IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 September 2022, accepted 30 September 2022, date of publication 5 October 2022, date of current version 19 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3212550

RESEARCH ARTICLE

Cloudlet Federation Based Context-Aware Federated Learning Approach

SANA LATIF[1], MUHAMMAD ZIAD NAYYER[2], IMRAN RAZA[1], (Member, IEEE), SYED ASAD HUSSAIN[1], (Member, IEEE), M. HASAN JAMAL[1], SOOJUNG HUR[3], AND IMRAN ASHRAF[3]

[1]Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan
[2]Department of Computer Science, GIFT University, Gujranwala 52250, Pakistan
[3]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea

Corresponding authors: Soojung Hur (sjheo@ynu.ac.kr) and Imran Ashraf (ashrafimran@live.com)

This work was supported by the 2021 Yeungnam University Research Grant.

ABSTRACT A Cloudlet federation can be beneficial to overcome the latency and resource scarcity challenges in a cloudlet deployment altogether, as a task can run on a cloudlet within the federation, sharing resources of member cloudlets. Nonetheless, the cloudlet federation is not context-aware in terms of latency, so to perform federated learning in cloudlet federation, the selection of a resource-efficient deep learning model is challenging. Additionally, the accuracy of a deep learning model can be affected if end-user devices are unreliable and provide incorrect data for training deep learning models at the cloudlets. Thus, resource and context-aware federated learning solutions are required for accurate and latency-critical applications such as COVID-19 detection using X-ray images. This paper presents a novel context-aware cloudlet federated learning solution for COVID-19 detection that monitors the resources of a cloudlet using a broker thereby minimizing latency without any impact on the accuracy of the deep learning model. Results show that the proposed model reduces the latency by 5% and increases the accuracy by 5% as compared to the state-of-the-art conventional federated learning approach.

INDEX TERMS Cloudlet federation, federated learning, edge computing, cloud computing.

I. INTRODUCTION

Cloud computing facilitates resource-limited devices such as mobile and Internet of Things (IoT) to carry out resource-intensive tasks on high-end remote servers. Irrespective of the numerous benefits provided by cloud computing, the distant cloud has latency limitations for systems such as smart vehicles and healthcare [1]. To overcome these limitations, edge computing (EC) based solutions are proposed that bring computation to closer proximity to the user thereby eliminating latency and resource scarcity challenges. Three types of edge architectures are proposed in the literature namely, mobile edge computing (MEC) [2], cloudlet computing [3], [4], [5], and fog computing. Edge resources are frequently used to perform Federated Learning (FL). FL is a decentralized machine learning (ML) approach that trains centralized deep learning (DL) models locally from datasets, distributed across multiple user devices [6]. However, cloudlet-based solutions exhibit storage and communication resource scarcity challenges for the training of DL models [7]. These challenges are inherently resolved by cloudlet federation [8]. MEC was introduced to resolve latency limitations, as FL in MEC provides the advantage of localized model training and uploading the locally updated model rules to a centralized cloud and fetching aggregated ML model parameters from the cloud [9]. Using cloud services to store, process, and analyze data proves to be advantageous as ample resources are available, however, as the

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

size of the ML model grows, specifically for DL models, provisioning of computational resources becomes difficult.

The problem of latency associated with cloud computing limits its use for certain applications such as smart vehicles and smart homes. Such scenarios require less than a few microseconds of response time for real-time communication [10]. Moreover, provisioning of high bandwidth for uploading a large amount of data to a cloud is also challenging [1], [7]. So, cloud computing-based state-of-the-art conventional federated learning approach (CONFLA) cannot be used for model training due to these constraints. Hence, in this work, a Context-Aware Cloudlet Federated Learning Approach (CACFLA) is presented that addresses the challenges of latency to run DL models on edge nodes. A shared DL model is trained at each edge node and upon resource scarcity issue at any edge node during training, the task is offloaded to another node in the federation. The training process is handled using a localized server that keeps track of clients' data for neural networks (NN) training. Moreover, the server keeps track of previously trained models to handle the bias in the results. The method decreases the latency and reduces the model convergence time.

A. MOTIVATION AND CONTRIBUTIONS

One of the major reasons for the spread of COVID-19 was initially the lack or absence of detection kits, their affordability, and their sensitivity to hot and humid weather causing the wrong detection of COVID-19 cases [11]. Most research studies either focused on the analysis and prediction of the spread, recovery, and death caused by this novel virus based on statistics across countries around the world [12] or its social and emotional effects based on data obtained from social media [13], [14]. Hence, accurate detection of COVID-19 was challenging in absence of an automated mechanism addressing the shortcomings. So, it is imperative to design a system that could accurately detect a COVID-19 infected person using NN models on X-ray images using FL, for timely control of the disease [15], [16], [17], [18], [19]. This requires a large number of patient records including ethical and legal considerations around patient confidentiality, as a centralized server is used to aggregate the patient records. Moreover, latency is also a major challenge in centralized FL, therefore, the proposed CACFLA scheme also implements the same use case. To the best of our knowledge, the use of cloudlet federation for resolving machine learning problems has not been reported in the literature so far. The detailed contributions of this research work are as follows:

- A context-aware cloudlet-based federated learning approach (CACFLA) is proposed that resolves compute-intensive DL models.
- CACFLA yields better results in terms of latency as compared to state-of-the-art CONFLA.
- The best DL model selection method is presented that yields better results in terms of accuracy and loss, achieving less convergence time.

This paper is structured as follows. Section II presents state-of-the-art approaches used for FL and their comparative analysis. Section III provides the details about implemented solution and its mathematical model. Section IV elaborates on the experimental setup and the discussion on the results followed by Section V which concludes this paper.

II. RELATED WORK

FL at edge proved beneficial in handling challenges faced by traditional centralized ML solutions. This section classifies some state-of-the-art solutions for the implementation of FL at the edge. The objective of this study is to identify FL solutions that use DL to perform context-aware computing for classification, prediction, and decision-making on the user's datasets. Training is performed either by deep NN (DNN) model partitioning or by offloading the model to other resourcefully rich edge nodes in the federation.

A. EXISTING DETECTION APPROACHES FOR COVID-19 BASED ON EDGE COMPUTING

An analysis of COVID-19 detection techniques provided in [20] shows that radiography-based detection methods using edge architecture are more effective and give better accuracy, especially with CT-scan images. In [19], authors used posteroanterior chest X-ray view (i.e., radiography). The dataset is passed to the convolutional neural network (CNN) based DL models for training and COVID-19 detection using edge architecture. However, in [17] authors have used FL architecture to identify COVID-19 using CNN models. The evaluation of edge and FL-based approaches show that FL performs better than EC for model training and prediction accuracy. Both approaches used have focused on the reduction of communication costs during the training process. Pang et al. [21] propose an FL framework to study the effectiveness of various prevention city plans to help prevent COVID-19 outbreaks using FL and maintaining data privacy.

1) DATA COMPRESSION AND PROCESSING BASED FL SOLUTIONS

The authors in [22] discuss the security-related problems and their solutions when performing ML tasks at the edge of the network. Moreover, the challenges of storing big data at a centralized location are also addressed in this work. This study resolves energy consumption, data integrity, and confidentiality issues. The proposed architecture consists of data sampling and compression techniques for low energy consumption and data privacy as well as security issues. This reduces transmission requirements and data dimensionality.

The problem of learning bias due to a generalized ML model training is addressed in [23]. A context-aware local machine learning approach is proposed for prediction and decision-making. Regression and evolutional NN are used for prediction and decision-making, respectively. This approach is context-aware in terms of input data based on which the ML model is selected for processing. The system architecture consists of regression-based NN for prediction and a

control module to configure and monitor context changes. The data size of the local node and network parameters are also considered to identify the context. The context-aware best model is selected based on absolute mean error and correlation function and is later used for decision-making. After the validation model, maximum accuracy is stored with the context information. Results indicate that the context-aware approach outperforms the generic approach even if general model training rounds are increased. However, this approach is not proposed for FL.

The authors in [24] discuss the problem of efficient management of sensor data in a heterogeneous network to perform ML inference tasks. The paper proposed a local general hierarchy-aware ML inference model for a heterogeneous network without affecting the prediction accuracy of IoT applications. The proposed hierarchy-aware approach is implemented on the decision tree and NN ML models. The architecture consists of offline and online phases, wherein a pre-trained ML model is used for the extraction of features and the calculation of split points of feature space. The feature space extracted from the locally available data model is split into a small, specialized model. These models can be trained using data available on each node. In the second phase, the process sends only a single optimal scalar value for each partition to sum it up in the cloud. As local nodes send only a single value to the cloud instead of multiple from each node, latency, and communication efficiency are maintained. Results indicate that energy consumption and network latency can be reduced to 67% and 63% even with less available bandwidth. The solution to providing cost-effective services for smart healthcare using EC is discussed in [25]. The proposed architecture consists of two components: a data compression module and an edge-based feature extraction module to adapt to a dynamic environment. By taking the data from hybrid sensing sources, data is aggregated and sent to nearby mobile edge nodes. Mobile edge nodes extract features from the multi-model data using DL techniques. The second step is the extraction of events from processed data using a frequency feature classifier (FFC). After feature extraction and classification, adaptive compression is performed using stacked autoencoders (SAE) for analysis and prediction. Results show that the proposed solution does not perform well for very low and high values of classification threshold (CT). For the middle region of CT, the accuracy is 98.3 percent.

Authors in [26] handle the challenges of scarcity of communication and computation resources at the edge. The system named filter pruning tensor train (FPTT) is proposed which consists of two layers. The first layer is filtered pruning and the second is tensor train decomposition. At first average kernel weight of each filter is selected, results are sorted and filters with smaller values are dropped out. In the second phase tensor train, decomposition is performed. Tensor train decomposition stores dimensional data in the form of a matrix based on low-rank matrix values. The calculated matrix is stored in a tensor train format. The model is tested with three varying settings. The first approach worked better for medium-scale DNN models. However, the second and third approaches showed the highest accuracy and better performance respectively in less dense networks. The author in [27] handles the statistical and systematic issues in FL. Challenges addressed in this paper are non-IID data, communication cost, dropout parties during the learning process, and fault handling to improve accuracy. A multitask learning-based framework MOCHA, an extension of the CoCoA optimization method, is proposed for the optimization of the learning process. The model is designed as multi-task learning (MTL) problem to handle data distribution. Simulations are performed on Google glass, vehicle sensors, and human activity recognition systems. Results show that MTL performs better than global and local model learning frameworks. With small data set global model performs better but for a large dataset error rate of Multi-Task Learning (MTL) seems better than other frameworks. Moreover, communication and statistical challenges are better addressed with MOCHA than by using CoCaA directly.

Network heterogeneity is a major issue that affects communication efficiency in FL which is addressed in [28]. The proposed model consists of one module each for compression and federated dropout. The compression phase reduces the number of features by using basis transform, subsampling, and quantization. The compression of the model reduces bandwidth demand for the transfer model. The second phase eliminates the unnecessary features of the model and transfers a specialized model to the local device for training. It reduces the number of iterations, and the model converges faster. The results show that in terms of the number of communication rounds, the downloaded model's size, corresponding updates size, and required local computations are reduced up to 14, 28, and 1.7 times, respectively, without degrading the model's accuracy. The authors in [29] resolve the resource-intensive live and archived data collection from distributed edge nodes. The proposed system is distributed into Eureka and front-end running components. User queries the Eureka system after which the query is processed to classify data at edge cloudlets in parallel. Eureka is divided into three main components; a filter container for software generality, an itemizer to generate a stream of data, and an item processor for the selection of items based on the query. Iterative refinement makes the training data selection process efficient. The cloudlets are selected based on the processing time for one item. Results show that early iteration gives low accuracy but as the number of iterations increases, the accuracy and efficiency of the model enhance.

2) DATA CLASSIFICATION USING CNN

The authors worked on adaptive decision-making based on context-based predictions in a smart home [30]. Reinforcement learning is applied to the raw sensor data for the extraction of context to update the decision-making process. The data captured from sensors is presented graphically in a 2D map in the form of an annotated and raw data corpus for

data extraction. Afterward, CNN is applied for context extraction and image processing. For the decision model (agent) reinforcement learning is used. Results show that the annotated corpus provides a higher F-measure in comparison with annotated data representation. Authors quantify the effect of resource and data heterogeneity on training time in [31]. Multiparty TensorFlow is used to emulate the FL environment. A CNN model is trained on non-IID data. Results indicate that as the resource allocation to each party and data size increase, model training time also increases. The problem of model training in a decentralized environment is investigated in [32]. The proposed model calculates the convergence bound of the ML model using gradient descent-based approaches in a federated environment. The proposed system uses SVM, regression, K-means, and CNN for training. Results show that the proposed control algorithm gives a near-optimum performance with different ML models and data distributions. Moreover, with synchronous updates, model convergence is fast as compared to asynchronous updates. Zhong et al. [33] propose an approach based on behavioral and content-based features incorporation to improve the prediction accuracy by adopting multiple DL models in a hierarchical tree structure.

The problem addressed in [34] is training the ML model during FL when available communication bandwidth is low. The proposed solution has two variations, structured updates, and sketched updates. After performing either of the techniques, results are compressed using quantization, subsampling, or random rotations. Results are calculated by either using a uniform and a varying number of clients or sketched and structured updates for fixed rounds. Without quantization sketched updates perform efficiently and use less bandwidth and converge in fewer iterations. Using quantization and only selected nodes for training, the subsampling rate can be decreased while having a minimal effect on accuracy. The authors optimized the FL process in [35] to improve the accuracy of the model and to efficiently utilize network resources. They present a FedAvg-based algorithm that optimizes the local and global model iterations to reduce network load. To improve the model accuracy online monitoring of clients and adjustment to global model parameters is made. Adjustment in parameters is made based on the priority assigned to each client at the start of each iteration. Moreover, the selection of clients at the start of each round reduces bias in the training process. CNN is used for the classification of nodes based on priority defined as low, medium, and high. The proposed model validates the test set with better performance.

3) MODEL OFFLOADING BASED SOLUTIONS

In [36], the authors focus on resource-constrained mobile devices. The study solves two challenges; computation offloading of DNN applications to an edge server and bandwidth limitation for offloading the task. The authors propose a 2-step pruning process for DNN model partition, a selection phase, and a deployment stage. The first step reduces the computation workload, and the second step reduces the transmission workload. A part of the model is kept on a local node for training and the other part is sent to the edge server. Accuracy and loss threshold values are 4%. After a two-step pruning process, transmission and computation costs are reduced by 25.6 and 6.01 times respectively. Results show better utilization of limited bandwidth and improved end-to-end latency. The authors present intelligent edge devices in [37] and used ML in mobile edge systems. The proposed solution consists of the integration of the FL framework in ME systems with deep reinforcement learning (DRL) techniques. The decision process is divided into three steps an information collection step, a cognitive computing step, and a request handling step. For computation and offloading DRL, an agent makes the decision based on the wireless channel, energy consumption, and inference results. Two use cases are implemented, (i) edge caching, and (ii) edge offloading using DRL. Results show that FL does not perform better than centralized learning but shows near-best performance.

The problem of optimization of DNN portioning and DNN right-sizing is addressed in [38]. The authors focus on the optimization of decision-making based on predefined latency and accuracy requirements. DNN models require rich resources and offloading of ML tasks to the cloud, this results in performance and latency overheads. The proposed framework Edgent is used for edge devices synergy for collaborative and on-demand ML co-inference tasks. Each client has a local agent Edgent which performs three tasks: training, optimization, and co-inference. Results show that the accuracy of the model increases as latency requirements become less important. The solution presented in [39] deals with the capability and connectivity challenges in IoT devices. The proposed solution offloads computation to edge nodes. It consists of a combination of FL and DRL agents for model training. DRL agent is deployed on IoT devices for decision-making and resource allocation in a dynamic environment. Results indicate that the DRL agent is slow and have low accuracy as compared to centralized model training.

Authors in [40] addressed the problem of computation offloading in a contention-based multiuser multi-channel wireless environment. The proposed solution consists of a fully distributed computation offloading (FDCO) algorithm. To handle communication overhead time division multiple access (TDMA) is used with carrier sense multiple access (CSMA) to avoid collisions in the channel. Results indicate that cloudlet computing gives optimal performance and lower system-wide execution costs by balancing transmission and computation costs. The authors in [41] studied the heterogeneity in network resources and high accuracy requirements for edge servers and industrial IoT (IIoT) nodes. The proposed framework focuses on the accuracy and heterogeneity constraints. Results show that the accuracy of the model increases as the number of local data increases. The model performs well with less congestion in the network but as the network becomes crowded the performance decreases. Authors in [42] addressed the constraint of high latency requirements for computation-intensive

TABLE 1. Summary of the literature review.

Ref.	Data input		ML Model			Technique	FL Objective			Tools	Focused parameters					
	Raw data	Processed data	Neural Networks	Decision Tire	Statistical Methods		Classification	Prediction	Optimization		Performance	Error	Accuracy	Latency	Energy	Efficiency
[23]		✔	✔			Control module		✔			✔	✔	✔			
[30]		✔	✔			DQN agent			✔		✔					
[24]		✔	✔	✔	✔	Hierarchy-aware inference algorithm	✔			Raspberry Pi3	✔		✔	✔	✔	
[36]	✔		✔			2 step pruning algorithm	✔	✔		PyTorch			✔	✔		
[37]		✔	✔			DRL agent			✔	TensorFlow	✔	✔				
[38]	✔		✔			Edgent			✔	Raspberry Pi			✔	✔		
[31]	✔		✔			FedCS			✔	TensorFlow			✔	✔		
[32]	✔					Control algorithm		✔	✔	Raspberry Pi	✔	✔	✔			
[10]	✔								✔	TensorFlow		✔	✔	✔		
[39]		✔	✔			DRL agent	✔	✔		✔	✔	✔				
[40]		✔	✔			Fully distributed computation offloading (FDCO) algorithm			✔		✔					
[25]	✔		✔			Frequency feature classifier	✔		✔			✔	✔			
[29]	✔		✔			Eureka	✔						✔			✔
[34]	✔		✔			FedAvg			✔		✔		✔			✔
[44]		✔	✔			FedAvg			✔				✔			
[26]			✔			Filter pruning and tensor train (FPTT)	✔		✔	TensorFlow		✔	✔			
[28]			✔			Federated dropoout algo+FedAvg	✔						✔			
[41]			✔			AMOLC			✔				✔			
[42]			✔		✔	BOMERANG	✔		✔	Raspberru Pi	✔		✔			✔
[22]							✔									
[45]		✔				CE-FedAvg			✔	Raspberry Pi			✔			
[46]			✔		✔	HierFAVG			✔		✔		✔			
[27]			✔			MTL-based MOCHA			✔		✔					
[35]	✔		✔			FedAvg			✔		✔		✔			
[8]	✔		✔			VIRTUAL			✔	TensorFlow	✔		✔			
[47]						Tasklet			✔		✔					

tasks in IoT devices. The proposed solution Boomerang is tested with two variations of the DL model: DRL-based Boomerang and regression-based Boomerang. Results show that regression-based Boomerang performs better but using DRL and DQL for decision making gives high accuracy. Zhang et. al [43] propose FL based prediction model sensing the ability of users under different contexts. The proposed scheme addresses the key issues in the data quality control of mobile crowd sensing.

4) MODEL AGGREGATION USING FedAvg ALGORITHM

The problem presented in [44] handles the issues associated with non-IID data during FL to increase accuracy. The proposed solution consists of a FedAvg algorithm. This algorithm gives higher accuracy with lower earth movers distance (EMD). To reduce the value of EMD authors proposed an algorithm to select a subset of data that is uniformly distributed in classes in comparison to cloud global data. By using this method EMD values are decreased, and the model converges with higher accuracy in fewer iterations. The results show that the accuracy of the model decreases by 55% when trained directly from non-IID data. On the contrary accuracy of the model increases by 30% for neural networks when 5% of global data is shared with the warm-up model. The study [45] works on reducing the number of iterations during FL. A communication-efficient CE-FedAvg algorithm is proposed to reduce the number of iterations by considering network parameters. It focuses on collecting data from edge devices and storing them on a nearby edge server to perform FL. The clients are randomly selected to perform model training on MNIST and CIFAR-10 datasets. To achieve the desired accuracy communication rounds are reduced 6 times and the model converged with three times fewer iterations per client. The problem of communication overhead during FL is addressed in [46]. A hierarchical FL model is proposed with the HierFAVG algorithm for aggregation at the edge server using MEC. Results show that performing aggregation at the edge server before sending data to the cloud can reduce communication overhead. Statistical challenges during FL are addressed by [8] by devising a technique for non-convex models trained on non-IID data. MTL learning framework is used to handle convex models. The authors introduced an extension of the FedAvg algorithm named VIRTUAL for non-convex problems for federated multi-task learning. Results show that the model shows optimal performance for non-convex and non-IID data.

Table 1 represents the summary of the literature review for FL. The literature review is divided into five sections. The first section presents the existing COVID-19 detection approaches based on EC. The second section shows that using

data compression techniques before model training significantly reduces the training time and number of iterations. The third section shows the significance of using the CNN model in classification for increased classification accuracy. CNN is frequently used for classification purposes due to its high accuracy. Regression-based NN models are commonly used for prediction While the fourth selection shows that offloading a resource-intensive DNN model to a resource-rich device i.e., bandwidth, storage DNN increases the training performance. In the last section, different modifications of the FedAvg algorithm are discussed to increase communication efficiency and decrease of usage of computation resources. Moreover, training models in existing studies are classified into NN, tree, ML, and AI models. All the approaches use edge resources for classification, decision-making, and optimization of the learning process. For decision-making, DL models are widely used. A 2-step pruning algorithm is also proposed in the literature to extract a context-aware model that will be used in our solution as a reference to make our ML model context-aware. The proposed approach will save the resources required to transmit large NN models as well as reduce latency during data transmission. We use Google-developed TensorFlow which is a widely used framework for the implementation of FL scenarios. Moreover, the literature review also presents that no FL-based or ME-based COVID-19 solutions based on X-ray or CT-scan images are discussed in the literature.

III. PROPOSED SOLUTION

The Cloudlet Federation for Resource Optimization Model (CFRO) architecture [3] is shown in Figure 1. It consists of a broker at a remote cloud and a federation of cloudlets in closer proximity to the users. The broker consists of a cloudlet registration module (CRM), information management module (IMM), and a decision support system (DSM), whereas each cloudlet in federation consists of a task management module (TMM), a device registration module (DRM), and a resource management module (RMM) and a broker agent acting on behalf of the broker.

The FL proposed solution extends the federated cloudlet architecture, as presented in Figure 2. The system consists of five novel modules to perform FL tasks. Three of them reside in the broker and two are inside cloudlet. Broker consists of a connection module, weight scaling module, and model aggregation module. Cloudlet contains a model training module and an aggregation Module.

The connection module inside the broker manages incoming clients, connecting them to take part in the learning process. It selects clients for training, assigns an ID to them, and creates a separate socket for communication. The weight scaling module finds client data cardinality to calculate the scaling factor. It assigns aggregated weights to each layer based on the calculated scaling factor and assigns those weights to the global model. To perform the aggregation task broker contains an aggregation module. This module keeps track of ML models from all clients using a local model

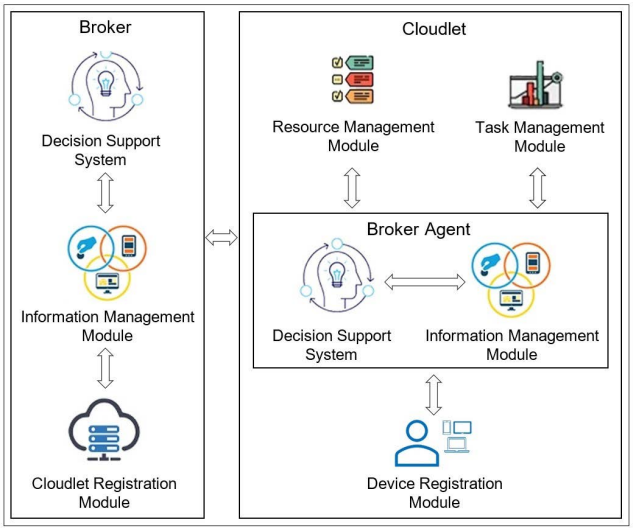


FIGURE 1. CFRO: Cloudlet federation for resource optimization.

management service and offloads the aggregation task to a client within the federation with maximum resources to update the generic model. Optimized model selection service selects ML model with maximum accuracy. Each client taking part in the learning process within the federation consists of a model training module and a model aggregation module. The model training module gets the global model for each training round and selects a model with maximum accuracy. The model aggregation module handles the offloaded model aggregation task. It consists of a model processing service, layer aggregation service, and an optimized model selection service to select the optimized ML model.

A. BROKER

The broker in FL has the responsibility to accept client connections, assign them unique IDs based on their IP addresses, and create a separate socket connection for communication. After that, it calculates a scaling factor based on the data cardinality of each client and assigns weights to each layer of the ML model. Based on the scaling factor calculated each client is assigned a learning rate, local epochs, and weights. To perform these tasks broker contains three modules named a connection module, a weight scaling module, and a model aggregation module.

1) CONNECTION MODULE

The connection module accepts connections from clients within the federation and assigns them individual IDs based on their Internet protocol (IP) addresses and connection information. It further communicates with the client to get data cardinality to find weights scaling factor and assign aggregated model weight to the global model. The client selection service accepts a connection from a client if it is a part of the federation. To ensure it, the server communicates with Information Management Module and matches the client parameters i.e., IP address. The server then generates a unique

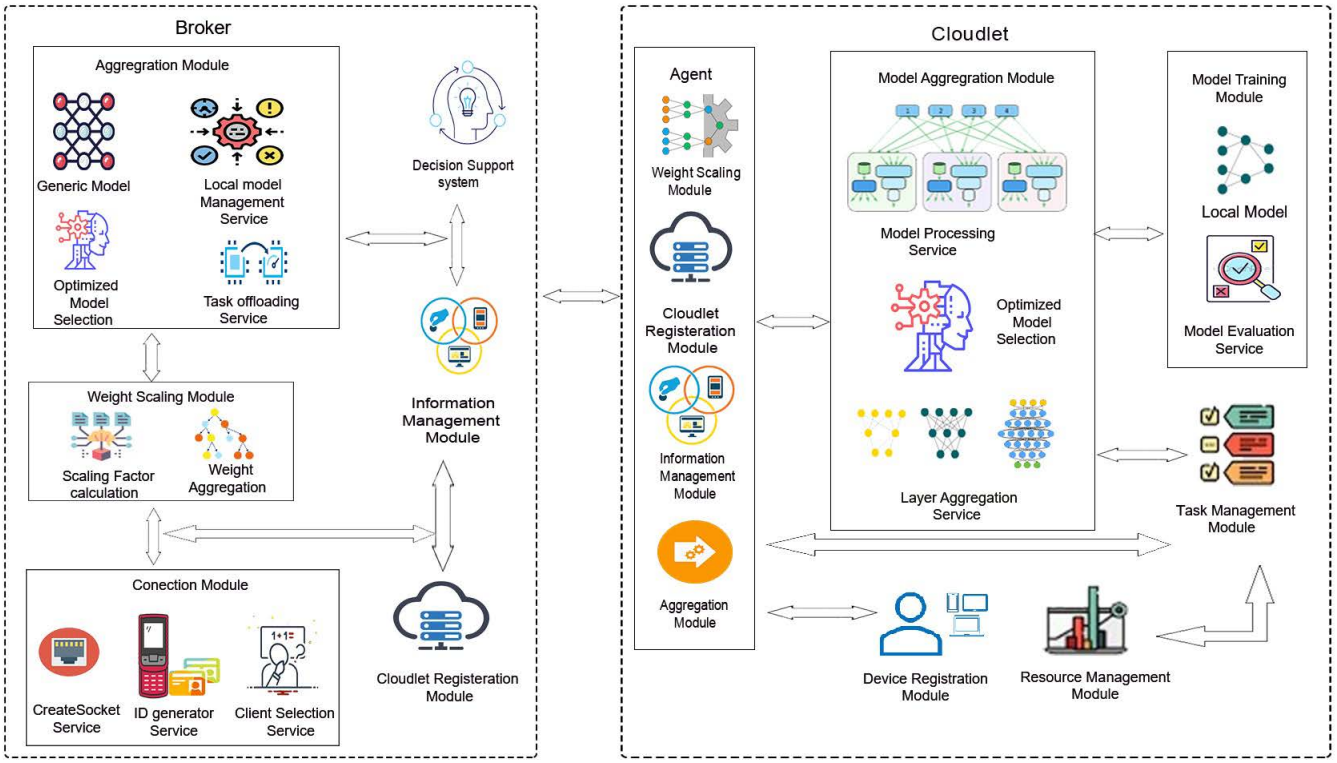


FIGURE 2. Federated cloudlet architecture based on CFRO.

ID for each client using its connection information and its IP address and sends the data to the socket creation service. The socket creation service creates a socket for each client and adds it to the client's list and informs the client about successful connectivity and sends data cardinality to start the training process.

2) WEIGHT SCALING MODULE

After getting data cardinality from the client, the value is sent to the weight scaling module to update the scaling value based on data cardinality and the number of total clients. As each client connects with the server its scaling factor is also updated based on the client and local training dataset. The weight scaling factor is used to scale model weights and is calculated using Equation 1.

$$WSF = \frac{\text{Total data of a client*Batch\_size}}{\text{sum of training data across clients}} \quad (1)$$

The model is scaled for each client based on the respective total amount of data set multiplied by the batch size used for each iteration to train the model. After that, the result is divided by the sum of the training dataset across all clients in the federation. This formula gives the updated scaling factor which is used to update the model weights in the model aggregation module.

Weight Aggregation service performs the aggregation task in two steps. Firstly, it scales local model weights using the weight scaling factor. Then, the mean of all local scaled weights is calculated to generate the global model.

3) MODEL AGGREGATION MODULE

The weight scaling factor is further used to scale a model weight for each client. After scaling the weight of each client, all models are used to get the mean value of each layer of the neural network and those weights are used as global weights. A local model management service is used to get local models from all clients in the federation. It ensures that the local model from all clients is received at the client-side, and it requests the respective client to send the local model again if it gets lost in the network due to issues of connectivity, etc. After receiving the model from all clients, the task offloading service communicates with IMM to get updated resource parameters and selects the client with optimized resources to offload the aggregation task. The client sends the aggregated model back to the server for evaluation. The model is evaluated based on its accuracy, it keeps track of the global model from the last iteration and if the accuracy of the previous global model is greater than the new model; the old model weights are assigned to the generic model for the next training epoch. The generic model service module defines a CNN model that consists of six CNN layers for the classification of images. The model is initialized with activation functions of rectified linear unit (ReLU) and hyperbolic tangent activation (tanh) at the start of the training process. These activation functions are used as they give higher accuracy and manage the output to pass it as input to the next layer. After initialization, it sends a global model for scaling and the model is sent to clients for training. This process is performed only once when the first client connects to the server to perform

the training. For each new round, this service sends updated weights from the last round to all clients to start the next epoch.

B. CLIENT CLOUDLET

Each client performs model training by getting updated global model weights from the server and training the model to get a local updated model of each client. These models are sent to the server to send the models to a client with maximum resources for aggregation tasks. This aggregation generates a model that is further evaluated based on its accuracy and based on the evaluation results weights are assigned to the global model.

1) MODEL TRAINING MODULE

This module is subdivided into two services; the local model service trains the local model and selects the local model with maximum accuracy and minimum loss value. Further validation is performed on the local dataset to check the model accuracy and loss on the test dataset. A model with the highest accuracy is selected as a local model to send to the server for global aggregation of models. The local model service gets the global model to start the training process based on updated weights, local training steps are performed with early stopping and checkpoint conditions. At the end of the training process, the NN model with maximum accuracy and minimum loss value is saved. The model trained on the local training set is evaluated on the validation dataset through the model evaluation service. If model accuracy is less than the global model it is sent back to the server without updating model weights to perform the aggregation task.

2) MODEL AGGREGATION MODULE

The model aggregation module activates when the broker offloads the aggregation task to a specific client. The client then splits the aggregation process into three steps. At first, it gets the model from the server and loads it into memory. In the next step, it scales the weight of each model using the global scaling factor and takes the mean of each layer across all clients. In the last step, the model is evaluated for an optimized model selection. The model processing service gets a model from all clients and loads it into memory to start the aggregation process. It also gets the latest scaling factor from the server to update NN weights. The layer aggregation service updates each model weight based on the scaling factor and then takes the mean of all local models. Updated weights are then assigned to the global model. The federated averaging algorithm, presented as Algorithm 2, is used for layer aggregation. The global model with updated weights is evaluated on the testing dataset. If model accuracy is greater than the last global model, the new model is sent to the server to start the next epoch. Root mean square proportional (RMSProp) is used as a global optimizer function. The global loss function used is mean squared error (MSE).

TABLE 2. Notations used for the model.

Notation	Definition
C	Set of cloudlets in the federation
C'	Set of workers in the federation
U	End-user devices
D	Set of patient record on worker i
W_G^t	Global averaging function
$L(W_G^t)$	Global loss minimization function
$L(W_i^t)$	Local loss minimization function
B	Batch for each training iteration
B'	Training batch
v_i'	Validation dataset
I'	Steps/epoch
M_{T_i}	Memory for iteration i
M_R	Memory capacity
l	Layer number
W	Learnable weights
b	Bias
g	Activation function
η	Learning rate
M	Local models
L'	Loss values set

C. MATHEMATICAL MODEL

The ML problem is designed as a convex problem having only one globally optimal CNN model for prediction. Suppose that a federation of cloudlets C has n cloudlets described as $C = c_1, c_2, c_3, \ldots, c_n$ and different set of resources available for each cloudlet are $S = s_1, s_2, s_3, \ldots, s_n$. Moreover, the workers are represented as $C' = c_1', c_2', c_3', \ldots, c_n'$ where $C' \subseteq C$ and the number of end-user devices connected to each cloudlet c_i are $U = u_1, u_2, \ldots, u_n$. The set of patient records on each cloudlet c_i is given by $D = d_1, d_2, d_3, \ldots, d_n$ where $n \in N$. Table 2 represents some commonly used notations and their explanation.

A relationship between a user and workers $U \rightarrow W$ is defined as many-to-one as many users can be connected to one worker at a time t. Moreover, a worker can store multiple data records of a single user, hence $W \rightarrow D$. Some cloudlets from multiple cloudlets in the federation are selected as workers for a single training round such that $\forall c_i \in C \exists C_j'$ where $c_i = c_j'$ for $i, j = 1, 2, 3, \ldots, n$. The goal of the training is to optimize the learning process while maximizing the global accuracy which is given by

$$Acc(w_G^t) = argmax(A(w_i^t)) \tag{2}$$

where w_G^t is global weight averaging function and $A(w_i^t)$ is the local accuracy optimization function. Similarly, $L(w_i^t) = argmin(L(w_i^t))$ is a local loss optimization function. The number of data records at the worker $c_{j\in N}'$ can be written as $D \in R^d$, where D_i represents the i^{th} data element. The Data matrix at each cloudlet is given by $D_i \in R^{dxd}$, $D_{i(mn)}$ designates the entry in m row and n^{th} column. Input at each worker c_i' is in the form of $n \times m$ matrix where $a_n m \in 0, 1, 2, \ldots, 255$ represents image pixel value at position $n \times m$ and $n = m = 128$. $Y_i' = y_1', y_2', y_3', \ldots, y_n'$ are corresponding data labels. Each element of the matrix D_i' is divided by 255 for one-hot encoding of the matrix to change categorical data with binary

data.

$$d_i' = \frac{D_i'}{255} \text{ such that } a_{mxn} \in \{0, 1\} \tag{3}$$

The training process starts after preprocessing the local input images dataset. Each local training round consists of a training step and a local aggregation step. After locally training the CNN model, the server in the federation receives the CNN model for global aggregation.

In the training step, a batch of n images is selected from input data such that T is the batch for training round i which is given by $T_i = \{t_1, t_2, t_3, \ldots, t_n\}$ such that $T_i \subseteq D$. The batch for training round i is split into training and testing datasets with a ratio of 0.8 to 0.2, respectively such that

$$T_i = \sum_{i=1}^{10} (T_i' + V_i') \tag{4}$$

where T_i' is the training dataset and V_i' represents the testing dataset for the i^{th} round. Training rounds consist of 10 global epochs and the number of epochs for each local training round, for each client, is determined using Equation 5.

$$I' = \frac{Length(T_i')}{Batch_size} \text{ where } I' = Steps\_per\_epoch > 0 \tag{5}$$

Here I' describes the local training steps and $Length(T_i')$ describes the length of training data set for training round i. The Batch size represents the total batch size for training round i. Training data set T_i' for iteration i is further divided into mini-batches each of size $n = 8$ such that $(x_i, y_i) \in (x_{train}, y_{train})$.

Constraint 1: The purpose of using mini-batch is to reduce memory consumption during model training such that $M_{T_i} \leq M^k$, where M_{T_i} is the memory required to load training data for iteration i and M^k is memory capacity of worker k. CNN will be used for classification with filter size 64, kernel size 3, and input dataset T_i'.

$$cov[m, n] = (x * h)[m, n] = \sum_{j=0} \sum_{k=0} h[j, k]x[m - j, n - k] \tag{6}$$

where m and n represent indexes of the output matrix, x represents the input image and h represents a filter matrix. The proposed Feedforward neural network consists of two steps. In the first step, convolution is performed, and input data is convolved with previous layers' learnable weights $W^{[l]}$, and then bias $b^{[l]}$ is added. In the second step, the activation function $g^{[l]}$ is applied to the updated weights $Z^{[l]}$. ReLU and tanh are activation functions (g) used for convolution layers.

$$Z^{[l]} = W^{[l]}.A^{[l-1]} + b^{[l]} \tag{7}$$

$$A^{[l]} = b^{[l]}(Z^{[l]}) \tag{8}$$

where l represents the number of layers and activation function $g(x)$ determines the values of output features.

$$ReLU(x) = \begin{cases} 0 & ifx < 0 \\ x & ifx \geq 0 \end{cases} \tag{9}$$

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{10}$$

The output of convolution layers is passed to an average pooling layer to calculate an aggregated value of extracted features using Equation 11.

$$h_j^l(x, y) = \frac{1}{k} \sum_{\bar{x} \in \mathbb{N}(x), \bar{y} \in \mathbb{N}(y)}^{k} h_j^{l-1}(\bar{x}, \bar{y}) \tag{11}$$

where $h_j^{l-1}(\bar{x}, \bar{y})$ is output matrix from the previous layer and $j = 0, 1, 2, \ldots, k$, $(\bar{x}, \bar{y})$ are mean of input features and k is kernel size.

RMSProp is used as an optimization function for each training round. Derivation, as follows, represents the gradient descent methods, where β_1, β_2 are decay rates and g_t is the gradient at time t.

$$Mean = m_w^{t+1} = \beta_1 m_w^t + (1 - \beta_1) \nabla_w g_t \tag{12}$$

$$Variance = v_w^{t+1} = \beta_2 m_w^t + ((1 - \beta_2) g_t^2)^2 \tag{13}$$

$$\widehat{m}_w = \frac{m_w^{t+1}}{1 - \beta_1^{t+1}} \tag{14}$$

$$\widehat{v}_w = \frac{(v_w^{t+1}}{1 - \beta_2^{t+1}} \tag{15}$$

$$w^{t+1} = w^t - \eta \frac{\widehat{m}_w}{\sqrt{\widehat{v}_w} + t} \tag{16}$$

To find loss(W) of the local model on worker C_i' mean squared error is used using the formula

$$Loss = L = \frac{1}{n} \sum_{i}^{n} (y_i, \widehat{y}_i) \tag{17}$$

where n is the number of devices $y_i = labels, \widehat{y}_i =$ predicted labels. The locally trained model is evaluated on validation of dataset $V_i = x_i, y_i$ where x=input image and y=image label. The algorithm of local training shows the input parameters required to train the model. Moreover, the cost value of each training round is sent to the broker to start the aggregation process.

Algorithm 1 Local Training

Inputs:
N number of data samples
I max number of local iterations
η learning rate
Output: Minimum cost value L

```
B ⟵ split the local dataset into the batches of size N
for each local iteration i = 1 ⟶ E do
    Shuffle training set randomly
    for batch B ∈ D do
        W ⟵ W − η(L(W; x^(i), y^(i)))
    end for
end for
```

Each training round ends with a cost value $L_{(i)}$and an accuracy value acc_i, which is sent to the aggregation server

along with a locally trained model M_i. Next, global training is performed based on the loss value received from each worker such that the aggregation server has n loss values given by the set $L_i' = \{l_1', l_2', \ldots, l_i'\}$ where $i = length(C') \in N$. Model M_i has the highest accuracy l_i', the value is selected as a global model and the respective model is selected as a global model to be sent to each worker for iteration $i+1$. The global optimization function is given by the formula:

$$Accuracy = max(accuracy_i') \tag{18}$$

where $i = \{1, 2, 3, \ldots, n\}$ and $accuracy_i'$ represents the accuracy obtained from cloudlet i after running each local iteration. A confusion matrix is used to determine model accuracy using the equation:

$$Accuracy = \frac{TP + TN}{Total} \tag{19}$$

Algorithms 2 (CONFLA) and 3 (CACFLA) are the two global aggregation methods used in this research work. CONFLA aggregates each layer of client NN and sends the aggregated model for the next training round. However, CACFLA checks the validation accuracy of all local models and selects the model with the highest accuracy as the global model.

Algorithm 2 Federated Averaging

Inputs:
N of cloudlets in Federation
K number of clients
B local data size
E number of local iterations
η learning rate
Output: Federated weights W

Broker averaging
Initialize w_0 with random values
for iteration $t = 1, \ldots, 10$ **do**
  $m \longleftarrow a$ random subset of $max(C.K, 1$ users
  $S_t \longleftarrow$ select subset of clients K such that $\S_t \subseteq K$
  for each client $K \in S_t$ **do**
    $W_{t+1}^K \longleftarrow ClientUpdate(K, W_t)$
    $W_{t+1}^K \longleftarrow \sum_{t=1}^{K} \frac{n_k}{n} W.t + 1$
  end for
end for
ClientUpdate(K,W)
$\mathcal{B} \longleftarrow$ split the local dataset into batches of size B
for each local iteration $i = 1 \longrightarrow E$ **do**
  for batch $b \in \mathcal{B}$ **do**
    $W \longleftarrow W - \eta.l(W, b)$
  end for
end for
Return W to broker

Algorithm 3 Global Averaging

Inputs:
N of Cloudlets in federation
E number of global iterations
η Learning rate
Local accuracy values acc'
Output: Feature weights W

Initialize w_0 with random values
for iteration $t = 1, \ldots, 10$ **do**
  $m \longleftarrow a$ random subset of max (C') users
  for each client K **do**
    $W_{t+1}^K \longleftarrow Getmodel(K_i)$
  end for
end for
$Accuracy_i' \longleftarrow max(K, L')$
$K_i \longleftarrow GetClient(L_i)$
$W_{t+1}^K \longleftarrow Getmodel(K_i)$
for each client K **do**
  $SendUpdate(K, W_{t+1}^K)$
end for
Max $Accuracy(k, L)$
for all accuracy values **do**
  $acc_i' \longleftarrow$ index of max accuracy value from K'
end for
$GetClient(K, L_i)$
for all Client K **do**
  $K_i \longleftarrow$ client address at index L_i'
end forReturn W to server

IV. RESULTS AND DISCUSSIONS

A. EXPERIMENTAL SETUP

For experimentation, the setup of CONFLA and CACFLA are established and the CNN model training is performed on the COVID-19 dataset [48]. All experiments are repeated five times, for both CONFLA and CACFLA, and the best results are reported in this paper. For both setups, three cloudlets are used. For CONFLA setup, all cloudlets are directly connected to the remote cloud virtual machine for which Amazon cloud services are used. For CACFLA, all cloudlets reside in close proximity to the user. All cloudlets are connected to the broker which is connected to the remote cloud virtual machine. Each cloudlet is configured with 4 cores, 8GB of RAM, 40GB of storage space, and the Ubuntu 14.04 LTS operating system. The virtualization environment is set up using VMware on client devices and EXS 6.0 in the data center. The detailed hardware specifications of all the cloudlets are given in Table 3. Metropolitan Area Network (MAN) is set up and all the cloudlets are connected to different ISPs.

TABLE 3. Specifications of testbed devices.

	RAM (GB)	HDD (GB)	Frequency (GHz)	CPU Cores	ISPs
Broker	8	40	2.8	4	PTCL
Cloudlet A	8	40	1.99	4	Huawei
Cloudlet B	8	40	1.99	4	Strom Fiber
Cloudlet C	8	40	1.99	4	PTCL

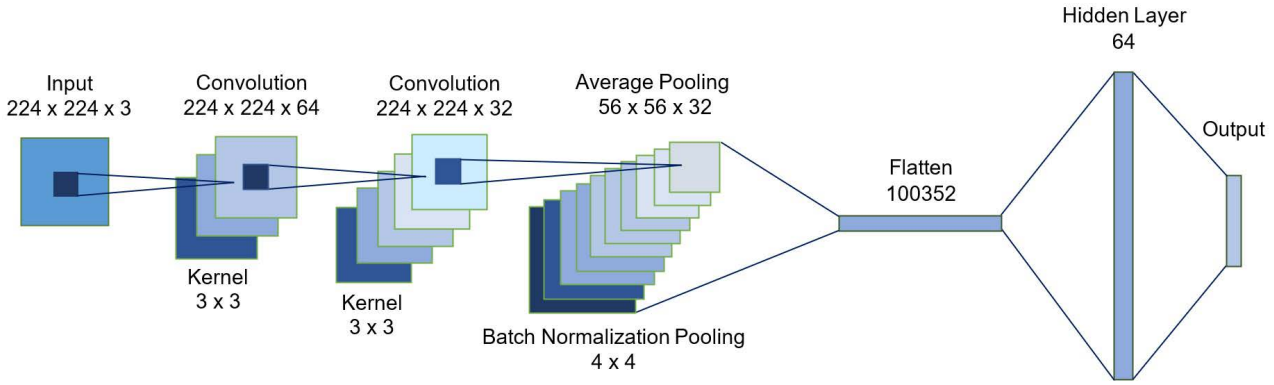


FIGURE 3. Architecture of the proposed deep learning model.

B. DEEP LEARNING MODEL

Figure 3 shows the architecture of our proposed DL model. Input X-ray images of size 224 × 224 × 3 are fed to the model. The first two values represent the width and height of the X-ray respectively and the third value represents the color channels (red, green, and blue) used, also known as the depth. Hence, a single neuron in the hidden layer has 224 × 224 × 3 = 150,528 weights. Two CNN layers of size 224 × 224 are used. The filter size of the two CNN layers is 64 and 32 respectively. The average Pooling layer performs the downsampling, reducing the dimensions to 56 × 56 × 32, after which the flattened layer converts the matrix into a single one-dimensional array. This vector is fed to a hidden layer that classifies the image and gives the final output.

The model predicts COVID-19 differentiating it from other types of pneumonia. The learning rate of the CNN model is initially defined as 0.005 which changes based on weights for each epoch. Local training epochs are decided dynamically based on the dataset of each client. However, the number of global training rounds is 10. RMSprop is used as a global and local optimizer and MSE is used as a local and global loss function. In the first step, a client opens a socket connection with the server. The server then assigns a unique id to the client and starts a new thread. After the client is successfully connected to the broker, X-ray images are resized into 128 × 128 images. In the next step cardinality of each dataset is sent to a broker to figure out the scaling factor based on that scaling factor weights are assigned to each CNN model and the model is sent to the client for training. At the end of the training round, each client sends its trained model back to the broker. The broker then selects a client within the federation with optimized resources to send the trained models from all clients to aggregate them and generate the global model for the next iteration. The global model is sent back to the broker for evaluation based on accuracy. If model accuracy increases from the last round, then the new model is selected otherwise the old model is again sent to all clients for the next local training round. At the end of the global epochs, each client is sent the updated model and the connection is terminated.

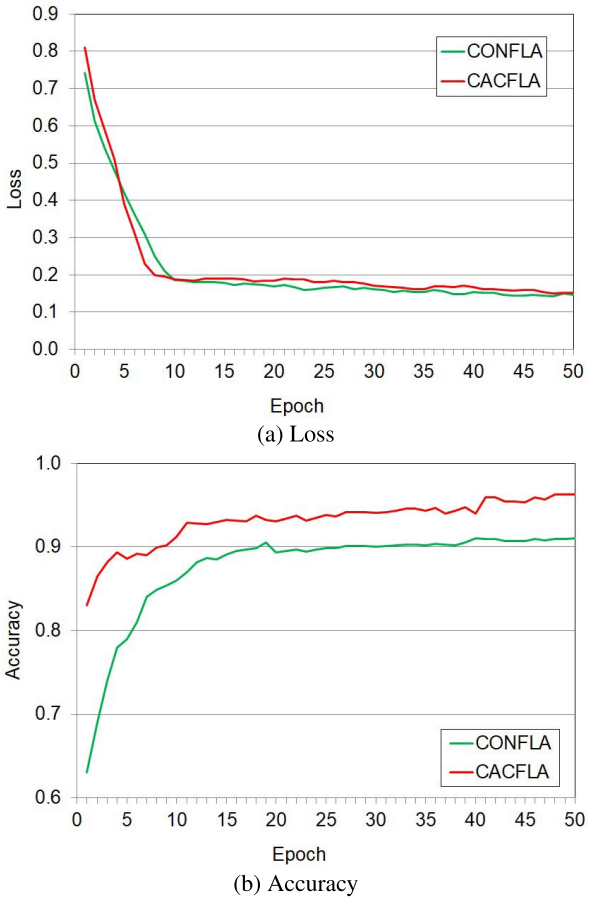


FIGURE 4. Accuracy and loss comparison of CONFLA and CACFLA.

C. RESULTS

The CNN model is initially trained for 50 epochs using CONFLA and CACFLA. The accuracy and loss for both approaches are shown in Figure 4. As observed in Figure 4a, it takes less than 10 epochs for model convergence for both approaches. So, for the rest of the experiments, the training rounds consist of 10 global epochs.

It is also observed that CACFLA has a slightly less convergence time than CONFLA. This is because it takes the CNN model from all clients in the federation and aggregates

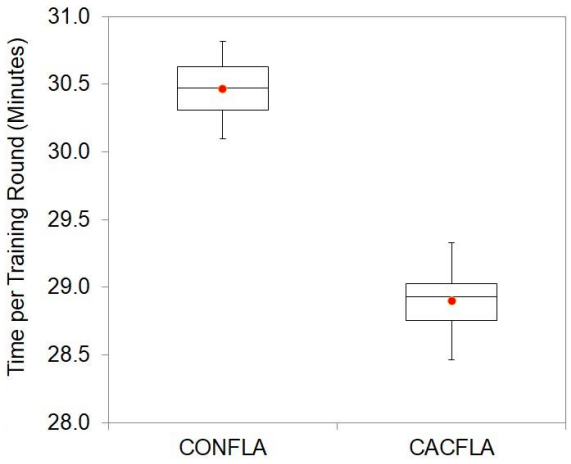


FIGURE 5. Comparison of time taken per training between CONFLA and CACFLA.

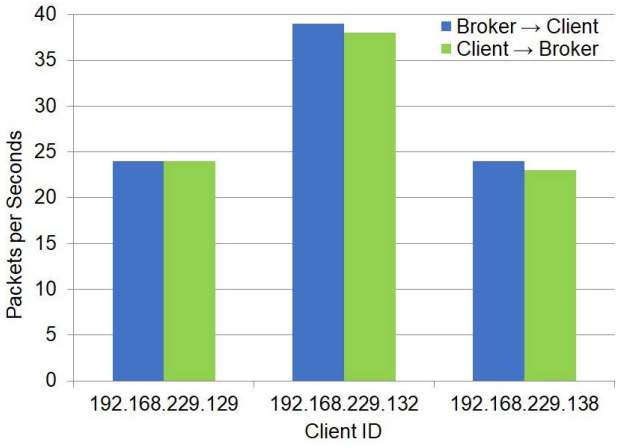


FIGURE 6. Clients' communication statistics with broker.

all the models to obtain a generic CNN model for the next epoch. The validation accuracy of CACFLA is 5% more as compared to CONFLA. This shows that choosing the best selection model in CACFLA gives better accuracy with a comparatively similar loss value as compared to CONFLA.

The model is further evaluated based on the time taken to complete ten training rounds in the FL setup. Figure 5 shows a comparison of the time taken by CONFLA and CACFLA to complete a single training round. The red dot represents the mean value while the end of the whiskers shows the maximum and minimum values of time taken per training round. In CONFLA, the server is placed at a distant Amazon cloud and each client in the federation sends its trained model to a distant cloud for aggregation. Since the server is placed at a greater distance, CONFLA takes more time to complete a training round. Therefore, as observed from Figure 5, having the server in the local vicinity of the user, as is the case with CACFLA, reduces the time to complete a training round. CONFLA takes 304.64 minutes to complete the ten training rounds while CACFLA takes 289.03 minutes, showing a 5% decrease in latency.

Figure 6 shows the statistics of data transmission between the broker and the three clients in the federation in terms of the total number of packets transmitted per second. It is observed that most of the communication of the broker is with the client 192.168.229.132. This is because it has the minimum number of COVID-19 X-ray images and due to this, its resources are mostly free for aggregation tasks as it requires less time to complete a single training round.

V. CONCLUSION

This research work addresses the challenges of computational and communication resources faced by federated learning using context-aware cloudlet federation. The challenges include latency, loss, accuracy, training time, and data bias that are present in the conventional cloud-based federated learning environment. CACFLA, a context-aware federated learning approach is proposed that offloads aggregated tasks to nearby clients based on the availability of computational resources. CACFLA is compared with CONFLA which is a state-of-the-art conventional federated learning approach. In both approaches, the number of local iterations and learning rate is decided dynamically, based upon the data each client possesses. CACFLA is evaluated based on latency, accuracy, and convergence time. Results show that CACFLA reduces the latency by 5% and yields 5% better accuracy as compared to CONFLA. Additionally, the convergence time of the CACFLA is also reduced as compared to CONFLA.

In future work, the system will be extensively tested on some other datasets, including the KEEL dataset [49]. Moreover, the time taken by the model to converge will also be improved using model compression techniques based on the analysis of different datasets. We also aim to improve the model aggregation method to increase the local training accuracy.

REFERENCES

[1] A. Kaur, V. P. Singh, and S. S. Gill, "The future of cloud computing: Opportunities, challenges and research trends," in *Proc. 2nd IEEE Int. Conf. I-SMAC (IoT Social, Mobile, Anal. Cloud)*, Palladam, India, Aug. 2018, pp. 213–219.

[2] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intell. Converged Netw.*, vol. 1, no. 2, pp. 181–198, 2020.

[3] M. Z. Nayyer, I. Raza, and S. A. Hussain, "CFRO: Cloudlet federation for resource optimization," *IEEE Access*, vol. 8, pp. 106234–106246, 2020.

[4] H. Liu, X. Long, Z. Li, S. Long, R. Rong, and H.-M. Wang, "Joint optimization of request assignment and computing resource allocation in multi-access edge computing," *IEEE Trans. Services Comput.*, early access, Jun. 6, 2022, doi: 10.1109/TSC.2022.3180105.

[5] R. Alakbarov, "Cloudlet selection strategy according to the types of applications in cloud networks," in *Proc. 5th Int. Conf. Comput. Informat. (ICCI)*, Cairo, Egypt, Mar. 2022, pp. 200–203.

[6] S. Tijani. (2020). *Federated Learning: A Step by Step Implementatin in TensorFlow*. Accessed: Apr. 1, 2022. [Online]. Available: https://towardsdatascience.com/federated-learning-a-step-by-step-implementation-in-tensorflow-aac568283399

[7] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, Nov. 2019.

[8] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," 2019, *arXiv:1906.06268*.

[9] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Global Internet of Things Summit*. Geneva, Switzerland: Springer, Jun. 2017, pp. 1–6.

[10] D. Conway-Jones, T. Tuor, S. Wang, and K. K. Leung, "Demonstration of federated learning in a resource-constrained networked environment," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Washington, DC, USA, Jun. 2019, pp. 484–486.

[11] Y. Ma, Y. Zhao, J. Liu, X. He, B. Wang, S. Fu, J. Yan, J. Niu, J. Zhou, and B. Luo, "Effects of temperature variation and humidity on the death of COVID-19 in Wuhan, China," *Sci. Total Environ.*, vol. 724, Jul. 2020, Art. no. 138226.

[12] R. Kumari, S. Kumar, R. C. Poonia, V. Singh, L. Raja, V. Bhatnagar, and P. Agarwal, "Analysis and predictions of spread, recovery, and death caused by COVID-19 in India," *Big Data Mining Anal.*, vol. 4, no. 2, pp. 65–75, Jun. 2021.

[13] P. D. Waggoner, R. Y. Shapiro, S. Frederick, and M. Gong, "Uncovering the online social structure surrounding COVID-19," *J. Social Comput.*, vol. 2, no. 2, pp. 157–165, Jun. 2021.

[14] A. Agarwal, S. Sharma, V. Kumar, and M. Kaur, "Effect of E-learning on public health and environment during COVID-19 lockdown," *Big Data Mining Anal.*, vol. 4, no. 2, pp. 104–115, Jun. 2021.

[15] I. Feki, S. Ammar, Y. Kessentini, and K. Muhammad, "Federated learning for COVID-19 screening from chest X-ray images," *Appl. Soft Comput.*, vol. 106, Jul. 2021, Art. no. 107330.

[16] S. Naz, K. T. Phan, and Y. P. Chen, "A comprehensive review of federated learning for COVID-19 detection," *Int. J. Intell. Syst.*, vol. 37, no. 3, pp. 2371–2392, Mar. 2022.

[17] Z. Li, X. Xu, X. Cao, W. Liu, Y. Zhang, D. Chen, and H. Dai, "Integrated CNN and federated learning for COVID-19 detection on chest X-ray images," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Jun. 20, 2022, doi: 10.1109/TCBB.2022.3184319.

[18] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Comput. Surveys*, vol. 55, no. 3, pp. 1–37, Apr. 2023.

[19] R. Jain, M. Gupta, S. Taneja, and D. J. Hemanth, "Deep learning based detection and analysis of COVID-19 on chest X-ray images," *Appl. Intell.*, vol. 51, pp. 1690–1700, Oct. 2021.

[20] I. H. Alsohaimi, "Analytical detection methods for diagnosis of COVID-19: Developed methods and their performance," *Biotechnol. Biotechnol. Equip.*, vol. 35, no. 1, pp. 196–207, Jan. 2021.

[21] J. Pang, Y. Huang, Z. Xie, J. Li, and Z. Cai, "Collaborative city digital twin for the COVID-19 pandemic: A federated learning solution," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 759–771, Oct. 2021.

[22] Y. Zhang, H. Huang, L.-X. Yang, Y. Xiang, and M. Li, "Serious challenges and potential solutions for the industrial Internet of Things with edge intelligence," *IEEE Netw.*, vol. 33, no. 5, pp. 41–45, Sep. 2019.

[23] N. Nascimento, P. Alencar, C. Lucena, and D. Cowan, "A context-aware machine learning-based approach," in *Proc. 28th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, Markham, ON, Canada, Oct. 2018, pp. 40–47.

[24] A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing, "Hierarchical and distributed machine learning inference beyond the edge," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*, Banff, AB, Canada, May 2019, pp. 18–23.

[25] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad, "Edge computing for smart health: Context-aware approaches, opportunities, and challenges," *IEEE Netw.*, vol. 33, no. 3, pp. 196–203, May/Jun. 2019.

[26] M. Alnemari and N. Bagherzadeh, "Efficient deep neural networks for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Milan, Italy, Jul. 2019, pp. 1–7.

[27] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[28] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018, *arXiv:1812.07210*.

[29] Z. Feng, S. George, J. Harkes, P. Pillai, R. Klatzky, and M. Satyanarayanan, "Edge-based discovery of training data for machine learning," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Seattle, WA, USA, Oct. 2018, pp. 145–158.

[30] A. Brenon, F. Portet, and M. Vacher, "Context feature learning through deep learning for adaptive context-aware decision making in the home," in *Proc. 14th Int. Conf. Intell. Environ. (IE)*, Rome, Italy, Jun. 2018, pp. 32–39.

[31] Z. Chai, H. Fayyaz, Z. Fayyaz, A. Anwar, Y. Zhou, N. Baracaldo, H. Ludwig, and Y. Cheng, "Towards taming the resource and data heterogeneity in federated learning," in *Proc. USENIX Conf. Oper. Mach. Learn. (OpML)*, Santa Clara, CA, USA, May 2019, pp. 19–21.

[32] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 1205–1221, Jun. 2019.

[33] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Mining Anal.*, vol. 3, no. 3, pp. 181–195, Sep. 2020.

[34] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[35] V. W. Anelli, Y. Deldjoo, T. D. Noia, and A. Ferrara, "Towards effective device-aware federated learning," in *Proc. Int. Conf. Italian Assoc. Artif. Intell.*, Rende, Italy, Nov. 2019, pp. 477–491.

[36] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Paris, France, May 2019, pp. 1–6.

[37] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[38] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun. (MECOMM SIGCOMM)*, Budapest, Hungary, Aug. 2018, pp. 31–36.

[39] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learning-based computation offloading optimization in edge computing-supported Internet of Things," *IEEE Access*, vol. 7, pp. 69194–69201, 2019.

[40] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2017.

[41] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68–74, Sep. 2019.

[42] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 96–103, Sep. 2019.

[43] W. Zhang, Z. Li, and X. Chen, "Quality-aware user recruitment based on federated learning in mobile crowd sensing," *Tsinghua Sci. Technol.*, vol. 26, no. 6, pp. 869–877, Dec. 2021.

[44] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[45] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5986–5994, Jul. 2020.

[46] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," 2019, *arXiv:1905.06641*.

[47] M. Breitbach, D. Schäfer, J. Edinger, and C. Becker, "Context-aware data and task placement in edge computing environments," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Kyoto, Japan, Mar. 2019, pp. 1–10.

[48] J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Q. Duong, and M. Ghassemi, "COVID-19 image data collection: Prospective predictions are the future," 2020, *arXiv:2006.11988*.

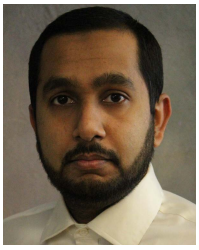
[49] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Log. Soft Comput.*, vol. 17, pp. 255–287, Jun. 2011.

SANA LATIF received the B.S. degree in software engineering and the M.S. degree in computer science from COMSATS University Islamabad, Lahore Campus, Pakistan. She is currently serving as a Visiting Lecturer with the Department of Computer Science, COMSATS University Islamabad, and the University of Central Punjab, Lahore. Her research interests include cloudlet computing, mobile edge computing, and federating learning.

MUHAMMAD ZIAD NAYYER received the M.S. degree in computer science from Government College University (GCU), Lahore, Pakistan, in 2011, and the Ph.D. degree in computer science from COMSATS University Islamabad, Lahore Campus. He is serving as an Assistant Professor with the Department of Computer Science, GIFT University, Gujranwala, Pakistan. He is an Active Member with the Advanced Communication Networks Laboratory. He has numerous publications on his account including impact factor journal publications and book chapters. His research interests include cloud computing, VM migration, mobile cloud computing, cloud federation, mobile edge computing, fog computing, and cloudlet computing.

IMRAN RAZA (Member, IEEE) received the B.S. and the M.Phil. degrees in computer science from Pakistan. He has been an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad, Lahore Campus, since 2003. He has authored and coauthored more than 50 journal and conference papers. He has been actively involved in simulating CERN O2/FLP upgrades. He has supervised and co-supervised many funded projects related to ICT in Healthcare. His research interests include cloud computing, mobile edge computing, SDN, NFV, wireless sensor networks, MANETS, QoS issues in networks, and routing protocols. He is a member of ACM.

SYED ASAD HUSSAIN (Member, IEEE) received the master's degree from Punjab University Lahore, Pakistan, then he proceeded to the U.K. for his master's from the University of Wales, Cardiff, UK, and the Ph.D. degree from Queen's University Belfast, U.K. He has been a Professor with the Computer Science Department, COMSATS University Islamabad (CUI), Lahore Campus, Pakistan, since 2010, where he has served as the Department Head, from 2008 to 2010 and from 2011 to 2017. He was the Dean of the Faculty of Information Sciences and Technology, CUI, Pakistan, from 2015 to 2020. He has been a Leader with the Communications Networks Research Group, Computer Science Department, CUI, since 2005. He was funded for his Ph.D. by Nortel Networks U.K. He has taught at Queen's University Belfast, U.K., Lahore University of Management Sciences (LUMS), and the University of the Punjab, Pakistan. Currently, he is supervising Ph.D. students with CUI and split-site Ph.D. students with Lancaster University, U.K., in the areas of cloud computing and cybersecurity. He is actively involved in collaborative research with CERN in Switzerland and different universities of the world such as Cardiff University, U.K., Lancaster University, U.K., Dalhousie University, Canada, and Charles Sturt University, Australia. He is supervising funded projects as a Principal Investigator in healthcare systems. He has authored and coauthored more than 85 journal and conference papers and has written a book titled *Active and Programmable Networks for Adaptive Architectures and Services* (Taylor and Francis, USA). He was awarded the Prestigious Endeavour Research Fellowship by the Australian Government for his post-doctorate at the University of Sydney, Australia, in 2010. He regularly reviews international journals and conference papers.



M. HASAN JAMAL received the B.S. degree in computer and information engineering from International Islamic University Malaysia, in 2005, the M.S. degree in computer engineering from the University of Engineering and Technology, Lahore, Pakistan, in 2008, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2015. From 2006 to 2010, he was a Research Associate and a Senior Research Associate at the Al-Khawarizmi Institute of Computer Science, UET, Lahore. He was a Graduate Technical Summer Intern at the Sandia National Laboratory, Albuquerque, NM, USA, in Summer 2015. Since 2016, he has been an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad, Lahore campus, Pakistan. His research interests include parallel and distributed systems, performance engineering, and scientific computing. He was a recipient of the Fulbright Scholarship for his Ph.D. studies.

SOOJUNG HUR received the B.S. degree from Daegu University, Gyeongbuk, South Korea, in 2001, the M.S. degree in electrical engineering from the San Diego State University, San Diego, in 2004, and the M.S. and Ph.D. degrees in information and communication engineering from Yeungnam University, South Korea, in 2007 and 2012, respectively. She is working as a Research Professor with the Mobile Communication Laboratory, Yeungnam University. Her current research interests include the performance of mobile communication, indoor/outdoor location, and unnamed vehicle.

IMRAN ASHRAF received the M.S. degree in computer science from the Blekinge Institute of Technology, Karlskrona, Sweden, in 2010, and the Ph.D. degree in information and communication engineering from Yeungnam University, Gyeongsan, South Korea, in 2018. He has worked as a Postdoctoral Fellow at Yeungnam University. He is currently working as an Assistant Professor with the Information and Communication Engineering Department, Yeungnam University. His research interests include indoor positioning and localization, advanced location-based services in wireless communication, smart sensors (LIDAR) for smart cars, and data mining.