Internet of Things 22 (2023) 100784

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Review article

# FedSDM: Federated learning based smart decision making module for ECG data in IoT integrated Edge–Fog–Cloud computing environments

Shinu M. Rajagopal [a,*], Supriya M. [a], Rajkumar Buyya [b]

[a] Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru, Amrita Vishwa Vidyapeetham, India
[b] CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne, Australia

ARTICLE INFO

*Keywords:*
Federated learning
Cloud computing
Fog computing
Edge computing
Internet of things
Public healthcare
Real-time systems
Resource management

ABSTRACT

Massive data collection in modern systems has paved the way for data-driven machine learning, a promising technique for creating reliable and robust statistical models. By combining the data into centralized storage to develop a reliable learning model, there are concerns with privacy, ownership, and strict rules. It is self-evident that the samples in the typical machine learning centralized server paradigm have vastly different probability distributions of data supplied by each user. As a result, the typical model needs to be personalized for critical medical applications, and the deployment needs an efficient mechanism that can adapt to varying user inputs. Due to the heterogeneous and dynamic nature of critical medical IoT applications in such Edge/Fog scenarios, the privacy of patients become a crucial problem. Federated Learning, the model trained on diversity helps in addressing these concerns when used. This paper proposes the integration of Federated Learning for distributed Edge–Fog–Cloud architecture in the IoT smart healthcare sector. This paper presents FedSDM, the Federated Learning-based Smart Decision Making framework for the ECG data in microservice-based IoT medical applications. This proposal makes use of the advantages of Edge/Fog computing for real-time critical applications. It deploys the Federated Learning model at the Edge, Fog, and Cloud layers for performance comparison. The parameters considered for performance evaluation are energy consumption, network usage, cost, execution time, and latency. The proposed method shows that Edge-based deployment outperforms Fog and Cloud in terms of energy consumption, network usage, cost, execution time, and latency (i.e.) 0.3%, 2%, 15%, 11%, and 3% when compared with Fog and 1.6%, 31%, 41%, 24 % and 85% against Cloud respectively.

## 1. Introduction

The unprecedented technological developments over the last two decades have accelerated data growth, which has led to data privacy issues when stored in centralized systems. This is visible in many domains, especially in the healthcare sector, where hospitals still store patients' sensitive information in centralized repositories. A recent study on cyber security statistics disclose that data breaches in the first half of 2020 alone revealed 36 billion records [1]. This motivates the introduction of the next level of artificial intelligence, which works on the concept of data privacy as its foundation to address the users' concern about the privacy of available data such as personally identifiable details, payment details, protected health information, confidential information, and

* Corresponding author.
*E-mail addresses:* mr_shinu@blr.amrita.edu (S.M. Rajagopal), m_supriya@blr.amrita.edu (Supriya M.), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.iot.2023.100784
Received 5 December 2022; Received in revised form 8 April 2023; Accepted 11 April 2023
Available online 14 April 2023

others [2]. Due to the heavy movement of data into and out of the Cloud, two additional challenges arise with Cloud/centralized-based techniques namely latency and data transfer cost. Users or end devices may be unwilling to disclose private data to preserve privacy or conserve their phone's limited bandwidth/battery capacity [2]. In such scenarios, Federated Learning enables smarter models with less latency, and reduced battery usage while ensuring privacy. Defense, telecommunications, Internet of Things (IoT), and pharmaceutics are just a couple of applications where it can be used.

Federated Learning (FL) is a machine learning methodology that involves training an algorithm across numerous decentralized servers or Edge devices while retaining data samples locally and preventing data transmission. Using data from tens to millions of remote devices helps build a global statistical model. FL allows devices like mobile phones to develop a shared prediction model while retaining all the training data on the end device, removing the requirement to store it in the Cloud. Training occurs while the end device is idle, plugged in, and connected to a WiFi network, ensuring that the performance is unaffected. Because of the rising computational capacity of these devices, as well as issues about transferring private information, storing data locally, and pushing network computing to the Edge are becoming more desirable. For IoT applications, FL can provide several significant advantages such as data privacy enhancement, low-latency network communication, enhanced learning quality, etc. [3].

IoT devices produce enormous amounts of data that need latency-sensitive processing, which cannot be accomplished when applications are placed in distant Cloud data centers [4]. Hence Cloud computing seems to be infeasible for delay-sensitive medical real-time IoT applications [5]. Fog or Edge computing, a Cloud extension, address smart IoT systems' needs and provide better resource management [6]. It allows seamless connection between smart medical devices and computational resources to have more control over data privacy and security [7]. It also supports critical latency-sensitive IoT applications that require faster responses with reduced consumption of energy and bandwidth [8]. In some medical emergencies, making a decision using the appropriate resources in a fraction of a minute has a direct impact on the life of the patients [9].

The mobility of end IoT devices causes the migration of requested application services from one computing node to another to maintain the promised QoS. As the user relocates from one place to another, the proximity to a Fog or Edge service may change, and hence user mobility restricts the benefits in practice [10]. In real-time, IoT device mobility can impact Fog computing systems when they change access points repeatedly. In addition to meeting the demands of latency and bandwidth on the network, Edge/Fog offers intelligent services at the Edge to fulfill the vital need of IoT applications in real-time.

IoT applications use various technologies to connect, manage, and operate IoT smart devices. Microservices, a type of service-oriented architecture, have attracted much interest nowadays [11]. Each microservice is responsible for a single sub-task or service, requiring fewer compute resources and lower communication overhead. Based on the resource availability and workload of Fog nodes, microservices can scale up and down dynamically due to loosely coupled modules provide advantages such as independent deployment, scalability, and fault isolation [12].

This paper aims at the development of FedSDM, a Federated Learning-based Smart Decision Making module for ECG data in microservice-based IoT medical applications.

The following are the research contributions of this paper:

- Design of an early warning system for ECG anomalies using Smart Decision Making module
- Integration of Federated Learning method to critical healthcare applications for the privacy of the end-user data
- Identification of an appropriate placement policy for the Federated Learning module in Edge, Fog, and Cloud layers

The rest of this paper is organized as follows. Sections 2 and 3 discuss Background and related work. Section 4 presents the proposed method followed by the experimental setup in Section 5. Section 6 summarizes the results, and Section 7 concludes the paper.

## 2. Background and motivation

The concept of intelligent healthcare involves utilizing AI to learn and analyze patient data. However, it can be challenging to find large and diverse datasets to train machine learning models in individual medical centers. This means that traditional centralized AI methods require sensitive data to be moved from medical facilities to data centers, which not only increases the demand for communication resources and energy, but also violates privacy. This has become a significant obstacle in promoting scientific collaboration between trans-national clinical medical research centers. A distributed AI technique known as Federated learning has emerged that enables the cooperative training of ML models without the sharing of patient data. Federated learning may prove to be an advantageous method for facilitating Internet of Things-based intelligent applications [13–15].

Due to the necessity for real-time processing, low latency, and privacy considerations, Edge/Fog computing is becoming more and more significant for medical applications [16]. Edge computing can limit the quantity of data that must be transferred to centralized servers or the cloud by processing and analyzing the data closer to the data's source, thereby reducing network traffic and delays [17]. Additionally, Edge computing can help address privacy concerns by keeping sensitive data within a local network and limiting access to authorized users only. In medical applications, where time and accuracy are critical, and privacy is essential, Edge computing has become a necessity [18,19].

Real-time ECG abnormality detection is one of the applications in medicine that has several advantages for patient care. First and foremost, it allows healthcare providers to quickly identify and respond to cardiac abnormalities, potentially saving lives. Early detection and treatment of cardiac abnormalities can prevent more serious and costly health issues down the road. Moreover, real-time ECG anomaly detection can help reduce healthcare costs and improve patient outcomes. By continuously monitoring ECG signals in real-time, the system can immediately detect anomalies and alert healthcare providers, who can take action to diagnose

**Table 1**
Comparison of aggregation algorithms in federated learning.

| Algorithm | Complexity | Accuracy | Convergence | Cost | Speed |
|---|---|---|---|---|---|
| FedAvg [20] | Low | High | Slow | Low | High(simple datasets) |
| FedMA [21] | Moderate | Moderate | Moderate | Moderate | Moderate(complex datasets) |
| FedProx [2] | Moderate | Moderate | Moderate | Low | Moderate(complex datasets) |
| FedPer [22] | Low | High | Moderate | Low | High(simple datasets) |
| FedDist [23] | High | Moderate | Moderate | Moderate | Moderate(complex datasets) |
| EdgeFed [24] | High | High | High | High | High(simple datasets) |

and treat the patient. Another benefit is that real-time ECG anomaly detection can improve the accuracy of diagnoses. In some cases, anomalies may be missed or misinterpreted when relying on visual inspections alone. With automated detection, the system can analyze the ECG signals with greater precision, reducing the risk of errors and false negatives. Additionally, real-time monitoring can help identify potential issues before they become acute, reducing the likelihood of hospitalizations and emergency room visits. Overall, real-time ECG anomaly detection has the potential to improve patient care, increase accuracy, and reduce healthcare costs, making it a valuable tool in healthcare. Since the healthcare issues related to ECG anomaly detection in microservice-based IoT systems are not sufficiently addressed by existing research on Edge/Fog/Cloud Federated learning approaches, we were motivated to do this study.

## 3. Related work

Federated Learning is appropriate for Edge/Fog/Cloud computing applications and can use the computation power of servers and data gathered from widely scattered devices. Effective aggregation of client models is essential to create a generalized global model. The fundamental approach is aggregating models from the distributed clients and obtaining a new general global average model. The resultant model is then distributed to clients again for further training. Federated Learning makes use of different aggregation strategies for global model update. The following paragraphs discuss state of the art in aggregation methods in Federated Learning, FL in Edge/Fog/Cloud IoT applications, anomaly detection and Smart Decision Making modules implementations in FL, smart healthcare applications using FL.

### *3.1. FL aggregation methods*

The literature proposes FedAvg as a privacy, security-preserving, and efficient communication aggregation algorithm for FL over-Edge devices. FedAvg assumes uniform involvement from all participants and excludes clients responding slowly [20]. The FedMA aggregation approach's foundation is a layer-wise learning strategy that matches and merges nodes with comparable weights. Independently trained layers interact with the server [21]. FedProx addresses the heterogeneity issue in federated networks by allowing each participant device to execute a different amount of work. It incorporates partial information from stragglers and adds a proximal term to account for heterogeneity, which promises a steady and precise convergence behavior [2]. The principle of the FedPer approach is that the model is divided into personalized and base layers. While the personalized layers are not communicating with the server, the base layers are aggregated using transfer learning methodologies by the federated server [22]. FedDist is a Federated Learning aggregation algorithm based on the Euclidean distance dissimilarity measurement. This algorithm includes a few advantages of FedAvg, and FedMA [23]. Separating the local update process from the global aggregation results in a decrease in mobile devices' overall communication and computation costs. Also, in varying bandwidth conditions, empirical testing shows that the suggested EdgeFed is comparatively more efficient than state-of-the-art algorithms, with a decrease in the computational cost and the cost of interconnection for mobile devices. This is achieved by offloading a few calculations from mobile clients to the Edge server [24].

A comparison of the above-discussed aggregation algorithms is presented in Table 1. This paper uses FedAvg for the proposed approach due to its easy deployment and less complicated implementation on Edge/Fog devices, resulting in reduced communication overhead.

### *3.2. Federated Learning in Edge/Fog/Cloud IoT applications*

Xia et al. give new insight into Federated Learning's Edge applications, development tools, communication effectiveness, privacy & security, scheduling, and migration [25]. Imteaj et al. examine the difficulties and problems of implementing FL in an IoT scenario [26]. Yu et al. offer a neural-structure-aware resource management approach with module-based Federated Learning, in which mobile clients are allocated with various sub-networks of the global model based on the condition of their local resources using both white box and black box approaches. Experiments show the effectiveness and flexibility of the strategy in utilizing resources [27]. Nguyen et al. evaluate the potential of FL for enabling a vast range of IoT services, including caching and data offloading for IoT devices, attack detection, location, crowd-sensing on mobile devices, and IoT privacy and safety. Additionally, a thorough analysis of the usage of FL in various critical IoT applications such as smart healthcare, unmanned aerial vehicles, smart transportation, smart cities, and smart industry is discussed [28].

A greedy heuristic method proposed in literature help in choosing the best fog node to act as a global aggregator. This helps in communication between the Edge and the Cloud and can lower the reliance on server-based execution. This FogFL architecture uses fog nodes to decrease energy consumption and communication latency of resource-constrained edge devices without influencing the rate of convergence of the global model, hence enhancing system dependability. Extensive deployment and testing claim that, in addition to fewer global aggregation rounds, FogFL holds 85% less energy and 92% less communication delay than state-of-the-art [29]. Zhou et al. utilize the combination of Paillier homomorphic encryption and blinding against model attacks to achieve the security aggregation of model parameters and enable IoT device data to fulfill differential privacy in resisting data attacks. Additionally, the proposal validates the scheme's ability to withstand collusion attempts performed by numerous malevolent actors, guaranteeing both model and data security. The study implemented on the Fashion-MNIST dataset claims that the proposed technique is effective for real-world applications as well [30]. EdgeFed, draws inspiration from Edge computing and aims to enhance the learning efficiency and reduce global communication frequency. It achieves this by separating the process of updating the local model, which is done independently by mobile devices. The edge server aggregates the outputs of these devices [24].

In order to reduce the model training loss and the overall time consumption, Zaw et al. develop an energy-aware resource management for Mobile Edge computing-enabled FL that takes into account the energy constraint of mobile devices and performed solution's convergence analysis and compare its effectiveness to the conventional FL technique [31]. To deliver FL as a Service (FLaaS) to industrial customers deployed on edge devices, Hiessl et al. suggest a FL system made up of a process description and software architecture. By grouping customers into cohorts with comparable data distributions, our method addresses skewed data [32].

### 3.3. Anomaly detection in IoT applications

Hasan et al. compares the effectiveness of various machine learning models in accurately predicting attacks and anomalies in IoT systems. The machine learning algorithms evaluated include Logistic Regression (LR), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN) [33]. Abusitta et al. presents an anomaly detection method for IoT, which utilizes deep learning to capture and learn resilient and beneficial features that remain unaffected by unstable environments. The extracted features are then used by the classifier to enhance the accuracy of malicious IoT data detection. The proposed deep learning model is based on a denoising autoencoder, which is utilized to derive features that can withstand the heterogeneous environment of IoT [34]. Chatterjee et al. provide an overview of the techniques used to identify abnormalities in IoT systems, as well as a conversation about how to group different algorithms for anomaly detection [35].

#### 3.3.1. ECG anomaly detection

Andrysiak et al. proposes a technique that combines the benefits and features of sparse representation of the analyzed ECG signal with the classification characteristics of the modeled neural network, in order to create a method that is both uncomplicated and efficient [36]. To overcome the limitations of current wearable devices used in ECG detection, Gu et al. suggest a heart rhythm abnormality classification model that is both lightweight and highly accurate based on traditional convolutional neural networks and the hardware acceleration techniques [37]. Nawaz et al. introduces an intelligent system that can automatically evaluate cardiovascular activity by detecting and classifying anomalies in raw one-dimensional (1D) electrocardiogram (ECG) signals from end to end. The raw ECG data is carefully pre-processed before being stored in the cloud, and then analyzed in detail to identify any anomalies. For anomaly detection in the 1D ECG time-series signals, a deep learning-based auto-encoder (AE) algorithm is employed [38]. Ji et al. introduce a technique for detecting anomalies in univariate time series data using a long short-term memory (LSTM) algorithm. This method learns the structural characteristics of non-anomalous training data, and then applies a statistical approach to detect anomalies based on prediction error in observed data [39].

### 3.4. Smart decision making in IoT applications

Cambra et al. showcase the benefits of using a tool that utilizes data in real-time decision-making. The data includes variable rate irrigation and specific parameters derived from field and weather conditions. The decision-making system processes data obtained from periodic sampling of field parameters, vegetation indices estimated through aerial images, and irrigation events like flow level, pressure level, and wind speed. The data is analyzed using a learning prediction system combined with the Drools rule engine in making decisions [40]. Kaur et al. propose a model that employs embedded sensors within a smart industrial system to gather data and identify the different industrial activities of employees. The identified activities are classified as positive, negative, or neutral. This information is then used to make cognitive decisions for employees based on game theory. The model aims to automate the cognitive employee evaluation system and decision-making process in smart industries, and it does so effectively and efficiently [41]. Bokhari et al. aims to explore the direct and indirect connections between Artificial Intelligence (AI), Social Innovation (SI), and Smart Decision-Making (SDM). The results thus obtained help local governments to establish smart cities, where social innovation is incorporated into the decision-making process. The study also emphasizes that smart decision-making should involve social innovations and share collected data with entrepreneurs, businesses, industries, and social innovators to benefit the society and all the relevant stakeholders [42].

#### 3.4.1. SDM in smart healthcare applications

Decision support systems (DSS) aim to provide experts with timely and relevant information. They offer tools for data processing, models, and knowledge to assist experts in making more informed decisions in various scenarios [43]. Zhou et al. suggest an approach for utilizing healthcare big data and involves a framework that enables smart and proactive data processing without requiring user interventions with an aim to maximize the utilization of data in decision-making. The framework comprises of five stages: intelligent data cleaning, customized data fusion, analysis mapping, exploratory visualization analysis, and generation of decision-making reports [44]. Quasim suggest a method for evaluating the technological integration efficiency of healthcare management using a Smart Healthcare Management Evaluation and Fuzzy Decision Making approach [45]. In this proposed study, the suggested strategy for SDM anomaly detection includes the utilization of an autoencoder, which is a type of unsupervised learning technique within the realm of AI. This neural network architecture is capable of compressing input data into a lower-dimensional representation and then reconstructing the input from this representation. This AI-based SDM can be implemented across various layers, including Edge, Fog, or Cloud.

### 3.5. Federated learning in healthcare

Among many applications, the healthcare sector deserves to be prioritized in terms of service quality compared to other domains. Critical functions such as simultaneous reporting and monitoring, tracking and alerts, and remote medical aid are all possible with IoT-based apps. The center for connected health policy conducted a study that observed that remote health monitoring systems lower the re-admission rates of heart failure patients by 50 percent [46]. Machine learning will not be able to realize its full potential or, eventually, make the leap from academic study to the clinical application without access to enough data. Rieke et al. examine the major contributing causes to this problem, evaluates the challenges faced in the field of digital health and discuss how Federated Learning can provide a solution [47]. Chen et al. propose FedHealth, a system that uses federated and transfer learning to aggregate and create reasonably personalized models. The model uses homomorphic encryption to ensure that no user data is leaked [48]. The design of a new aggregation protocol uses a secure hardware component and an Ethereum-native encryption toolkit to prevent the user data from leakage [49]. Kumar et al. present a framework that collects a modest amount of data from multiple hospital sources and uses blockchain-based Federated Learning to train a global deep learning model [50].

In order to train deep neural networks, Yuan et al. suggest an enhanced Federated Learning framework that helps the IoT device and the associated centralized server to overlook the training computation. The communication overhead is found to be decreased by the sparsification of activations and gradients. According to empirical research, the proposed system only necessitates less synchronization traffic than plain-vanilla Federated Learning while guaranteeing a low accuracy loss [51]. Analysis of the various Federated Learning systems, highlight the implications and potentials in healthcare and also summarizes the general difficulties in using Federated Learning in the bio-medical domain [52]. Nguyen et al. present a state-of-the-art overview of the use of FL in healthcare domains, including smart health data management, remote medical monitoring, medical imaging, and COVID-19 detection. The major takeaways from the study are also emphasized, along with an analysis of several recent smart healthcare projects in Florida [53].

Microservices architecture is suggested as a new design style that is simpler to update and deploy Fog IoT applications due to its fundamental properties, such as small granularity and low coupling. Microservice deployments are significant today because of their high performance and suitability for IoT applications [54]. Compared to service-oriented and monolithic architectures, the purpose of microservice architecture is to divide the system into discrete, independent components that can be connected to share services and architectures [55]. Each microservice is responsible for a single sub-task or service, requiring fewer compute resources and lower communication overhead. The development of IoT microservices for the healthcare sector is the primary discussion of the article by Benayache et al.. Zhao et al. describes an architecture based on the microservice container Fog system to execute delay-sensitive and cost-effective mobility applications in which the costs are calculated as the sum of the computation and communication expenses [56]. A novel approach to learning the microservice applications using predictive autoscaling deployed on containerized Fog computing infrastructure proposed by Abdullah et al. seems to have less number of rejected requests and SLA violations compared to existing systems [57]. Samodha et al. investigate the factors that distinguish microservices-based application scheduling in Fog computing from other application models. In addition, the work analyzes the integration of microservices for IoT applications using application modeling, placement composition, and performance evaluation [58]. The introduction of the Internet of Healthcare Things (IoHT) platform for the healthcare environment uses the broker-less architecture for data collection, user management, device management, and remote device control [59].

### 3.6. Research gaps

Federated Learning in the Edge layer for medical anomaly detection is a promising approach to enable the development of accurate and efficient anomaly detection models while preserving the privacy and security of sensitive medical data. However, there are several research gaps that need to be addressed to fully realize the potential of FL in this domain. One of the research gaps is the incorporation of SDM modules across multiple layers of computing. To the best of our knowledge, no publications have addressed ECG anomaly detection using IoT microservice applications using SDM. This paper aims to propose a microservice-based Federated Learning model for one of the critical medical applications, ECG monitoring which has improved data privacy, increased data diversity, more efficient use of resources and real-time updates. The proposed FedSDM model predicts the ECG data anomalies by applying Federated Learning in Edge, Fog, and Cloud layers and brings out a policy of usage at the appropriate level.

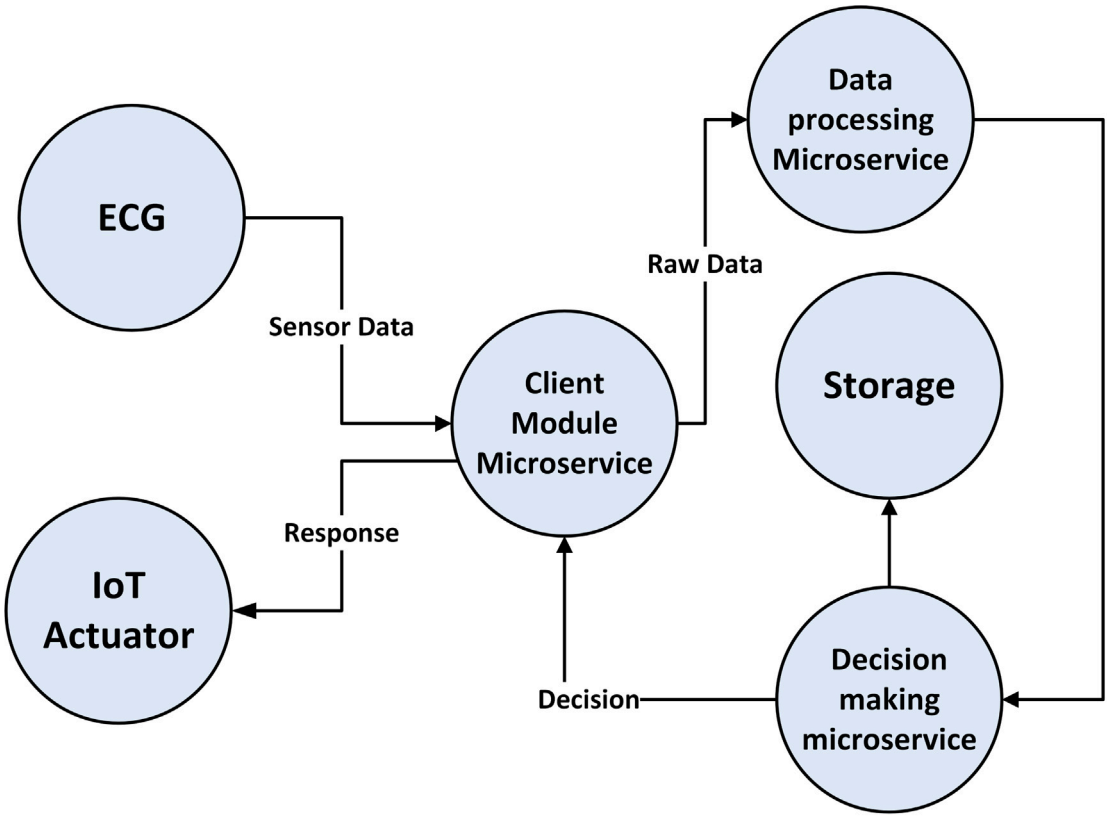



**Fig. 1.** FedSDM — Data flow diagram.

## 4. Proposed method

### *4.1. Application model*

Healthcare systems are facing enormous challenges due to the pandemic and chronic diseases. The proposed architecture is based on expectations extracted from the actual application of the healthcare sector. If successfully applied, Edge/Fog computing can reduce the latency experienced in the quality of service (QoS) and minimize the bandwidth usage in any healthcare application and later can be extended to other time-critical applications. The main difference between Fog-based and Cloud-based systems is the computing capacity and storage of the Fog devices between the patient and the Cloud data center. In most cases, the devices are not effectively used. With Fog technology, the under utilization of such intermediate devices can be addressed. The proposed Fog-based architecture includes the concept of Fog node virtual machine partitioning to run medical IoT device data efficiently. Multiple processes running on a single Edge/Fog node might cause congestion, preventing tasks from running smoothly. To solve this problem, Edge/Fog nodes can establish virtual machines responsible for delivering computational resources to the tasks. Virtual machines are primarily separate modules, each of which does a certain task.

Medical IoT applications are increasingly being implemented using modular architectures, which use microservices architecture to allow time-sensitive tasks inside Edge/Fog and latency-tolerant jobs inside the Cloud. Hence, the proposed architecture selects the microservice architecture for designing and modeling critical real-time medical applications. The following subsections discuss the application model in detail. The data flow diagram for the proposed model is depicted in Fig. 1.

#### *4.1.1. Multi-tier architecture*

A multi-tier architecture paradigm has been identified for an integrated Fog healthcare application. IoT devices such as sensors and actuators constitute tier-0. The Edge/Fog (proxy servers, gateways) and the Cloud nodes constitute the architecture's tier-1 and tier-2. In the proposed Fog-based smart healthcare system, the Fog layer act as a supportive intermediate layer for processing and analyzing real-time critical healthcare data near end-users. The Cloud, which acts as tier 3, supplies additional processing and storage resources if the Fog devices cannot handle the incoming request requirements. Virtualized processing cores, storage, and memory are considered as the resources at Fog nodes. If a request meets the resource requirements such as CPU, memory, and

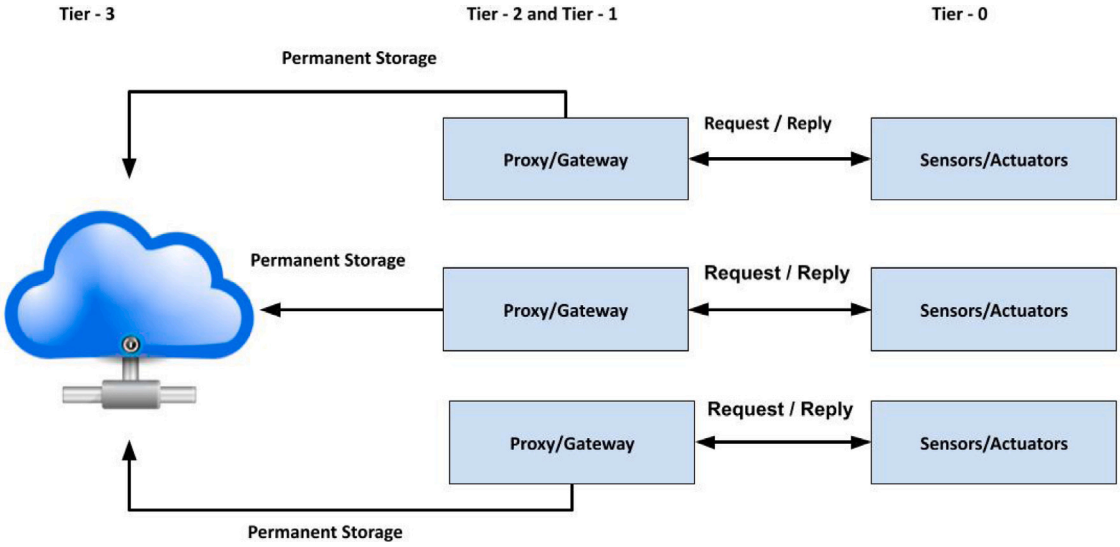



**Fig. 2.** FedSDM — Multi-tier architecture.

bandwidth, the request can be processed by the current Fog or Edge device else, the request can be transmitted to the neighboring device.

The proposed approach is simulated using iFogSim2 [12], the details of which are presented in Section 4. The workflow of the proposed work is as follows: Electrocardiogram signals are sensed by sensors connected to the patient and transmitted to the Edge/Fog nodes. All sensors have the same sensing frequency. A tuple is a data structure consisting of a set of attributes, and is used to represent a unit of information that can be transmitted between the nodes in a Fog computing environment. The tuple transmit rates of the sensors are set using transmitDistribution, an attribute in the Sensor class in iFogSim2. ECG_TRANSMISSION_TIME is set as 5 ms. Actuators are used to carry out corresponding actions based on the outcomes of applications. The Edge/Fog nodes process the data received from the sensor IoT device, and the patient's health status is relayed to the patient's smartphone. The Edge/Fog nodes are located near the network's Edge, closer to the IoT device, so that the patients receive immediate action in real-time. The data center at the Cloud layer is the upper layer of the proposed system, which stores the permanent healthcare data of the patient. Data stored in the Cloud can be retrieved anytime by the users connected with the application. The proposed health monitoring system's multi-tier architecture is depicted in Fig. 2.

#### *4.1.2. Mobility*

The mobility of Fog nodes or users raises an issue for the Fog computing platform by maintaining resources close to users at all times [60]. In addition, IoT device mobility can impact the performance of Fog applications because of the rapid movement of the devices from one access point to another. The mobility component of iFogSim2 allows distributed node movements and also enable to customize the movement direction and speed of the end devices. This helps in the better simulation of the system which is at par in real-time.

#### *4.1.3. Clustering*

Resource augmentation can considerably assist resource-limited Fog resources, particularly computational and storage capacity, used for resource-critical applications. As a result, a clustering technique that allows resource augmentation from among Fog resources is needed. The clustering component of iFogSim2 allows distributed dynamic coordination and collaboration among multiple nodes. Each node can query and register its cluster members according to the specific clustering policies.

#### *4.1.4. Microservices*

Application development has shifted from monolithic design to microservice architecture to fully use the Fog computing paradigm. The proposed system's application model comprises a collection of microservices. Microservices architecture use for building an application as a collection of small services that each runs in their process and communicates using lightweight protocols. Microservices allow one to build a system out of a collection of small, isolated units that can manage their respective data. Advantages include heterogeneity, scalability, resilience, organizational alignment, ease of deployment, and composability, which will open up the possibility of large-scale IoT application development. Due to its excellent performance and applicability for IoT applications, microservice deployments are more relevant nowadays. The proposed system's application model is made up of a collection of microservices which is depicted in Fig. 1. The vertices indicate each microservice, while the edges show the data relationships between them. There are three microservices in this design, which are described below.

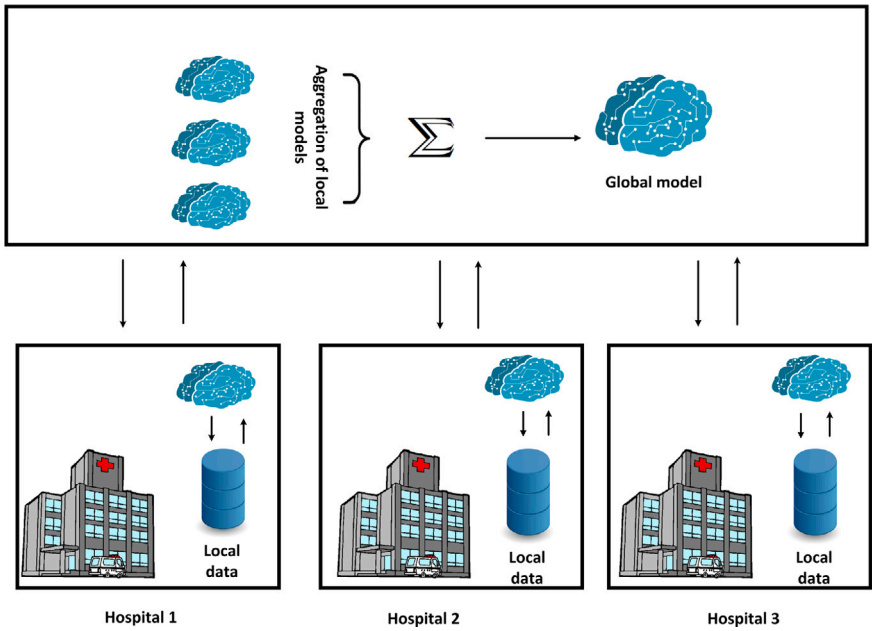



**Fig. 3.** Structure of FedAvg for FedSDM.

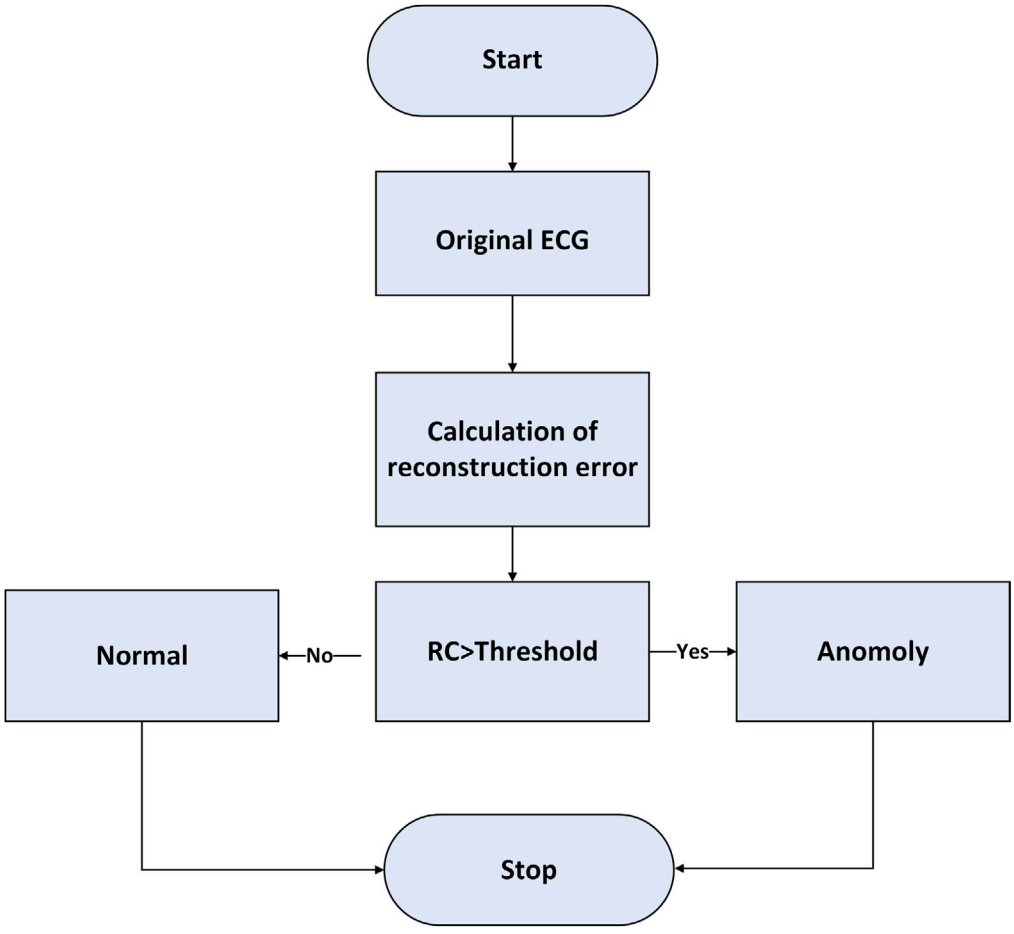



**Fig. 4.** Flowchart — Autoencoder in FedSDM.

- Client microservice: It is the Fog computing-based healthcare system's front end. The sensor ECG signals attached to each smartphone are received by the client microservice, deployed on the users' cellphones, and transmitted to the preprocessing microservice on the Fog device.
- Preprocessing microservice: Before transferring data for further processing, the preprocessing microservice performs data validation and cleaning to remove the noise of ECG sensor data.
- Decision-making microservice: The microservice is in charge of real-time decision-making whether the ECG is normal or anomalous. Based on the decision, a warning signal regarding the patient's health is sent to the client microservice.

These microservices communicate with one another to keep track of users' health. Based on the placement policy, preprocessing and decision-making constitutes a time-critical service that can be deployed on Edge or Fog. The data for permanent storage is being moved to the storage module. An effective resource provisioning methodology uses a multi-level hierarchical Fog architecture in which application placement requests are handled at the Fog node at different levels. It models an IoT-critical medical application as groupings of containerized microservices and uses a decentralized placement strategy to distribute them within the Fog environment.

### 4.2. Federated learning model

Federated Learning is a distributed privacy-preserving machine learning paradigm in which a central server connects with various end devices, including smartphones, laptops, and security cameras, with limited computation and storage availability. Hence the clients avoid sharing the data with the server. Clients receive the server's most recent global model for each communication round, and a small percentage of clients use stochastic gradient descent (SGD) to update it throughout several rounds. The new global model is then obtained by aggregating these updated parameters on the central server. Most of the server's Cloud deployments need enormous storage and computing capacity. In the proposed system, the Edge/Fog devices use a methodology named FedAvg to launch the Federated Learning module. A communication-efficient approach for distributed training with many clients is federated averaging (FedAvg). Compared to traditional training and learning, FedAvg considerably lowers the communication cost between servers and clients by involving many local SGD updates and one aggregate by the server in each communication cycle [61]. The FedAvg module used in this work is depicted in Fig. 3.

To initiate the process, an overall model is downloaded from the central server and is trained with local data over several epochs. The outcomes are the local updates. These local model updates collected from the end devices are aggregated by the FegAvg algorithm to generate the global model, which is continued until the required performance is achieved. The proposed method considers three scenarios where the FedAvg is deployed in different layers: Edge, Fog, and Cloud.

The proposed approach also uses autoencoders, a particular type of neural network used for training and testing the data to detect anomalies in the ECG (electrocardiogram) readings. An autoencoder consists of three components: encoder, code, and decoder. The encoder compresses the input and produces the code, which is later used by the decoder to reconstruct the input. An encoding technique, a decoding technique, and a loss function to compare the output with the objective are required when building an autoencoder. Autoencoders can only compress data meaningfully similar to what they have been trained on. Although the autoencoder's output will not be a perfect replica of its input, it will be a similar degraded representation. The encoder and decoder are both fully linked feedforward neural networks. Fig. 4 depicts the flowchart of the proposed system's autoencoder.

The architecture of the autoencoder is presented in Fig. 5(a). The proposed approach has two layers in both the encoder and decoder, without accounting for the input and output, as shown in Fig. 5(b). The number of nodes per layer reduces with each subsequent encoder layer and grows back in the decoder. In terms of layer structure, the decoder and encoder are also symmetrical. The loss function is the mean squared error in the proposed system configuration.

The proposed architecture implements an encoder and a decoder using an ANN architecture. The ECG data is fed as input to the model, and the model tries to reconstruct it. The error between the original data and the reconstructed output will be called the reconstruction error. Based on this reconstruction error, the ECG data is classified as anomalous. In order to do this, the model is first trained on the standard ECG data and is tested on the complete test set. The autoencoder reconstructs the abnormal ECG when the input is provided. However, since it has been trained only on the standard ECG data, the output will have a more significant reconstruction error. The input is classified as anomalous if the reconstruction error exceeds the threshold. The proposed system uses the Keras Subclassing API to build the model, as it provides reasonable control over the model compared to Sequential API. Autoencoders are unsupervised learning models, but the proposed method trains them using the supervised method, so it is more like they are used as self-supervised.

Large-scale FL experiments can be deployed and carried out using FL frameworks. The flower is a comprehensive FL framework that offers new tools to conduct large-scale FL experiments and consider highly heterogeneous FL device environments. It can run FL experiments with clients up to 15M in size. It uses only a pair of high-end GPUs. We have selected the flower framework to implement the Federated Learning module of the proposed system. The architecture of the flower framework for the proposed system is depicted in Fig. 6(a). As can be seen in Fig. 6(b), the autoencoder is added to the flower framework.

### 4.3. Proposed method

The proposed method works as follows: ECG sensor values collected from the patient are stored in the Edge device (Ex: mobile phones). The client microservice and the data preprocessing microservice, which resides on the Edge, do the required computation. The preprocessed ECG values are fed to the Smart Decision Making module, which checks for the anomaly. If any

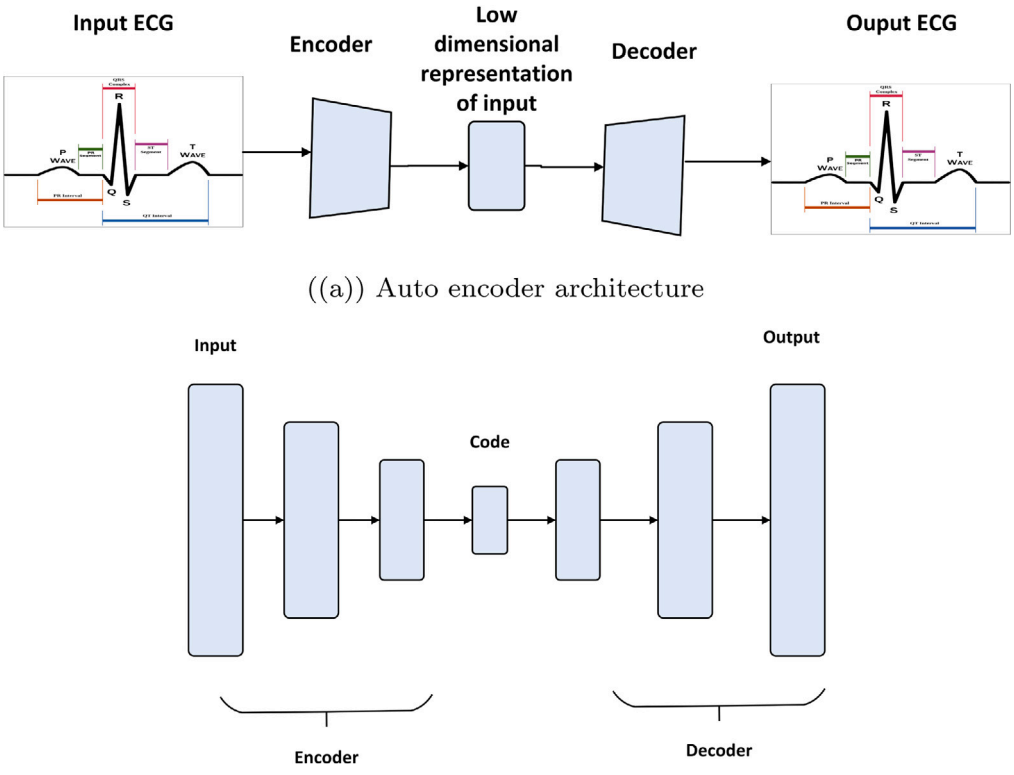



**Fig. 5.** Auto encoder and ANN.

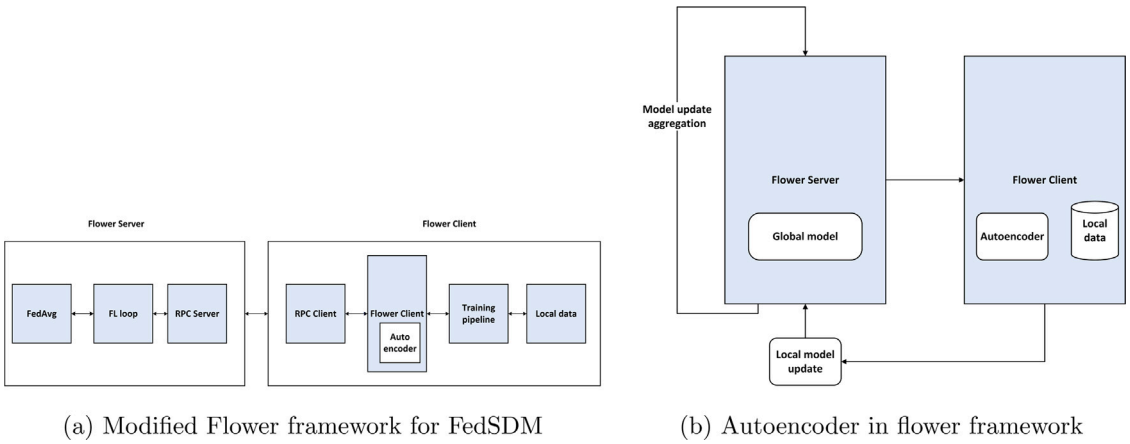



**Fig. 6.** Auto encoder integrated Flower framework.

anomaly exists, the notification is sent to the end device. The proposed architecture compares the results of placing the FL-based decision-making module at different layers, as mentioned in the previous section. In all the placement policies, the local updates from the corresponding devices/nodes are aggregated in the respective layer (Edge/Fog/Cloud). The diagrammatic representation of the different deployment policies is presented in Fig. 7. Each deployment has been compared for its performance in the learning efficiency and the QoS parameters, which are presented in the next section.

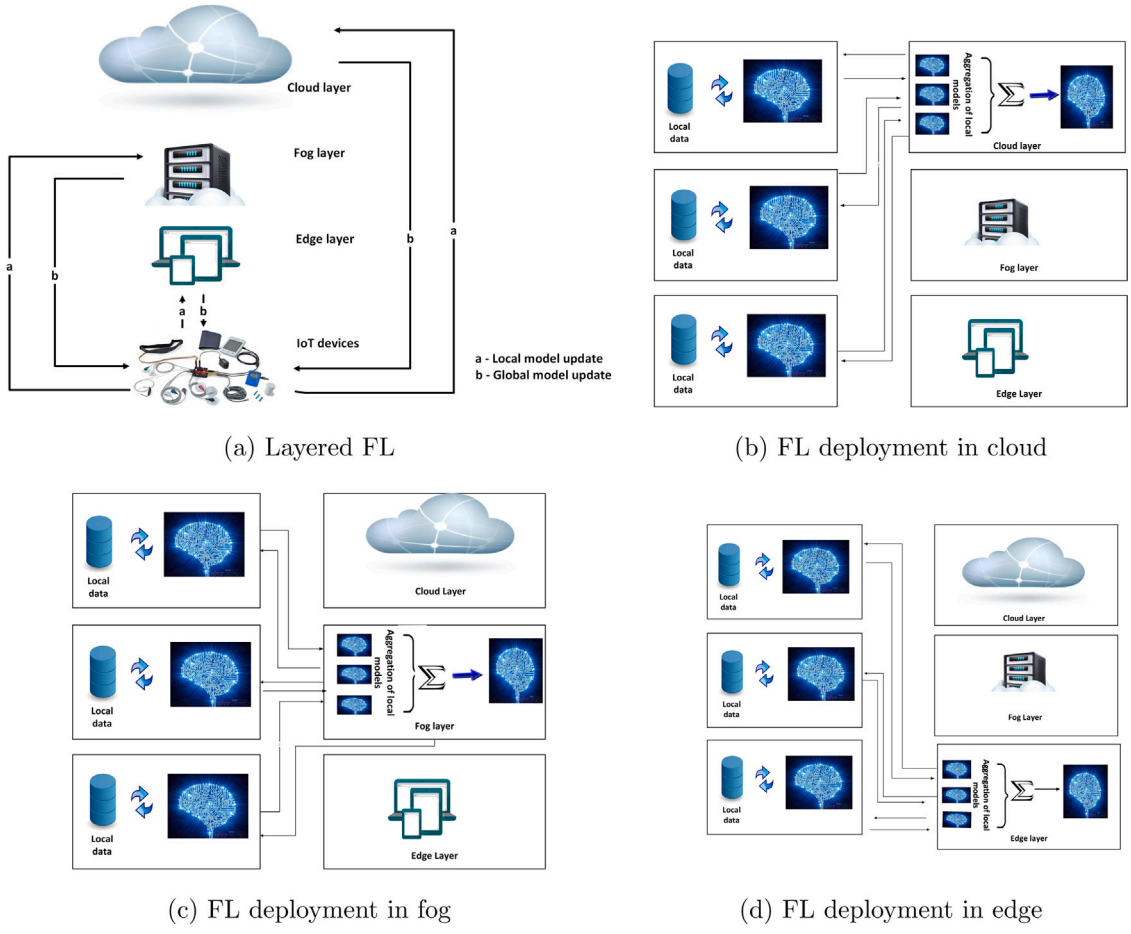



**Fig. 7.** FL deployment in different computing layers.

### *4.4. Evaluation metrics*

This section explains the performance evaluation measures used to evaluate the FedSDM. Key Performance Indicators (KPIs) that compare Edge/Fog/Cloud computing in various usecase implementations can help organizations to evaluate the performance and effectiveness of each approach in different scenarios. Here are some KPIs that can be used for comparing Edge/Fog/Cloud computing:

Latency: Measuring the round-trip time between a device and the Edge/Fog/Cloud node can help to identify any latency issues and evaluate which approach is better suited for the specific use case.

Bandwidth usage: Measuring the amount of data transferred between the device and the Edge/Fog/Cloud node can help to evaluate which approach is more effective in reducing bandwidth usage.

Availability: Edge/Fog/Cloud computing rely on distributed nodes to process data. Measuring the availability of these nodes can help to evaluate the best approach which is more reliable in terms of availability.

Processing time/Execution Time: Measuring the time taken to process data on the Edge/Fog/Cloud node can help to evaluate the approach that is more effective in handling the workload.

Energy consumption: Measuring the energy consumed by the Edge/Fog/Cloud implementations can help to identify the most energy-efficient node.

Security: Measuring the security solutions can help to evaluate more secure implementation in handling sensitive data.

By comparing these KPIs, organizations can determine the approach that best fits their requirement and ensure that they are making the most effective use of their computing resources [62–64]. We have chosen latency, energy consumption, network use, cost, and execution time as evaluation criteria in our proposed method, as they are crucial for evaluating critical medical applications. The subsequent section outlines the evaluation parameters for our proposed approach.

**Table 2**
Notations used in evaluation metrics equations.

| Symbol | Meaning | Symbol | Meaning |
|---|---|---|---|
| $T$ | Tasks | $E_0$ | Power required for the server in an idle state |
| $N$ | Nodes | $NU$ | Network use |
| $m$ | The Number of servers | $l$ | Latency experienced by the network |
| $n$ | Number of VMs inside the host | $TNS$ | Tuple network size |
| $EXT(N_i)$ | The execution time required by node $N_i$ | $cost(T_k^i)$ | Cost for processing task $T_k^i$ |
| $TXT$ | Total execution time | $M(T_k^i)$ | Memory needed by task $T_k^i$ |
| $TL$ | Total latency | $Bw(T_k^i)$ | Bandwidth needed by task $T_k^i$ |
| $CAL$ | Current average latency | $NU$ | Network use |
| $CC$ | Simulator clock | | |
| $ET$ | Execution time of the tuple | | |
| $E$ | Total energy consumption | | |
| $EN_{ik}$ | Energy consumption by the task $T_k$ | | |

#### 4.4.1. Evaluation parameters for proposed approach

A set of n independent tasks are delivered to the system at each time, assuming that $T_k$ represents the $k$th task denoted as follows:

$$T = \{T_1, T_2, T_3, \dots, T_n\} \tag{1}$$

The assumed infrastructure comprises Edge/Fog/Cloud nodes, which are processors with characteristics such as CPU rate, CPU usage cost, bandwidth usage cost, and memory usage cost. The set of m processors is made up of fog nodes as mentioned below:

$$N = \{N_1, N_2, N_3, \dots, N_m\} \tag{2}$$

where $N_i$ represent the $i$th processing node. The processor $N_i$ allocated with job $T_k$ is denoted by $T_k^i$.

A set of one or more tasks may be assigned to one processor for computing:

$$N_i Tasks = \{T_x^i, T_y^i, \dots, T_z^i\} \tag{3}$$

The subsequent information discuss the performance metrics employed to assess the implementation of FedSDM across the Edge, Fog and Cloud layers.

*Execution time.* The execution time (EXT) required by node $N_i$ to finish a set of $N_i Tasks$ assigned to it is:

$$EXT(N_i) = \sum_{T_k^i \in N_i Tasks} EXT(T_k^i) \tag{4}$$

$$EXT(T_k^i) = \frac{length(T_k^i)}{CPU_i} \tag{5}$$

where length($T_k^i$) denote the number of instructions in the task $T_k$. The node $N_i$'s CPU rate is represented by $CPU_i$ and depends on factors such as clock rate, core count, instruction level parallelism, etc. Total execution time is the total time taken by the system to complete all the tasks, defined from the time when the request is received until the last task, or the time when the last machine completes. Total execution time is determined by the formula:

$$TXT = \sum_m [EXT(N_i)] \tag{6}$$

The time used to complete the job while utilizing system services is included in the task's execution time. Execution times differ amongst tasks because they rely on how intensive the processing and input–output activities are.

*Latency.* The application's control loop is Client Microservice → Preprocessing Microservice → Decision Making Microservice → Client Microservice which is described in Section 4.1.4. The control loop in our proposed approach has a relatively shorter latency, indicating better coordination and placement of computational resources. The iFogSim2 simulator provide multiple methods for module placement, including the one called edge-ward placement which has been utilized in this work. This method involves shifting microservices towards the top of the fog hierarchy, so that it leads to a deployment of a singular instance of each microservice along the route from the edge to the cloud. Due to the restricted resource capacity of fog nodes, this strategy places microservice instances in higher fog layers, increasing average latency.

$$TL = \sum_m CAL \tag{7}$$

Total latency ($TL$) represented in Eq. (7) directly depends on the allocation of VMs in fog devices in which the tasks are distributed for execution. Total latency is the summation of the current average delay (CAL) experienced by every VM inside the host for $m$ servers, where CAL is calculated as follows.

$$CAL = CC - ET \tag{8}$$

Here, *CC* denote the simulator clock and *ET* is the execution time of the tuple. *CC*, the simulator time is recorded by the simulator when the response is received by the IoT device and *ET*, when the request is received by the fog node. The difference between these times give the total latency experienced by the tuple, which include all the delays in the communication path, such as transmission delay, propagation delay, processing delay, and queuing delay.

*Energy consumption.* Energy consumption is one of the significant parameters that we have in computing systems. This typically includes infrastructure for data communications for backup power supplies, environmental controls, including cooling systems, fire control, and different security technologies. Infrastructure operational costs are impacted by the power supply. Hence methods must be put in place to lower these costs. The edge-ward method places the majority of microservices on cloud VMs, which increases the energy consumption of cloud resources. The amount of energy used depends on how many microservices are active on each tier. Efficiency, affordability, availability, dependability, and environmental protection of devices are all greatly impacted by energy consumption reduction.

We model the server or host's energy consumption as the sum of two components: the fixed energy for the server in an idle state and the variable energy for server utilization while processing the requests. Energy consumption depends on the server's number of VMs and the allocated MIPS allocated for each VM.

The variable energy for server utilization while processing the requests is $EN_{i_k}$. E is the total energy consumption which can be calculated by

$$\sum_{i=0}^{m}(\sum_{k=1}^{n} EN_{i_k} + E_0) \tag{9}$$

$$EN_{i_k} = e_1 * EXT(T_k^i) \tag{10}$$

where $EN_{i_k}$ is the energy consumption by the task $T_k$ running on the virtual machine or node i. Operating the data center requires $E_0$, the fixed energy of the server in idle state, and $e_1$, the energy consumption per unit time in node $N_i$. The suggested method makes some fixed assumptions regarding the simulation setup, including the distance between fog nodes, energy efficiency, and power consumption of communication devices. However, we are aware that these factors typically affect the amount of energy used for communication, therefore we will address this in the future to improve our model.

*Network usage.* A crucial metric for comparing various approaches is the overall volume of data delivered across a network. Particularly on large networks, high data transfer may result in network congestion, service interruptions, or an increase in the control loop's average delay for the applications. In comparison to cloud operations, the latency can be significantly reduced if the edge of the network can manage the portion of the workload. Additionally, the Edge-to-Cloud traffic is to be maintained. Data transmission size could be considerably decreased by data pre-processing at the edge and fog devices. However, bandwidth conservation is essential because many endpoints connect to the network and many database servers are needed to run them. Network usage depends on the latency experienced by the network and the tuple size of the data for $'n'$ VMs in the host as listed in Eq. (11).

$$NU = \sum_{n}(l * TNS) \tag{11}$$

where $l$ denote the latency experienced by the network and $TNS$ denote the tuple network size. Tuple network size refers to the number of tuples that can be processed simultaneously within the network. Total network use depends on the number of VMs in fog devices in which the tasks are distributed for execution.

*Cost.* Costs include network hardware, infrastructure, network communications, processing, and storage costs. The investment made by service providers in Fog computing also include the placement of processing and communication workloads in the fog device. One of the main issues with Fog computing is cost saving.

Processing cost is defined as:

$$cost(T_k^i) = c_1 * EXT(T_k^i) + c_2 * M(T_k^i) + c_3 * Bw(T_k^i) \tag{12}$$

where $c_1$ denote the CPU usage fee per time unit in node $N_i$, and EXT($T_k^i$) is given in Eq. (5). $c_2$ denote the memory usage fee per data unit in node $N_i$ and M($T_k^i$) represent the memory needed by task $T_k$. Task $T_k$ processed in node $N_i$ needs an amount of bandwidth Bw($T_k^i$), which is the sum of input and output file size. $c_3$ is the bandwidth usage fee per data unit. The following formula is used to determine the cost of each task in the Edge–Fog–Cloud system in total.

$$Totalcost = \sum_{T_k^i \in N_i Tasks} cost(T_k^i) \tag{13}$$

Our proposed strategy aims to deploy FedSDM in Edge/Fog/Cloud layers and evaluate the above-mentioned parameters. Table 2 contains the list of acronyms used in this section.

**Table 3**
Configuration parameters [12].

| Parameter | Cloud | Fog | Smartphone |
|---|---|---|---|
| CPU length(MIPS) | 44 800 | 2800 | 2800 |
| RAM (MB) | 40 000 | 4000 | 4000 |
| Uplink BW (MB) | 100 | 10 000 | 10 000 |
| Downlink BW (MB) | 10 000 | 10 000 | 10 000 |
| Busy power (J) | 16 ∗ 103 | 107.339 | 87.53 |
| Idle power (J) | 16 ∗ 83.25 | 83.433 | 82.44 |

## 5. Experimental setup

### 5.1. iFogSim2

The market is very competitive with simulators for simulating Cloud, Fog, and Edge devices. Few examples are FogNetSim+, Edge–Fog, Yet Another Fog Simulator (YAFS), EdgeNetworkCloudSim, PureEdgeSim. For modeling and simulating Edge/Fog/Cloud computing infrastructures and services, we have chosen iFogSim2, an extension of Cloudsim, since this framework can be used to develop and deploy experiments for Edge/Fog/Cloud devices that handle compute, memory, I/O, and VM allocation, as well as VM power models, among other things. The iFogSim2 simulator, an extension of the iFogSim simulator, holds the properties of service migration, distributed cluster building across Fog nodes, and microservice orchestration. This simulator helps validate the proposed approach's performance in the Fog computing environment. The components of the iFogSim2, such as mobility, clustering, and microservices, are loosely coupled and can be utilized for simulation in such scenarios. iFogSim2 incorporates real datasets for assessing the performance of different service management strategies in Fog computing settings, unlike most existing solutions. It includes node clustering, mobility management, and microservice orchestration methodologies that can be used as benchmarks for comparing performance [12].

All iFogSim core classes, such as FogDevice, AppModule, Sensor, and Actuator, have object references in the Controller class, and it can access the Tuple class through an Application object. The Clustering element of iFogSim2 allows distributed dynamic coordination and collaboration between multi-fog devices. To add the mobility component to the iFogSim2 simulation, it includes classes such as DataParser and MobilityController. The functions of these classes are described below:

- The data parser class separates and assimilates location data from many IoT end devices so that application services may be handled based on their unique mobility patterns.
- MobilityController class helps to dynamically start the required sequential and parallel actions on separate FogDevice and AppModule referenced objects for mobility management.

During simulation, the proposed model assumes two mobility patterns: 'RANDOM MOBILITY' and 'DIRECTIONAL MOBILITY'. In the 'DIRECTIONAL MOBILITY' model, the time period between two of these motions is equal to ensure that the user/IoT device maintains a constant speed. The 'DIRECTIONAL MOBILITY' being used has many consecutive coordinates lying at the exact distances across the Melbourne Central Business District (CBD) for a user or IoT device. These coordinates help construct the events to simulate the movement of the associated end IoT device. Numerous random mobility patterns are available in the simulator to represent users' RANDOM MOBILITY model behaviors. It includes the user's direction, speed, stopping time at each location, and duration within each Edge/Fog node's communication range. These patterns help simulate the user's real-time behavior and enable a better evaluation of the proposed system.

### 5.2. Simulation environment

This section explains the simulation environment used to evaluate the proposed approach. The sensors detect the ECG of the patient and send the data to the Fog nodes regularly. Data is processed and analyzed on the Fog nodes to determine whether the patient's health status is normal or critical. The results are subsequently sent to the Cloud for storage and the patient's smartphone. The Fog nodes' connection to the Cloud server is established through the proxy server. The client module is integrated in IoT devices to get sensor data. The processing module is embedded in Fog nodes to process and analyze the incoming data to assess the patient's health status. The Fog node then communicates the results to the associated IoT device, which displays them. It must define values for numerous parameters in iFogSim2 when generating Fog devices, such as CPU length, RAM, Bandwidth, and so on. The settings used for device configuration in iFogSim2 [65] are listed in Table 3.

Fog devices are the computational devices in iFogSim2. Computational gadgets, on the other hand, come in various levels. The parent node acts as a Cloud server and is placed on Level 3. The Fog nodes are connected to the Cloud server via a proxy server at Level 2. Fog nodes are closer to the user at Level 1, which is considered as Edge device, giving more frequent computational and storage capacities. Sensors and actuators are used in Level 0 IoT devices. The MicroserviceFogDevice, Sensor, and Actuator classes of iFogSim2 simulate the physical topology. The scenarios were simulated on an Intel Core i7 CPU running at 1.80 GHz and 4 GB of RAM. The fractional selectivity of the input–output relationship inside a module is set to 1.0.

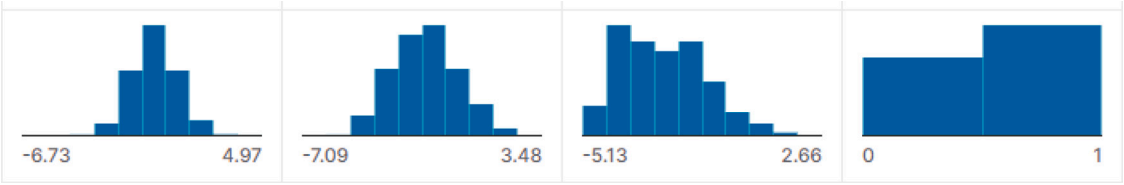



**Fig. 8.** Dataset description.

(a) User mobility (b) Resource location

**Fig. 9.** Data set for mobility.

## 6. Results and discussion

### 6.1. Dataset

#### 6.1.1. Dataset for federated learning

The dataset [66] considered for this work has 140 columns representing the ECG readings and a label encoded as 0 or 1, denoting whether the ECG is abnormal or normal. Columns 0–139 contain the ECG data point for a particular patient. These are floating point numbers. The first three column value ranges along with the final label description, are presented graphically in Fig. 8 to understand the spread of values in the dataset. The first column of Fig. 8 displays the maximum and minimum values for the first data point of the patient set in the dataset, while columns 2 and 3 represent the same information for the second and third data points respectively. The final illustration in the figure provide a label indicating whether the ECG is classified as normal or abnormal. The dataset has 58% of the tuples belonging to the normal class and the remaining belonging to the abnormal class.

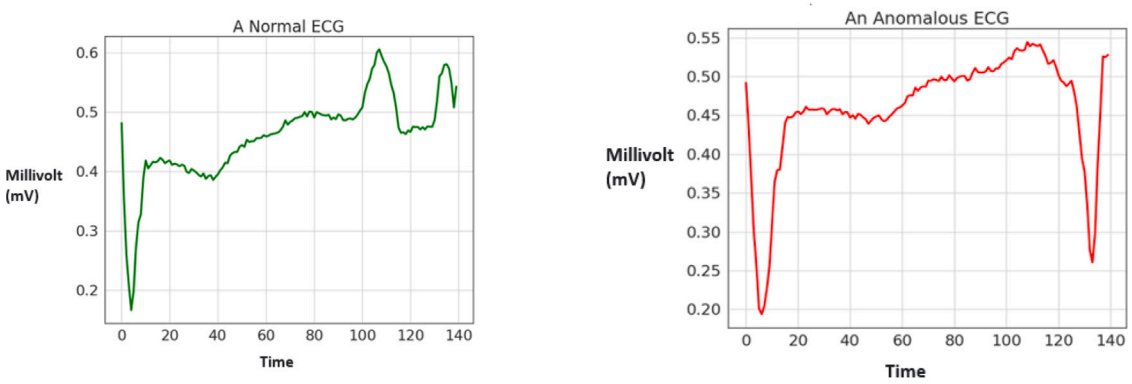
#### 6.1.2. Dataset for mobility

EUA dataset [67] provides the position information of many Fog nodes distributed across CBD zones of major Australian cities, namely Melbourne and Sydney. The dataset is segmented into various regions and is divided into several blocks, among which a random node is picked as the proxy server to ensure granularity. Within a block, all nodes except the proxy server serve as the IoT devices' gateway. This repository includes a collection of EUA datasets gathered from real-world data sources. The datasets have been made available to the public in assisting Edge computing research. The data corresponds to the Australian region and helps better simulate a natural time environment. The user mobility pattern and the resource location of the dataset are shown in Fig. 9.

### 6.2. Analysis and observations

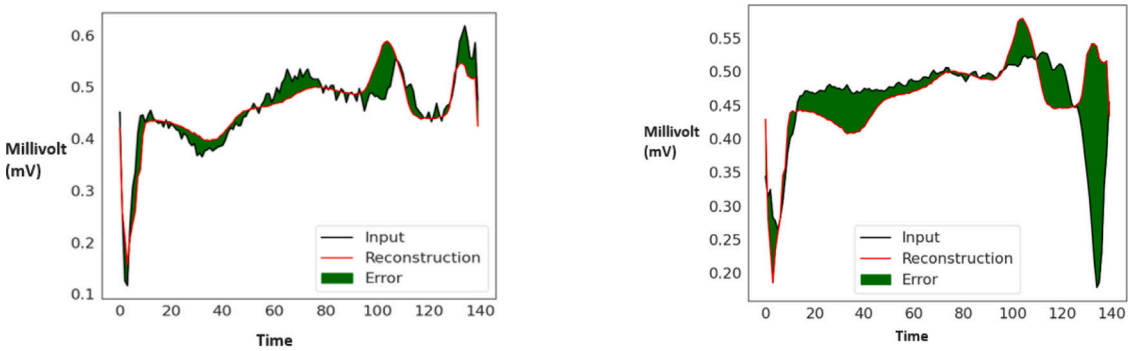
This section presents the results and the observation. The model is evaluated as described in the previous sections for varying placement policies. Fig. 10 presents the normal and abnormal ECG samples. Fig. 11 shows the reconstructed normal and abnormal ECG plots. The reconstructed ECG helps in predicting whether the ECG is anomalous. The reconstructed one with the error beyond a threshold is considered anomalous. The error calculated from these figures helps in this classification. Fig. 12 highlights the training and the testing loss graphically.

Fig. 13 compares the identified performance parameters for different placement policies discussed in this work. It could be observed that the deployment of the FL module in the Edge layer reduces the Cloud energy consumption by 2% with a decrease in network use of 32%. This, in turn, reduces the cost by 50%, the execution time by 32%, and the latency by 86%. All the above comparisons are against the placement of the FL module in the Cloud layer. While analyzing the results of placement of the FL module in the Fog layer against the Cloud layer, Cloud energy consumption decreases by 2%, network use by 31%, cost by 41%, execution



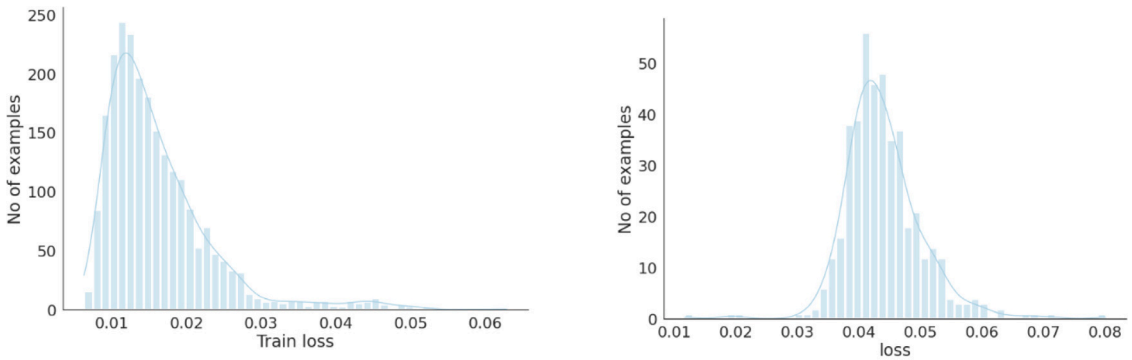

(a) Normal ECG (b) Anomalous ECG

**Fig. 10.** Normal ECG and anomalous ECG plot.




(a) Reconstructed Normal ECG (b) Reconstructed anomalous ECG

**Fig. 11.** Reconstructed normal and anomalous ECG plot.




(a) Train loss (b) Test loss

**Fig. 12.** Train and test loss graphs.

time by 23%, and latency by 85%. Table 4 presents the above discussions in a consolidated manner for better understanding of the results. However, the router energy consumption is found to be more (i.e.) 2.3% and 2.4% for FL module deployment in the Edge, and Fog layers since more computations are performed in those layers. A similar comparison of placing the FL module in Edge and

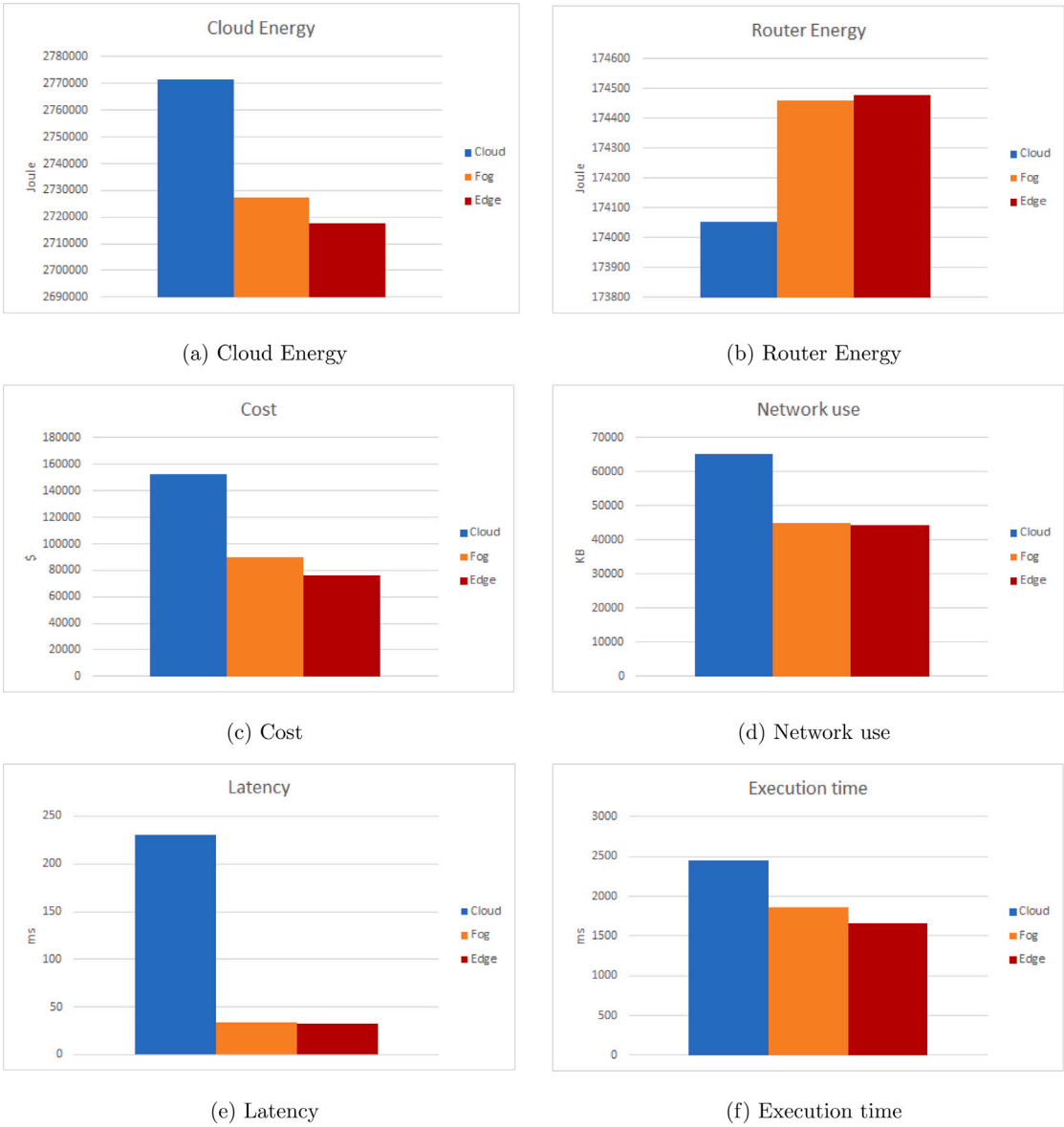



**Fig. 13.** Result of FL deployment in Edge/Fog/Cloud layers.

**Table 4**
Comparison of Edge and Fog FL placement against cloud.

| Metric | Edge FL placement | Fog FL placement |
|---|---|---|
| Energy consumption (J) | 2% | 2% |
| Network use (KB) | 32% | 31% |
| Cost ($) | 50% | 41% |
| Latency (ms) | 86% | 85% |
| Execution rime (ms) | 32% | 23% |

Fog yields a performance increase of 0.3%, 2%, 15%, 11%, and 3% for energy consumption, network usage, cost, execution time, and latency, respectively as presented in Table 5. Table 6 shows the number of simulations conducted and the average results for each of the parameters. In conclusion, FL module deployment in the Edge layer is superior to FL module deployment in Fog or Cloud, which adds to the fact that the integration of AI on Edge enables smart healthcare systems. This could also support real-time or advanced remote patient monitoring by immediately processing the clinical tests.

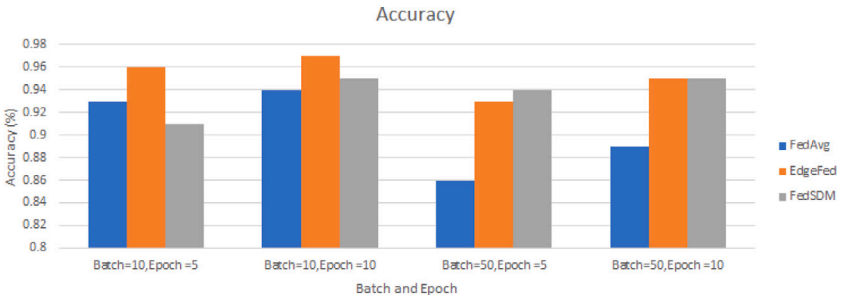



**Fig. 14.** Accuracy comparison of FedSDM with FedAvg and Edgefed.

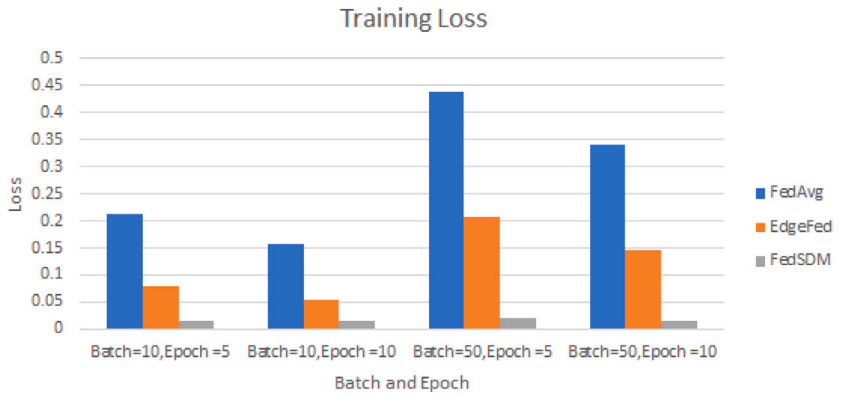



**Fig. 15.** Training loss comparison of FedSDM with FedAvg and Edgefed.

**Table 5**
Comparison of Edge FL placement against Fog.

| Metric | Edge FL placement |
|---|---|
| Energy consumption (J) | 0.3% |
| Network use (KB) | 2% |
| Cost ($) | 15% |
| Latency (ms) | 11% |
| Execution time (ms) | 3% |

**Table 6**
Number of simulations and the mean parameter values for the implementation of Edge FL compared to Fog FL implementation.

| Metric | Number of simulations | | | | | Average |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Energy consumption (J) | .34% | .37% | .30% | .25% | .36% | .3% |
| Network use (KB) | 1.6% | 1.2% | 1.3% | 1.5% | 2% | 1.5% |
| Cost ($) | 14.8% | 11.3% | 12.7% | 13.7% | 12% | 12.9% |
| Latency (ms) | 2.9% | 1.5% | 3.4% | 3.3% | 1.7% | 2.5% |
| Execution time (ms) | 10.8% | 10.1% | 11.3% | 11% | 9.6% | 10.5% |

### *6.2.1. Comparative analysis*

In order to have an effective conclusion, the proposed approach has been compared for its accuracy and the training loss parameters against the existing results presented in the literature [24]. Figs. 14 and 15 show the contrast of the parameters used for various batch sizes and epochs. These results also prove the conclusion statement in the previous sub-section.

## 7. Conclusion

Due to the heterogeneous and dynamic nature of critical medical IoT applications in Fog scenarios, the privacy of patients become a crucial problem. This paper investigates the Federated Learning-based Smart Decision Making module for ECG Data in microservice-based IoT medical applications. In addition, we also examine the performance of the proposed system with three different placement policies considering the deployment at Edge, Fog and Cloud layers. Future work will include addressing this

work's limitations and experimenting with the model's energy usage. Also, we want to put the suggested method into action. To boost prediction models, we will also look into, improve, and deploy more aggregation techniques. In order to increase system security in real-time Edge/Fog/Cloud scenarios, we want to leverage blockchain techniques.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request

## References


[1] S. Wadhwani, 36 billion data records exposed (so far) in 2020: Risk based security, 2020, URL https://www.spiceworks.com/it-security/data-security/news/36-billion-data-records-exposed-so-far-in-2020-risk-based-security/.

[2] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.

[3] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. (2021) 1, http://dx.doi.org/10.1109/COMST.2021.3075439.

[4] J. Santos, T. Wauters, B. Volckaert, F. De Turck, Towards end-to-end resource provisioning in Fog Computing over low Power Wide Area Networks, J. Netw. Comput. Appl. 175 (2021) 102915.

[5] V.K. Quy, N.V. Hau, D.V. Anh, L.A. Ngoc, Smart healthcare IoT applications based on fog computing: architecture, applications and challenges, Complex Intell. Syst. (2021) 1–11.

[6] S. Tuli, N. Basumatary, S.S. Gill, M. Kahani, R.C. Arya, G.S. Wander, R. Buyya, HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments, Future Gener. Comput. Syst. 104 (2020) 187–200.

[7] I. Martinez, A.S. Hafid, A. Jarray, Design, resource management, and evaluation of fog computing systems: A survey, IEEE Internet Things J. 8 (4) (2020) 2494–2516.

[8] M. Laroui, B. Nour, H. Moungla, M.A. Cherif, H. Afifi, M. Guizani, Edge and fog computing for IoT: A survey on current research activities & future directions, Comput. Commun. 180 (2021) 210–231.

[9] A. Shakarami, H. Shakarami, M. Ghobaei-Arani, E. Nikougoftar, M. Faraji-Mehmandar, Resource provisioning in edge/fog computing: A comprehensive and systematic review, J. Syst. Archit. (2021) 102362.

[10] C. Puliafito, D.M. Gonçalves, M.M. Lopes, L.L. Martins, E. Madeira, E. Mingozzi, O. Rana, L.F. Bittencourt, MobFogSim: Simulation of mobility and migration for fog computing, Simul. Model. Pract. Theory 101 (2020) 102062.

[11] B. El Khalyly, A. Belangour, M. Banane, A. Erraissi, A comparative study of microservices-based IoT platforms, Int. J. Adv. Comput. Sci. Appl. (IJACSA) 11 (7) (2020) 389–398.

[12] R. Mahmud, S. Pallewatta, M. Goudarzi, R. Buyya, IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments, J. Syst. Softw. (JSS) (ISSN: 0164-1212) (2022) Elsevier Press, Amsterdam, the Netherlands.

[13] D.G. Nair, C. Aswartha Narayana, K. Jaideep Reddy, J.J. Nair, Exploring SVM for federated machine learning applications, in: Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2022, Springer, 2022, pp. 295–305.

[14] V.P. Pillai, R.K. Megalingam, System partitioning with virtualization for federated and distributed machine learning on critical IoT edge systems, in: Congress on Intelligent Systems: Proceedings of CIS 2021, Volume 2, Springer, 2022, pp. 443–453.

[15] H.G. Abreha, M. Hayajneh, M.A. Serhani, Federated learning in edge computing: A systematic survey, Sensors 22 (2) (2022) http://dx.doi.org/10.3390/s22020450, URL https://www.mdpi.com/1424-8220/22/2/450.

[16] D. Jose, J. Swaminathan, A model in healthcare cloud for securing the data using fog computing, in: ICT Infrastructure and Computing: Proceedings of ICT4SD 2022, Springer, 2022, pp. 441–448.

[17] H. Pydi, G.N. Iyer, Analytical review and study on load balancing in edge computing platform, in: 2020 Fourth International Conference on Computing Methodologies and Communication, ICCMC, IEEE, 2020, pp. 180–187.

[18] M. Prabhu, A. Hanumanthaiah, Edge computing-enabled healthcare framework to provide telehealth services, in: 2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), IEEE, 2022, pp. 349–353.

[19] P.D. Bharathi, V.A. Narayanan, P.B. Sivakumar, Fog computing enabled air quality monitoring and prediction leveraging deep learning in IoT, J. Intell. Fuzzy Systems (Preprint) (2022) 1–22.

[20] P. Yu, Y. Liu, Federated object detection: Optimizing object detection model with federated learning, in: Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, 2019, pp. 1–6.

[21] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, Y. Khazaeni, Federated learning with matched averaging, 2020, arXiv preprint arXiv:2002.06440.

[22] M.G. Arivazhagan, V. Aggarwal, A.K. Singh, S. Choudhary, Federated learning with personalization layers, 2019, arXiv preprint arXiv:1912.00818.

[23] E. Sannara, F. Portet, P. Lalanda, V. German, A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison, in: 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE, 2021, pp. 1–10.

[24] Y. Ye, S. Li, F. Liu, Y. Tang, W. Hu, EdgeFed: Optimized federated learning based on edge computing, IEEE Access 8 (2020) 209191–209198.

[25] Q. Xia, W. Ye, Z. Tao, J. Wu, Q. Li, A survey of federated learning for edge computing: Research problems and solutions, High-Confid. Comput. 1 (1) (2021) 100008.

[26] A. Imteaj, U. Thakker, S. Wang, J. Li, M.H. Amini, A survey on federated learning for resource-constrained iot devices, IEEE Internet Things J. 9 (1) (2021) 1–24.

[27] R. Yu, P. Li, Toward resource-efficient federated learning in mobile edge computing, IEEE Netw. 35 (1) (2021) 148–155.

[28] D.C. Nguyen, M. Ding, Q.-V. Pham, P.N. Pathirana, L.B. Le, A. Seneviratne, J. Li, D. Niyato, H.V. Poor, Federated learning meets blockchain in edge computing: Opportunities and challenges, IEEE Internet Things J. (2021).

[29] R. Saha, S. Misra, P.K. Deb, FogFL: Fog-assisted federated learning for resource-constrained IoT devices, IEEE Internet Things J. 8 (10) (2020) 8456–8463.

[30] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, Y. Zhang, Privacy-preserving federated learning in fog computing, IEEE Internet Things J. 7 (11) (2020) 10782–10793.

[31] C.W. Zaw, S.R. Pandey, K. Kim, C.S. Hong, Energy-aware resource management for federated learning in multi-access edge computing systems, IEEE Access 9 (2021) 34938–34950.

[32] T. Hiessl, S.R. Lakani, J. Kemnitz, D. Schall, S. Schulte, Cohort-based federated learning services for industrial collaboration on the edge, J. Parallel Distrib. Comput. 167 (2022) 64–76.

[33] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, Internet Things 7 (2019) 100059.

[34] A. Abusitta, G.H. de Carvalho, O.A. Wahab, T. Halabi, B.C. Fung, S. Al Mamoori, Deep learning-enabled anomaly detection for IoT systems, Internet Things 21 (2023) 100656.

[35] A. Chatterjee, B.S. Ahmed, IoT anomaly detection methods and applications: A survey, Internet Things 19 (2022) 100568.

[36] T. Andrysiak, Machine learning techniques applied to data analysis and anomaly detection in ECG signals, Appl. Artif. Intell. 30 (6) (2016) 610–634.

[37] M. Gu, Y. Zhang, Y. Wen, G. Ai, H. Zhang, P. Wang, G. Wang, A lightweight convolutional neural network hardware implementation for wearable heart rate anomaly detection, Comput. Biol. Med. (2023) 106623.

[38] M. Nawaz, J. Ahmed, Cloud-based healthcare framework for real-time anomaly detection and classification of 1-D ECG signals, Plos One 17 (12) (2022) e0279305.

[39] Z. Ji, J. Gong, J. Feng, A novel deep learning approach for anomaly detection of time series data, Sci. Program. 2021 (2021).

[40] C. Cambra Baseca, S. Sendra, J. Lloret, J. Tomas, A smart decision system for digital farming, Agronomy 9 (5) (2019) 216.

[41] N. Kaur, S.K. Sood, Cognitive decision making in smart industry, Comput. Ind. 74 (2015) 151–161.

[42] S.A.A. Bokhari, S. Myeong, Use of artificial intelligence in smart cities for smart decision-making: A social innovation perspective, Sustainability 14 (2) (2022) 620.

[43] M.W. Moreira, J.J. Rodrigues, V. Korotaev, J. Al-Muhtadi, N. Kumar, A comprehensive review on smart decision support systems for health care, IEEE Syst. J. 13 (3) (2019) 3536–3545.

[44] S. Zhou, R. Zhang, D. Chen, X. Zhu, A novel framework for bringing smart big data to proactive decision making in healthcare, Health Inform. J. 27 (2) (2021) 14604582211024698.

[45] M.T. Quasim, A. Shaikh, M. Shuaib, A. Sulaiman, S. Alam, Y. Asiri, Smart healthcare management evaluation using fuzzy decision making method, 2021.

[46] D.H. Brahmbhatt, M.R. Cowie, Remote management of heart failure: an overview of telemonitoring technologies, Cardiac Fail. Rev. 5 (2) (2019) 86.

[47] N. Rieke, J. Hancox, W. Li, F. Milletari, H.R. Roth, S. Albarqouni, S. Bakas, M.N. Galtier, B.A. Landman, K. Maier-Hein, et al., The future of digital health with federated learning, NPJ Digit. Med. 3 (1) (2020) 1–7.

[48] Y. Chen, X. Qin, J. Wang, C. Yu, W. Gao, FedHealth: A federated transfer learning framework for wearable healthcare, IEEE Intell. Syst. 35 (4) (2020) 83–93, http://dx.doi.org/10.1109/MIS.2020.2988604.

[49] J. Passerat-Palmbach, T. Farnan, R. Miller, M.S. Gross, H.L. Flannery, B. Gleim, A blockchain-orchestrated federated learning architecture for healthcare consortia, 2019, arXiv preprint arXiv:1910.12603.

[50] R. Kumar, A.A. Khan, J. Kumar, A. Zakria, N.A. Golilarz, S. Zhang, Y. Ting, C. Zheng, W. Wang, Blockchain-federated-learning and deep learning models for COVID-19 detection using CT imaging, IEEE Sens. J. (2021) 1, http://dx.doi.org/10.1109/JSEN.2021.3076767.

[51] B. Yuan, S. Ge, W. Xing, A federated learning framework for healthcare iot devices, 2020, arXiv preprint arXiv:2005.05083.

[52] J. Xu, B.S. Glicksberg, C. Su, P. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, J. Healthc. Inform. Res. 5 (1) (2021) 1–19.

[53] D.C. Nguyen, Q.-V. Pham, P.N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, W.-J. Hwang, Federated learning for smart healthcare: A survey, ACM Comput. Surv. 55 (3) (2022) 1–37.

[54] D. Yu, Y. Jin, Y. Zhang, X. Zheng, A survey on security issues in services communication of microservices-enabled fog applications, Concurr. Comput.: Pract. Exper. 31 (22) (2019) e4436.

[55] A. Benayache, A. Bilami, S. Barkat, P. Lorenz, H. Taleb, Msm: A microservice middleware for smart WSN-based IoT application, J. Netw. Comput. Appl. 144 (2019) 138–154.

[56] X. Zhao, C. Huang, Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network, IEEE Access 8 (2020) 56680–56694.

[57] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, A. Erradi, Predictive autoscaling of microservices hosted in fog microdata center, IEEE Syst. J. 15 (1) (2020) 1275–1286.

[58] S. Pallewatta, V. Kostakos, R. Buyya, Microservices-based IoT applications scheduling in edge and fog computing: A taxonomy and future directions, 2022, http://dx.doi.org/10.48550/ARXIV.2207.05399, URL https://arxiv.org/abs/2207.05399.

[59] L.N.T. Thanh, N.N. Phien, H.K. Vo, H.H. Luong, T.D. Anh, K.N.H. Tuan, H.X. Son, et al., IoHT-MBA: an internet of healthcare things (IoHT) platform based on microservice and brokerless architecture, Int. J. Adv. Comput. Sci. Appl. 12 (7) (2021).

[60] A. Mseddi, W. Jaafar, H. Elbiaze, W. Ajib, Joint container placement and task provisioning in dynamic fog computing, IEEE Internet Things J. 6 (6) (2019) 10028–10040, http://dx.doi.org/10.1109/JIOT.2019.2935056.

[61] T. Sun, D. Li, B. Wang, Decentralized federated averaging, IEEE Trans. Pattern Anal. Mach. Intell. (2022).

[62] M.S. Aslanpour, S.S. Gill, A.N. Toosi, Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research, Internet Things 12 (2020) 100273.

[63] M. Aazam, S. Zeadally, K.A. Harras, Fog computing architecture, evaluation, and future research directions, IEEE Commun. Mag. 56 (5) (2018) 46–52.

[64] D.P. Abreu, K. Velasquez, M. Curado, E. Monteiro, A comparative analysis of simulators for the cloud to fog continuum, Simul. Model. Pract. Theory 101 (2020) 102029.

[65] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.

[66] kaggle.com/code/devavratatripathy/ecg-anomaly-detection-using-autoencoders, https://www.kaggle.com, https://https://www.kaggle.com/code/devavratatripathy/ecg-anomaly-detection-using-autoencoders.

[67] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, Y. Yang, Optimal edge user allocation in edge computing with variable sized vector bin packing, in: International Conference on Service-Oriented Computing, Springer, 2018, pp. 230–245.