of training intelligent models that ensures data privacy without requiring the transmission of sensitive information to exter-

✉ Lamees Rababa
lwrababa21@cit.just.edu.jo

Omar AlZoubi
oaalzoubi@just.edu.jo

Muneer Bani Yassein
masadeh@just.edu.jo

[1] Jordan University of Science and Technology, Irbid, Jordan

[2] Effat University, Jeddah, Kingdom of Saudi Arabia

Published online: 21 August 2025

nal sites [2, 3]. FL aims to achieve a delicate balance between the need to analyze vast datasets and the paramount importance of maintaining user privacy. Technically, FL represents a collaborative artificial intelligence method that manages the training of models by linking multiple clients with a central server, all without the need to share raw datasets [4].

Against the backdrop of the proliferation of interconnected devices and the remarkable surge in data generation, machine learning grapples with unprecedented opportunities and intricate challenges. Within this context, federated learning (FL) emerges as a robust model uniquely equipped to tackle the complexities arising from the distributed nature of data across diverse devices while concurrently harnessing collective intelligence for model training. Diverging from traditional centralized machine learning methodologies, FL empowers the training of models directly on decentralized devices, such as smartphones, terminals, and Internet of Things (IoT) devices. This decentralized training paradigm not only upholds the sanctity of data privacy by retaining sensitive information locally but also capitalizes on the rich diversity of data sources to enhance model performance. However, as FL gains traction, it presents complex challenges, among which effective traffic management takes center stage. The decentralized nature of FL requires the exchange of model updates between the central server and participating devices. This communication overhead incurred during these exchanges can become a significant bottleneck, especially in scenarios with bandwidth constraints or resource-constrained devices. As FL systems expand to accommodate more extensive and diverse networks of devices, the need to improve communication efficiency becomes increasingly apparent.

This research explores one of the challenges in federated learning (FL): the issue of high communication traffic. An FL model transmits model parameters, updates, and collected information across the network. The overwhelming volume of these connections can become prohibitive, hindering the scalability and applicability of FL systems in the real world. Therefore, this study seeks to investigate innovative solutions to alleviate this problem, enhancing the feasibility and effectiveness of FL. Motivated by the necessity of simplifying communication within standardized learning frameworks, this research focuses on integrating compression techniques, including quantization, pruning, and clustering, which have successfully reduced the size of machine learning models without compromising their predictive capabilities. Leveraging these compression techniques in FL offers a compelling way to mitigate communication challenges by conveying more concise model representations while maintaining the integrity of shared knowledge across the decentralized network. This study aspires to contribute new insights and practical strategies that can advance FL toward enhanced scalability, improved efficiency, and sustainable relevance in the era of distributed intelligence.

The remainder of this research is presented as follows. Section 2 discusses the related work about FL and compression techniques. Section 3 briefly introduces relevant information about the FL, compression techniques, and CNNs. Section 4 describes the proposed FL implemented with the Flower framework and outlines how it reduces traffic between the central server and clients. Section 5 provides and discusses our experimental results. Finally, Sect. 6 provides our conclusions.

2 Literature Review

The term federated learning was first coined by researchers from Google [5]. The authors noted that within the FL framework, the model undergoes training across many decentralized terminal devices, such as smartphones and Internet of Things (IoT) devices. Importantly, model updates are integrated without exchanging raw data. Furthermore, FL addresses privacy concerns by keeping data local, thus facilitating collaborative model training. This chapter discusses related work on the architecture of FL and its associated challenges.

2.1 Federated Learning in Medical Data Analysis

Several studies explored the use of FL in medical imaging analysis, employing diverse methodologies and frameworks. For example, Da Silva et al. [6] utilized the Flower framework and FedAvg strategy to build a FL environment for pneumonia detection in chest X-ray images, achieving a notable accuracy of 98.46% with ResNet-18 in a simulated setup involving eight hospitals. Moreover, Zheng Li et al. [7] introduced FedFocus, an FL framework for COVID-19 detection on chest X-ray images, enhancing training efficiency and accuracy through dynamic parameter aggregation weights based on individual model training loss. Real-world experiments with population-divided training sets demonstrated superior performance over baseline models.

In a collaborative initiative led by Reina et al. [8], the viability of FL in medical imaging analysis was investigated, with the FL-health project utilizing advanced deep learning algorithms and Intel Xeon Scalable processors. Initial experiments demonstrated FL achieving a commendable training accuracy of up to 90%, highlighting its promising potential in healthcare applications. Additionally, in recent research, Wang et al. [9] introduced FRESH, an innovative healthcare framework utilizing FL with ring signature defense against Source Inference Attacks (SIAs). FRESH incorporates wearable devices for local data collection, edge computing for processing, and central server-based FL for joint model

training. Implementing certificate ring signatures enhances privacy by concealing update sources, demonstrating effectiveness in reducing SIA success rates. Similarly, Hossen's work [10] focused on skin disease classification, employing a bespoke dataset and a CNN model with FL. The approach, implemented across diverse client counts, showcased average accuracies of 81.21%, 86.57%, 91.15%, and 94.15%, providing an innovative and secure solution to human skin disease classification.

The performance issues of FL were investigated by Huang et al. [11], who tackled challenges in applying centralized machine learning algorithms to decentralized and non-identically independently distributed (non-IID) electronic medical records (EMRs). They proposed the Community-Based FL (CBFL) algorithm, which clusters non-IID intensive care unit (ICU) EMR data into clinically meaningful communities, achieving superior performance compared to a baseline FL algorithm. Similarly, Islam et al. [12] addressed the challenge of centralized data collection by implementing FL for brain tumor identification in MRI images, achieving an accuracy of 91.05% with an ensemble FL model while ensuring privacy-protected tumor classification.

The data privacy challenge was approached by several researchers, including Cui et al. [13], who introduced the "FeARH" algorithm, a privacy-preserving variation of FL, to address the challenges of centralizing electronic medical records. Experimental results demonstrated comparable performance to both centralized machine learning and traditional FL approaches. Similarly, Al Aziz et al. [14] expanded a genomic data privacy solution that was initially ranked third in iDash 2020. When tested with two machine learning algorithms, it achieved a high AUC of 0.89 in just 8 s with a privacy budget of 5, showcasing its potential applicability to diverse genomic datasets and machine learning algorithms.

2.2 Traffic Efficiency in Federated Learning

In the domain of improving traffic efficiency in federated learning (FL), Cui et al. have made numerous contributions [15–17]. They introduced the Model Update Compression by Soft Clustering (MUCSC) algorithm, which minimizes communication traffic volume in FL networks by transmitting only cluster centroids and cluster IDs. An enhanced version, B-MUCSC, addresses scenarios with limited network resources, achieving a high compression rate by grouping insignificant model updates. A systematic investigation of the trade-off between compression error and model accuracy in FL was conducted, leading to the introduction of an adaptive framework that strategically adjusts compression rates for optimal accuracy. Their work also explored BCFL, a blockchain-based communication-efficient FL framework, demonstrating efficiency through substantial reductions in communication traffic and training time on the CIFAR-10 and FEMNIST datasets. Additionally, they examined optimal transmission strategies in FL over lossy Internet conditions with the introduction of FedLC. This approach, which incorporates model compression, forward error correction (FEC), and retransmission, significantly improved model accuracy or reduced communication traffic compared to state-of-the-art baselines in experiments on public datasets. These studies collectively provide valuable insights into addressing traffic-related challenges and optimizing efficiency in practical FL scenarios. Table 1 summarizes the studies discussed in terms of their focus, contributions, and research problems addressed.

To overcome the limitations of federated learning (FL) in medical image analysis, an enhanced approach focuses on improving communication efficiency and reducing resource consumption while maintaining model accuracy. This approach employs strategies for compressing model updates and minimizing transmitted data, which enhances scalability and makes FL more suitable for resource-constrained medical systems. Additionally, dynamic parameter aggregation optimizes data consumption and boosts efficiency. FL ensures medical data privacy by enabling training without direct access to the data, thereby preserving confidentiality. Moreover, an intelligent compression mechanism dynamically balances data size and model accuracy based on network conditions, minimizing data loss and ensuring stable training. Finally, adaptive transmission strategies reduce latency and enhance connection stability in high-data-loss environments. These improvements make FL more efficient and reliable for real-time medical applications, particularly in settings with strict resource and privacy constraints.

3 Background

This chapter explores the fundamental concepts of federated learning (FL), convolutional neural networks (CNN), and compression techniques, with a specific focus on critical compression strategies: pruning, quantization, and clustering. Understanding these elements is essential for appreciating the challenges and opportunities in modern machine learning applications.

3.1 Federated Learning

FL is a decentralized machine learning approach that allows model training across various devices or servers while keeping the data local. Traditional machine learning algorithms rely on centralized data repositories, which raises privacy and security concerns [5]. As shown in Fig. 1, FL enables models to be trained on data distributed across devices without the need to exchange raw data.

Table 1 Summary of related work in federated learning

Study	Focus	Key contributions	Gaps addressed
Da Silva et al. (2023) [6]	Pneumonia detection in chest X-ray images	Utilized Flower framework and FedAvg strategy, achieving 98.46% accuracy with eight hospitals	Limited scalability of FL in medical contexts; real-world hospital data integration
Zheng Li et al. (2022) [7]	COVID-19 detection using FL for chest X-ray images	Introduced dynamic parameter aggregation for improved efficiency and accuracy in FL	Lack of model efficiency due to non-adaptive parameter aggregation
Islam et al. (2023) [12]	Brain tumor identification using FL on MRI images	Proposed a privacy-preserving ensemble FL model achieving 91.05% accuracy	Centralized data collection limitations in FL
Hossen (2023) [10]	Skin disease classification using FL	Introduced a CNN model for skin disease classification across multiple client setups	Lack of generalizability across diverse skin conditions and environments
Reina et al. (2022) [8]	FL in medical imaging	Demonstrated up to 90% training accuracy for healthcare with Deep Learning and Intel Xeon processors	Need for integration of advanced FL techniques in real-time healthcare applications
Wang et al. (2023) [9]	FL with ring signature defense against SIAs in healthcare	Implemented FRESH for enhancing privacy with wearable devices and edge computing	Privacy concerns in healthcare FL with Source Inference Attacks (SIA)
Huang et al. (2019) [11]	Community-Based FL for non-IID EMR data	Proposed CBFL to address non-IID challenges in ICU EMR data, improving performance over baseline FL	Challenges with centralized machine learning for non-IID datasets
Cui et al. (2021) [13]	Privacy-preserving FL for centralized medical records	Introduced FeARH algorithm for privacy in FL, showing comparable performance to traditional FL	Data centralization challenges and privacy concerns in medical FL
Al Aziz et al. (2022) [14]	Genomic data privacy solution in FL	Achieved high AUC (0.89) with privacy-preserving techniques for genomic data	Lack of fast and efficient methods for privacy in genomic datasets
Cui et al. (2021) [15]	Traffic efficiency in FL	Introduced Model Update Compression by Soft Clustering (MUCSC) to minimize traffic	Reducing communication volume in bandwidth-limited FL networks
Cui et al. (2022) [16]	Tradeoff between compression and model accuracy	Developed an adaptive framework for optimal compression	Balancing compression errors with accuracy in FL scenarios
Cui et al. (2023) [17]	Optimal transmission strategies for FL	Proposed FedLC to optimize model transmission under lossy conditions	Overcoming communication inefficiency in lossy FL networks

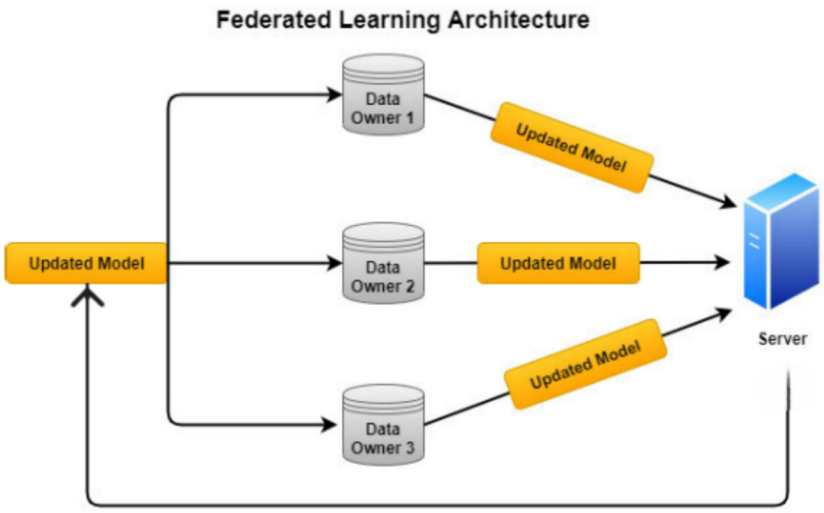


Fig. 1 General FL architecture [18]

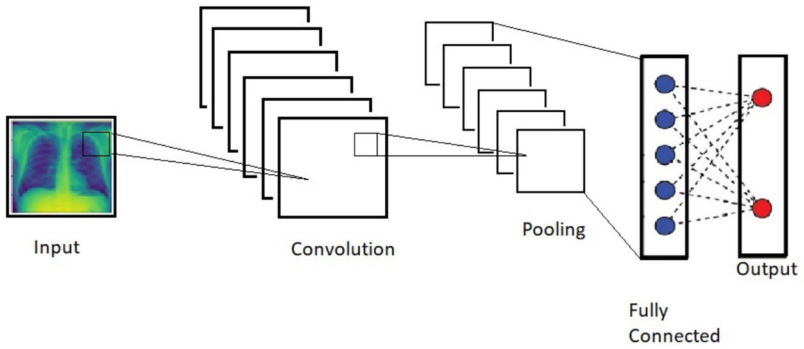


Fig. 2 CNN architecture

Let us consider a simple FL scenario with a global model denoted as W and N participating clients (devices or servers). The process can be outlined as follows:

Step 1: Initialization: Initialize a global model W with random parameters or pretrained weights.

Step 2: Client Update: Each client i performs local training on its data and computes a model update denoted as W_i'. This is the gradient or change to the model weights based on the local data.

$$W_i' = \text{LocalTraining}(W, \text{Data}_i) \tag{1}$$

Step 3: Aggregation: The model updates from all clients are aggregated to obtain a new global model W'. This aggregation can be performed using various methods, such as averaging [5]:

$$W' = \frac{1}{N} \sum_{i=1}^{N} W_i' \tag{2}$$

Step 4: Update Global Model: The new global model W' becomes the updated model for the next round of training.

Step 5: Steps from 2 to 4 are repeated for a predefined number of rounds or until convergence.

3.2 Key Principles of FL

This section explains several key principles of federated learning (FL). FL employs a decentralized training approach, training models on data distributed across multiple devices rather than centralizing it [19]. It ensures privacy-preserving measures by keeping data local and sharing only model updates, allowing learning from diverse datasets without compromising user information [20]. FL effectively handles heterogeneous and non-IID data, which is common when users have varied data distributions [21]. The process involves model aggregation, where updates from local models are combined using methods like averaging to create a global model [19]. This typically follows an iterative process, continuously refining the global model until desired performance is achieved [19]. To minimize communication costs, only model updates are exchanged, enhancing communication efficiency, especially with large datasets or limited bandwidth [22]. Security is addressed through security considerations, implementing encryption and secure aggregation during communication [20]. FL systems are designed to be robust to device failures, employing strategies like redundancy to maintain stability despite device dropouts [23]. Lastly, considerations such as behavioral trust

mechanisms can help ensure model integrity by detecting and removing malicious devices, while incentives encourage honest contributions to the training process [24, 25].

3.3 Convolutional Neural Networks

CNNs have revolutionized the field of computer vision and image processing. Inspired by the human visual system, CNNs excel at extracting hierarchical features from input data. They are particularly effective in tasks such as image recognition, object detection, and segmentation [26]. Figure 2 illustrates the architecture of a convolutional neural network (CNN), a type of deep learning model commonly used for image recognition and processing tasks. The architecture comprises several key components:

Key components of convolutional neural networks (CNNs) involve several crucial layers. Convolutional Layers apply filters or kernels to the input data, capturing spatial hierarchies and allowing the network to recognize patterns at various scales, making them effective for understanding complex visual features. Pooling Layers downsample the spatial dimensions of the data, reducing computational complexity while preserving essential information, with common operations including max pooling and average pooling.

Fully Connected Layers process the flattened representations of the extracted features, enabling the network to make predictions based on learned features and are typically located at the end of CNN architectures. Finally, the Output Layer provides the final output, which may be a single value for regression tasks or multiple values for classification tasks. In Fig. 2, the output layer indicates a binary or multi-class classification, as suggested by the presence of two distinct colored nodes.

3.4 Compression Techniques

Compression techniques are crucial in optimizing deployment models, especially in resource-constrained environments. These techniques aim to reduce model size, memory footprint, and computational requirements while preserving performance. Compression techniques involve the following:

Pruning Pruning involves removing redundant or less essential weights from a model, thereby reducing its size and computational requirements. Various pruning methods exist, such as magnitude-based pruning and structured pruning [27].

Mathematically, the pruning operation for weight pruning can be represented as:

$$\text{Prune}(W, p) = \text{Top}_{1-p\%}(|W|) \tag{3}$$

where:

- W: Represents the weight matrix.
- p: The pruning percentage.
- $\text{Top}_{1-p\%}(|W|)$: Retains the top $(1-p)\%$ of the weights according to their absolute values.

Quantization Quantization involves reducing the precision of model parameters, typically from 32-bit floating-point numbers to lower-bit representations. This significantly reduces memory requirements and accelerates inference [27].

The general formula for quantization:

$$Q(x) = \text{Round}\left(\frac{x}{\text{Scale}}\right) \times \text{Scale} \tag{4}$$

where:

- x is the original floating-point value (e.g., weight or activation).
- Round is the rounding operation.
- Scale is a scaling factor that is often determined during the quantization process.

In this formula, the value x is divided by the scale, rounded to the nearest integer, and then multiplied by the scale to obtain the quantized value $Q(x)$.

Clustering lustering is a compression technique that groups similar weights, allowing the model to represent them with a single centroid value. This reduces the number of unique weights in the model and, consequently, the overall size of the model [28].

Here is a simplified representation of the K-means clustering algorithm:

Step 1: Initialization

- Choose the number of clusters k.
- Randomly initialize k cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$.

Step 2: Assignment

$$\text{Cluster}(\mathbf{x}_i) = \arg\min_j \ \text{Distance}(\mathbf{x}_i, \boldsymbol{\mu}_j) \tag{5}$$

Step 3: Update

$$\boldsymbol{\mu}_j = \frac{1}{\text{Number of points in cluster } j} \sum_{\mathbf{x} \text{ in cluster } j} \mathbf{x} \tag{6}$$

Step 4:

- Repeat steps 2 and 3 until convergence (when centroids do not change significantly or a maximum number of iterations is reached).

The distance function Distance($\mathbf{x}_i, \boldsymbol{\mu}_j$) can vary, but it is often the Euclidean Distance.

In terms of formulas, the distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ in Euclidean space is given by:

$$\text{Distance}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{7}$$

Table 2 presents a comparison of three model optimization techniques: pruning, quantization, and clustering. Pruning reduces model size and computational cost by removing unnecessary weights or neurons, potentially leading to a slight loss in accuracy. Quantization decreases the precision of weights by mapping them to a smaller set of values, which reduces both model size and inference costs, although it may result in some accuracy loss. Clustering groups similar data points, moderately reducing model size and potentially lowering computational costs during training, with a moderate impact on accuracy.

The features in Table 2 were chosen to provide a comprehensive comparison of these techniques, covering their definitions, objectives, processes, effects on model size and accuracy, computational costs, and common use cases. These features emphasize the strengths and trade-offs of each method, highlighting their roles in optimizing models, reducing computational overhead, and their practical applications in areas such as model and data compression, as well as image segmentation.

4 Methodology

4.1 Proposed Approach

In this section, we discuss the proposed approach configured for two clients engaging in collaborative model training over five rounds. The CNN model architecture is detailed, specifying layers, activation functions, and optimization metrics, including measures to prevent overfitting. The pruning techniques involved reducing model parameters by 50%, while clustering employed eight clusters with a focus on preserving sparsity patterns. Quantization, a process that significantly reduce model size, is explained, emphasizing the shift from 32-bit floating-point precision to 8-bit integers to minimize storage requirements. These components contribute to a comprehensive understanding of the machine learning workflow, encompassing collaborative training, model architecture, and optimization techniques.

4.2 Data Preparation and Analysis

The model construction began with retrieving, cleaning, and analyzing data. The dataset contains chest X-ray (CXR) images, including 3,500 "Normal" images and 700 images of patients with "Tuberculosis," sourced from a publicly accessible repository [29]. This dataset is utilized to train our model. Figure 3 shows a sample of the dataset, with the heatmap representation highlighting anatomical structures and varying X-ray intensities.

4.3 Neural Network Architecture

Our proposed neural network architecture consists of an Input Layer, Convolutional Layer, Max-Pooling Layer, Flattening Layer, Dense Layer, and Output Layer. Table 3 provides a detailed explanation of the layers in a convolutional neural network (CNN) designed for binary classification tasks.

- *Input Layer:* Specifies that the input image size is 250. This means that each input image entered into the CNN has dimensions of 250x250 pixels.
- *Convolutional Layer:* Describes a convolutional layer with 12 filters, each of size 3x3 pixels, and utilizing the ReLU (Rectified Linear Unit) activation function. This layer applies these filters to the input image to detect various features such as edges and textures.
- *Max-Pooling Layer:* Indicates that this layer reduces the spatial dimensions of the feature maps produced by the convolutional layer. Max-pooling typically involves taking the maximum value from a set of values within a defined window, thus down-sampling the data and reducing the computational load for subsequent layers.
- *Flattening Layer:* This layer transforms the two-dimensional feature maps into a one-dimensional vector. This is necessary to prepare the data for the fully connected dense layer that follows.
- *Dense Layer:* Describes a fully connected dense layer with a single neuron and a sigmoid activation function. This layer combines the features extracted by the previous layers to make a final prediction. The sigmoid activation function is used for binary classification, providing an output probability between 0 and 1.
- *Output Layer:* Binary classification.

This architecture is designed for image classification tasks, with the convolutional layers capturing hierarchical features, followed by max-pooling for spatial dimensionality reduction. The flattened data is then input into a dense layer for further processing, and finally, a single neuron in the output layer performs binary classification using the sigmoid activation function.

Table 2 Comparison of pruning, quantization, and clustering

Feature	Pruning	Quantization	Clustering
Definition	Removal of unnecessary weights or neurons	Mapping weights to a smaller set of values	Grouping similar data points into clusters
Objective	Reduce the number of parameters in a model	Reduce the precision of weights	Reduce the number of unique values
Process	Identify, and eliminate insignificant weights	Round off weights to a smaller set	Group data points based on similarity
Impact on Model Size	Reduces model size significantly	Reduces model size significantly	Reduces model size moderately
Impact on Accuracy	May lead to slight loss in accuracy	May lead to loss in accuracy	May lead to moderate loss in accuracy
Computational Cost	Reduces inference computational cost	Reduces inference computational cost	May reduce computational cost in training
Common Use Cases	Model compression, speed up inference	Model compression, speed up inference	Data compression, image segmentation

Fig. 3 Sample of the dataset

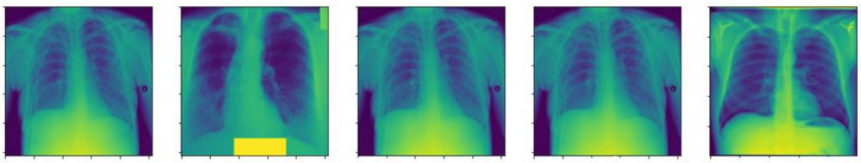


Table 3 CNN architecture

Layer	Details
Input Layer	Image size equal to 250
Convolutional Layer	12 filters, 3x3 kernel, ReLU activation
Max-Pooling Layer	Reduces spatial dimensions
Flattening Layer	Prepares data for the final dense layer
Dense Layer	Single neuron, sigmoid activation function for binary classification

The model is compiled using the Adam optimizer and binary cross-entropy loss, with accuracy as the evaluation metric. The EarlyStopping callback is implemented to prevent overfitting, monitoring the validation loss with a minimum delta of 0.001 and a patience set to 1 epoch.

4.4 Federated Learning Implementation

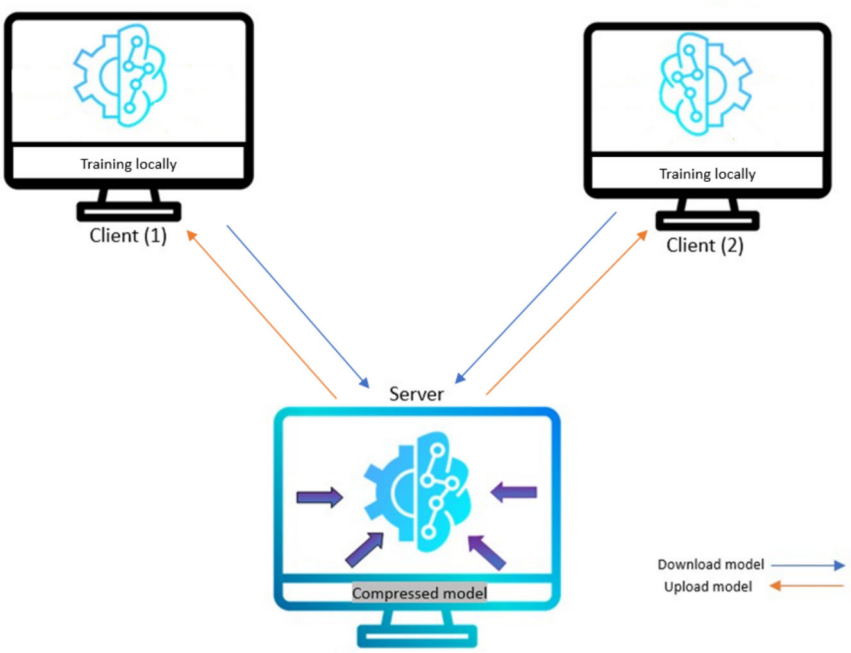
After preparing the data and building the model, the federated learning (FL) task begins. FL starts with a central server initializing a global model, which is then distributed to two clients. Each client independently trains the model using local data and sends the updated weights back to the central server, where they are aggregated to form a new global model. This process is repeated iteratively until a predefined number of rounds is completed. The final global model is then evaluated.

The number of rounds was fixed at 5, based on several experiments that indicated this provided an optimal balance between training time and model performance. Testing showed that adding more rounds did not significantly improve accuracy but greatly increased computational and communication costs. By limiting the training to 5 rounds, the model achieved satisfactory accuracy and robustness without unnecessary resource expenditure, thus ensuring efficiency in the federated learning process.

Figure 4 illustrates the methodology of federated learning (FL) employed in the depicted system. The process begins with a central server that initializes a global model, which is then distributed to two clients (Client 1 and Client 2). The steps in the figure are as follows:

1. The central server compresses the global model and sends it to the clients. This is represented by the orange arrows pointing from the server to each client.
2. Each client independently trains the received model using its local data. This training process is not explicitly shown in the figure but is implied as the clients handle the model after downloading it.

Fig. 4 Our FL methodology



3. After the clients complete the local training, they compress their updated model parameters and download them back to the central server. This is represented by the blue arrows pointing from each client back to the server.
4. The central server then aggregates the model updates from all clients to form a new, improved version of the global model. This step is implied in the central server's depiction and is crucial for the iterative nature of federated learning.

4.5 Compression Techniques

This section discusses techniques used to compress the models before sending them to clients. These techniques reduce traffic between the server and clients, thereby improving upload and download speeds. The following compression techniques were employed. First, Pruning involves reducing the number of model parameters and enhancing efficiency by eliminating redundant or less critical weights. This technique minimizes the model's size and improves communication speed between the server and clients [30]. Second, Clustering involves grouping model parameters to form a clustering algorithm. This technique optimizes the model's structure, reducing communication overhead during the federated learning (FL) process [31]. Third, Quantization typically leads to a significant reduction in the model's size by representing weights with fewer bits. This technique is crucial for decreasing the data transmitted between the server and clients [32]. Lastly, Quantization with Pruning and Clustering (QPC) integrates these three methodologies to optimize compression efficiency. As illustrated in Fig. 5, this hybrid (QPC) approach aims to achieve a more efficient and compact model. We conduct experiments to extract results and compare them with the performance of individual compression techniques.

Algorithm 1 FL with Compression Techniques

```
Build the model at the central server
Compress the model
Send the compressed model to the two clients
for each round r = 1 to 5 do
    For each client:
        Train the model on their data
        Send the trained model to the server
    End For
    Aggregate the models at the server
    Update model
    Send the Updated model to the clients
end for
Evaluate and save the results at the central server
```

Algorithm 1 outlines a federated learning (FL) process with model compression techniques. Initially, a model is built and compressed at the central server, then sent to two clients. Over five rounds, each client trains the model on their local data and sends the updated model back to the server. The server aggregates these models, updates the global model, and sends it back to the clients. After completing the rounds, the server evaluates and saves the final model results.

By adopting the FL approach, we achieve collaborative model training while prioritizing data privacy. The server facilitates seamless information exchange, allowing each client to enhance the model using localized data. This iterative process continues until convergence or a set number

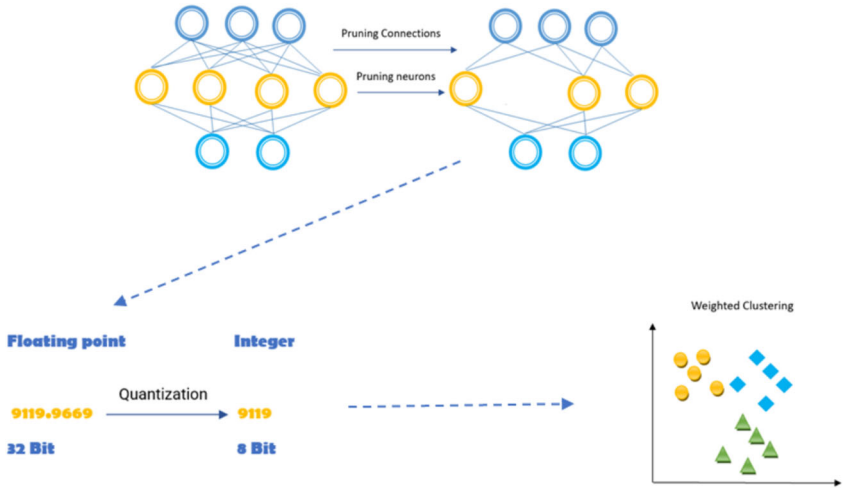


Fig. 5 QPC technique

of rounds. Additionally, by integrating compression techniques, we focus on finding a balance between reducing model size and maintaining accuracy, optimizing communication in the FL process. This comprehensive strategy anticipates the system's efficiency, promoting collaborative model improvement and secure data handling.

4.6 Experimental Setup

Several experiments were conducted to identify the most effective techniques to reduce traffic congestion and increase upload and download speeds between the central server and the clients. The goal is to enhance the FL model, which serves as a secure platform for specific institutions and hospitals concerned about the security of their sensitive data. We experimented with three compression techniques: Quantization, Pruning, and Clustering, along with a hybrid of these three techniques which we name it (QPC). The evaluation of results focused on reducing traffic between the server and the clients and on model accuracy, measured through three performance metrics: download speed, upload speed, and accuracy. The experiments utilized chest X-ray images for tuberculosis classification, training the model to determine whether a given image corresponds to a person infected with tuberculosis or is healthy. For the experimental environment, we utilized the Spider program through Anaconda, selected to ensure compatibility of the experiments.

Flower is employed as an FL framework alongside several libraries [33], including Pandas and NumPy for data analysis and cleaning, and TensorFlow and Keras for deep learning. Flower is an open-source framework created by a team of developers at OpenMined, a leading company in open-source information technology and software. It is designed to provide a robust and flexible framework to facilitate the development of FL applications. Flower allows the creation of virtual clients and the smooth execution of experiments. Its features include simplified communication abstractions between servers and clients, scalability to meet diverse needs, compatibility with TensorFlow and PyTorch, and customization capabilities to address specific requirements. Flower aims to simplify the development of FL models and enables seamless integration with various applications and systems.

4.7 Hyper-Parameters

FL is discussed in a specific configuration involving two clients and five rounds. This approach ensures collaborative and decentralized machine learning, where two client devices contribute to model training over five rounds.

The convolutional neural network (CNN) architecture consists of an input layer with an image size of 250, followed by a convolutional layer with 12 filters, a 3x3 kernel, and ReLU activation. Subsequently, a max-pooling layer reduces spatial dimensions, and a flattening layer prepares the data for the final dense layer with a single neuron and a sigmoid activation function for binary classification. The model is compiled using the Adam optimizer and binary cross-entropy loss, with accuracy as the evaluation metric. The EarlyStopping callback is implemented to prevent overfitting, monitoring the validation loss with a minimum delta of 0.001 and a patience set to 1 epoch.

Regarding the compression techniques, the specified parameters set a Pruning schedule with a sparsity level of 50%, which is the default setting. This means that during training, 50% of the model's weights will be removed, focusing on those that are less important. This makes the model smaller and more efficient without significantly losing performance. The pruning schedule dictates when and how the weights will be pruned throughout the training process, ensuring that the model gradually adjusts and maintains its ability to learn effectively even as its parameters are reduced.

For Quantization, which significantly reduces the size of the model by using fewer bits to represent the weights, models typically use 32-bit floating point numbers, but quantization converts them to less precise formats. In our experiments, we converted the weights to 8-bit integers, as this is the default parameter for quantization. This reduction in bit width means that model parameters and activations use less memory and require less computation. Consequently, smaller models are loaded and transmitted more quickly, leading to reduced traffic.

Additionally, the Clustering parameters refer to the settings used to configure a clustering algorithm, a technique for grouping similar weight points together. One of the key parameters in clustering is the number of clusters, which determines how many groups the algorithm will create from the weights. In this context, the number of clusters is specified as 8. This choice is based on the common practice of using a relatively small number of clusters to balance compression and accuracy. This specific number is chosen to achieve a balance between providing adequate detail to capture meaningful patterns in the model weights while avoiding computational overhead.

5 Results and Discussion

This section presents and discusses the results of a series of experiments designed to assess the efficacy of the proposed federated learning (FL) framework in conjunction with various compression techniques. The evaluations centered on three pivotal parameters: upload speed, download speed, and accuracy. Tables 4, 5, and 6 illustrate how different compression methods influence overall efficiency and model performance.

Table 4 shows the results of experiment (1). In this experiment, we investigated the effects of different optimization techniques on upload speed, download speed, and accuracy, using the Default configuration as a baseline. The Default method achieved an upload speed of 0.769 GB/s, a download speed of 16.032 GB/s, and an accuracy of 0.9952. The Quantization method provided slight improvements in both upload (0.778 GB/s) and download speeds (16.105 GB/s) with a small reduction in accuracy to 0.9914.

Pruning significantly increased upload and download speeds to 6.513 GB/s and 229.124 GB/s, respectively, while maintaining an accuracy of 0.9886. Clustering offered the highest upload and download speeds of 33.162 GB/s and 532.787 GB/s, respectively, with a slightly lower accuracy of 0.9891. The combined Quantization, Pruning, and Clustering (QPC) approach demonstrated a balanced performance, achieving upload and download speeds of 14.252 GB/s and 360.344 GB/s, respectively, along with an accuracy of 0.9714.

The results indicate that all techniques are comparable to the default in terms of accuracy. Notably, the clustering technique outperformed other techniques for both upload and download speeds. The download speed differed from the default by 516.755 GB/s, and the upload speed differed from the default by 32.393 GB/s.

Table 5 presents the results of experiment (2). The Default method served as the baseline, achieving an upload speed of 6.6574 GB/s, a download speed of 59.4926 GB/s, and an accuracy of 0.9872. Quantization significantly enhanced the upload and download speeds to 26.2503 GB/s and 234.0295 GB/s, respectively, although it resulted in a notable drop in accuracy to 0.9514. Pruning demonstrated similar upload speeds to the Default method (6.7308 GB/s) but showed a marginal improvement in download speed (59.8317 GB/s) while maintaining a slightly lower accuracy of 0.9732. Clustering offered substantial improvements in both upload (20.7455 GB/s) and download speeds (453.4209 GB/s), although with a moderate accuracy reduction to 0.9663. Finally, the QPC method, which combines Quantization, Pruning, and Clustering, yielded balanced results with upload and download speeds of 21.0561 GB/s and 378.3862 GB/s, respectively, and an accuracy of 0.9605. The evaluation shows that using both Quantization and Clustering methods demonstrated significant improvements in communication efficiency. However, this enhancement is accompanied by a trade-off in accuracy. These findings emphasize the critical need to achieve a careful balance between speed and precision.

Table 6 presents the results of experiment (3). The Default method served as the benchmark, yielding an upload speed of 7.0697 GB/s, a download speed of 26.8515 GB/s, and the highest accuracy of 0.9948. The Quantization technique significantly enhanced both upload (27.1912 GB/s) and download speeds (250.1789 GB/s), but resulted in a reduced accuracy of 0.9746. Pruning demonstrated a balanced improvement, achieving an upload speed of 13.6392 GB/s and a download speed of 156.376 GB/s, with a slight decrease in accuracy to 0.9882. Clustering delivered the highest upload speed of 35.2328 GB/s and a notable download speed of 117.384 GB/s, although with a moderate reduction in accuracy to 0.9706. The QPC method presented a moderate increase in upload (9.3347 GB/s) and download speeds (27.522 GB/s), with an accuracy of 0.9698..

The differences in results across Experiments (1), (2), and (3), despite using the same dataset and hyperparameters, can be attributed to algorithmic sensitivity. This sensitivity indicates that the optimization techniques–Default, Quantization, Pruning, Clustering, and QPC–respond differently to specific characteristics and nuances within the weights. For instance, methods like Clustering and QPC may perform variably depending on the distribution, density, and inherent patterns of the weights, which can affect their ability to group

Table 4 Experiment (1)

Experiment	Upload speed (GB/s)	Download speed (GB/s)	Accuracy
Default	0.769	16.032	0.9952
Quantization	0.778	16.105	0.9914
Pruning	6.513	229.124	0.9886
Clustering	33.162	532.787	0.9891
QPC	14.252	360.344	0.9714

Table 5 Experiment (2)

Experiment	Upload speed (GB/s)	Download speed (GB/s)	Accuracy
Default	6.6574	59.4926	0.9872
Quantization	26.2503	234.0295	0.9514
Pruning	6.7308	59.8317	0.9732
Clustering	20.7455	453.4209	0.9663
QPC	21.0561	378.3862	0.9605

Table 6 Experiment (3)

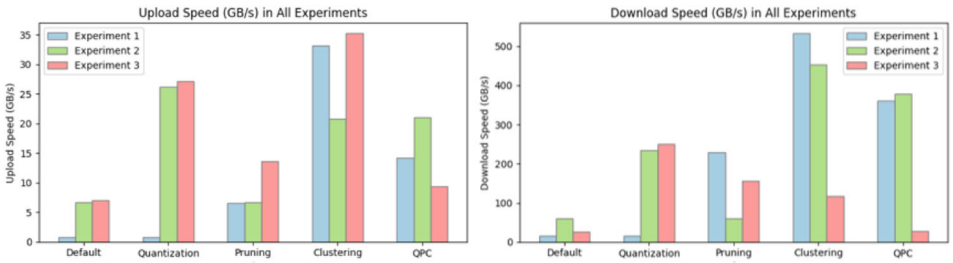
Experiment	Upload speed (GB/s)	Download speed (GB/s)	Accuracy
Default	7.0697	26.8515	0.9948
Quantization	27.1912	250.1789	0.9746
Pruning	13.6392	156.376	0.9882
Clustering	35.2328	117.384	0.9706
QPC	9.3347	27.522	0.9698

and optimize weight points effectively. Similarly, Quantization might be more or less effective based on the precision and range of weight values, while Pruning's efficiency can vary with the amount of redundant or irrelevant information present. These sensitivities lead to fluctuations in upload and download speeds.

The analysis provided by Fig. 6 demonstrates that, following multiple experiments, the clustering technique emerges as the best approach for compression. The results indicate that Clustering yields superior outcomes in both upload and download speeds while maintaining exemplary accuracy. This effectiveness significantly contributes to reducing traffic between the server and clients. Clustering outperforms both Quantization and Pruning due to its ability to preserve the underlying model structure. It typically has a lesser impact on model accuracy and information loss, providing a more favorable trade-off. Unlike Pruning and Quantization techniques, Clustering avoids drastic modifications to weights or the removal of units, leading to smoother integration with the model. Additionally, Clustering tends to have a milder effect on performance metrics such as upload and download speeds. These experimental findings highlight the critical importance of meticulous selection and fine-tuning of compression techniques tailored to specific use cases and priorities. The insights gained contribute valuable knowledge for optimizing the proposed FL framework, aligning with the nuanced requirements of communication efficiency and model accuracy in real-world applications.

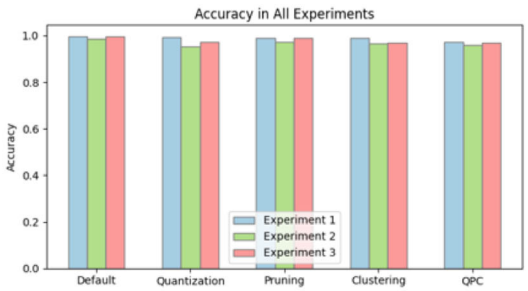
5.1 Limitations

While the proposed method offers significant advancements in improving communication efficiency and reducing resource consumption in federated learning (FL) for medical image analysis, it has several limitations. Firstly, the study is confined to a specific medical imaging technique—X-rays—and does not extend to other modalities, such as CT scans, MRI, or ultrasound. The performance of the proposed method may vary when applied to different imaging techniques, and its generalizability to other medical imaging data remains to be established. Secondly, only three compression techniques–Pruning, Clustering, and Quantization–were explored in this study, along with the hybrid method (QPC). However, other FL optimization techniques, including adaptive aggregation, differential privacy, and homomorphic encryption, were not considered. Future research could expand on these methods to provide a more comprehensive evaluation of FL's potential in medical image analysis. Another limitation involves the trade-offs between model accuracy and computational efficiency due to the compression techniques. While the focus of the study was on enhancing communication efficiency and reducing resource



(a) Upload speed (GB/s) in All Experiments

(b) Download speed (GB/s) in All Experiments



(c) Accuracy in All Experiments

Fig. 6 Comparison of upload speed, download speed, and accuracy in all experiments

consumption, it is possible that the applied compression methods may introduce slight reductions in model accuracy or performance, depending on the specific use case. Lastly, this study does not aim to address all the limitations found in related work. Instead, its primary focus is on improving communication efficiency and resource consumption while striving to maintain acceptable levels of model accuracy.

5.2 Future Work

Even though this study produced encouraging findings, there are several alternative directions that might be pursued in the future. Suggested areas for improvement include integrating additional privacy-preserving techniques, such as homomorphic encryption or secure multi-party computation, to enhance data security in federated learning (FL). Furthermore, scalability studies are recommended to assess the proposed FL framework's efficiency with compression techniques as the number of participating clients increases, particularly in large-scale setups. Additionally, applying the framework and compression techniques to real-world applications, such as healthcare or finance, would allow for a practical assessment of performance and feasibility in domains with stringent data privacy requirements. Importantly, the impact of the proposed techniques on energy consumption and latency in resource-constrained environments, such as IoT devices or mobile clients–especially in healthcare settings–should be explored. These factors are crucial for ensuring that the proposed solutions remain practical and efficient in environments where minimizing energy usage and reducing latency are essential. By addressing these research directions, there is potential for significant advancements in FL and compression techniques, fostering adaptability and applicability in diverse real-world scenarios. In conclusion, the integration of compression techniques in FL not only addresses communication challenges but also enhances scalability and efficiency. The findings contribute to the ongoing optimization of FL, guiding future research toward exploring different compression dimensions, diverse datasets, and broader applications in the evolving landscape of distributed intelligence, healthcare, and resource-constrained environments.

6 Conclusion

This research paper focuses on addressing communication traffic challenges in federated learning (FL), a vital method for preserving data privacy in healthcare and distributed intelligence. The study emphasizes the importance of medical image analysis and introduces FL as a privacy-preserving solution. The primary concern is the bottleneck in communication traffic within FL systems, prompting the research to propose innovative solutions involving compression techniques such as quantization, pruning, and clustering. The experiments conducted assessed the effectiveness of the proposed FL framework using various compression methods, revealing clustering as the promising approach. This technique demonstrates superior upload and download speeds

while maintaining high accuracy, significantly reducing traffic between the central server and clients.

Author Contributions All authors contributed equally to this work.

Data Availability Data will be made available upon reasonable request.

Declarations

Conflict of interest The authors declare that they have no Conflict of interest to report regarding the present study.

References

1. Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.A.; Ciompi, F.; Ghafoorian, M.; Van Der Laak, J.A.; Van Ginneken, B.; Sánchez, C.I.: A survey on deep learning in medical image analysis. Med. Image Anal. **42**, 60–88 (2017)
2. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
3. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. (TIST) **10**(2), 1–19 (2019)
4. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**, 1–210 (2021)
5. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
6. Silva, F.R.; Camacho, R.; Tavares, J.M.R.: Federated learning in medical image analysis: a systematic survey. Electronics **13**(1), 47 (2023)
7. Li, Z.; Xu, X.; Cao, X.; Liu, W.; Zhang, Y.; Chen, D.; Dai, H.: Integrated CNN and federated learning for Covid-19 detection on chest x-ray images. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2022)
8. Mouhni, N.; Elkalay, A.; Chakraoui, M.; Abdali, A.; Ammoumou, A.; Amalou, I.: Federated learning for medical imaging: an updated state of the art. Ingénierie des Systèmes d'Information **27**(1), 143–150 (2022)
9. Wang, W.; Li, X.; Qiu, X.; Zhang, X.; Brusic, V.; Zhao, J.: A privacy preserving framework for federated learning in smart healthcare systems. Inf. Process. Manag. **60**(1), 103167 (2023)
10. Hossen, M.N.; Panneerselvam, V.; Koundal, D.; Ahmed, K.; Bui, F.M.; Ibrahim, S.M.: Federated machine learning for detection of skin diseases and enhancement of internet of medical things (IOMT) security. IEEE J. Biomed. Health Inform. **27**(2), 835–841 (2023)
11. Huang, L.; Shea, A.L.; Qian, H.; Masurkar, A.; Deng, H.; Liu, D.: Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. J. Biomed. Inform. **99**, 103291 (2019)
12. Islam, M.; Reza, M.T.; Kaosar, M.; Parvez, M.Z.: Effectiveness of federated learning and CNN ensemble architectures for identifying brain tumors using MRI images. Neural Process. Lett. **55**(4), 3779–3809 (2023)
13. Cui, J.; Zhu, H.; Deng, H.; Chen, Z.; Liu, D.: Fearh: federated machine learning with anonymous random hybridization on electronic medical records. J. Biomed. Inform. **117**, 103735 (2021)
14. Al Aziz, M.M.; Anjum, M.M.; Mohammed, N.; Jiang, X.: Generalized genomic data sharing for differentially private federated learning. J. Biomed. Inform. **132**, 104113 (2022)
15. Cui, L.; Su, X.; Zhou, Y.; Pan, Y.: Slashing communication traffic in federated learning by transmitting clustered model updates. IEEE J. Sel. Areas Commun. **39**(8), 2572–2589 (2021)
16. Cui, L.; Su, X.; Zhou, Y.; Liu, J.: Optimal rate adaption in federated learning with compressed communications. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, pp. 1459–1468 (2022)
17. Cui, L.; Su, X.; Zhou, Y.: A fast blockchain-based federated learning framework with compressed communications. IEEE J. Sel. Areas Commun. **40**(12), 3358–3372 (2022)
18. Aledhari, M.; Razzak, R.; Parizi, R.M.; Saeed, F.: Federated learning: a survey on enabling technologies, protocols, and applications. IEEE Access **8**, 140699–140725 (2020)
19. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
20. Mothukuri, V.; Parizi, R.M.; Pouriyeh, S.; Huang, Y.; Dehghantanha, A.; Srivastava, G.: A survey on security and privacy of federated learning. Future Gener. Comput. Syst. **115**, 619–640 (2021)
21. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R., et al.: Advances and open problems in federated learning. Found. Trends® Mach. Learn. **14**, 1–210 (2021)
22. Shahid, O.; Pouriyeh, S.; Parizi, R.M.; Sheng, Q.Z.; Srivastava, G.; Zhao, L.: Communication efficiency in federated learning: aachievements and challenges. arXiv preprint arXiv:2107.10996 (2021)
23. Gupta, M.; Goyal, P.; Verma, R.; Shorey, R.; Saran, H.: Fedfm: Towards a robust federated learning approach for fault mitigation at the edge nodes. In: 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 362–370. IEEE (2022)
24. Yang, S.; Huang, J.; Li, Z.; Guo, S.: A truthful and reliable incentive mechanism for federated learning based on reputation mechanism and reverse auction. Electronics **12**(3), 517 (2023)
25. Jiang, W.; Li, F.; Zhang, Q.: A comprehensive survey of incentive mechanism for federated learning. arXiv preprint arXiv:2106.15406 (2021)
26. O'Shea, K.; Nash, R.: An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458 (2015)
27. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T.: A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282 (2017)
28. Ye, S.; Zhang, T.; Zhang, K.; Li, J.; Xie, J.; Liang, Y.; Liu, S.; Lin, X.; Wang, Y.: A unified framework of DNN weight pruning and weight clustering/quantization using admm. arXiv preprint arXiv:1811.01907 (2018)
29. Chest tuberculosis detection—kaggle.com. https://www.kaggle.com/code/aissaasasd/chest-tuberculosis-detection. Accessed 30-12-2023
30. Blalock, D.; Gonzalez Ortiz, J.J.; Frankle, J.; Guttag, J.: What is the state of neural network pruning? Proc. Mach. Learn. Syst. **2**, 129–146 (2020)
31. Bakker, B.; Heskes, T.: Clustering ensembles of neural network models. Neural Netw. **16**(2), 261–269 (2003)

32. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision, pp. 291–326. Chapman and Hall/CRC, (2022)
33. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; Gusmão, P.P.B., et al.: Flower: a friendly federated learning research framework. arXiv preprint arXiv:2007.14390 (2020)