Hindawi
Security and Communication Networks
Volume 2023, Article ID 9821793, 1 page
https://doi.org/10.1155/2023/9821793

Retraction

Retracted: FLOM: Toward Efficient Task Processing in Big Data with Federated Learning

Security and Communication Networks

Received 25 July 2023; Accepted 25 July 2023; Published 26 July 2023

Copyright © 2023 Security and Communication Networks. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] C. Wu and Y. Li, "FLOM: Toward Efficient Task Processing in Big Data with Federated Learning," *Security and Communication Networks*, vol. 2022, Article ID 5277362, 16 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 5277362, 16 pages
https://doi.org/10.1155/2022/5277362

WILEY | Hindawi

RETRACTED

Research Article

FLOM: Toward Efficient Task Processing in Big Data with Federated Learning

Chunyi Wu and Ya Li

Institute of Artificial Intelligence and Big Data, Chongqing College of Electronic Engineering, Chongqing 401331, China

Correspondence should be addressed to Chunyi Wu; wucy13@mails.jlu.edu.cn

Received 15 October 2021; Accepted 10 November 2021; Published 27 January 2022

Academic Editor: Jian Su

Copyright © 2022 Chunyi Wu and Ya Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the diversification and individuation of user requirements as well as the rapid development of computing technology, the large-scale tasks processing for big data in edge computing environment has become a research focus nowadays. Many recent efforts for task processing are designed and implemented based on some traditional protocols and optimization methods. Therefore, it is more difficult to explore the task allocation strategy that maximizes the overall system revenue from the perspective of global load balancing. In order to overcome this problem, a large-scale tasks processing approach called Federated Learning based Optimization Methodology (FLOM) for large-scale tasks processing was presented to achieve accurate task classification and overall load balancing while satisfying task allocation requirements. FLOM performs the data aggregation and establishes the personalized models by federated learning. The deep network model is designed for deep feature learning of task requests and hosts in the substrate network. The experimental results show the capability of FLOM in terms of large-scale task classification as well as allocation.

1. Introduction

Recently, edge computing [1–3] technology is in the rapid development. The large-scale tasks processing based on edge computing environment for big data [4–6] has become one of the research hotspots. In order to realize the parallel processing of large-scale tasks, we can exploit the combination mode of cloud data center [7, 8] and multiple edge computing centers to realize data sharing and multicategory tasks processing. An efficient task processing approach can achieve the parallel processing capability of multiple clusters, including cloud data center and edge computing center. On the contrary, the system may encounter the increased computing cost and load imbalance when the task processing approach based on traditional framework and optimization model [9] is in the face of the problem of multiple computing centers and large-scale multicategory tasks. Therefore, it is necessary to explore an efficient task processing approach in the process of big data processing.

In recent years, some new computing paradigms have attracted wide attention and developed rapidly. Nowadays, with the increasing number of task requests from users and the diversification of task types, the combination of different domains has become a trend to overcome this issue. As shown in Figure 1, the purpose of this work is to combine the idea of distributed computing with the cutting-edge artificial intelligence [10–12] to realize the technology integration of multiple fields, so as to complete the reasonable allocation of large-scale tasks in the edge computing environment of big data [13, 14]. Our goal is to design and implement an efficient task processing approach to achieve accurate task classification and global load balancing, so as to reduce the cost of task processing.

Based on this motivation, we propose a large-scale task processing approach for big data processing based on the idea of federated learning in the edge computing environment. It employs the federated learning framework to realize the data interaction and model parameter updating of multiple edge computing centers and the cloud center

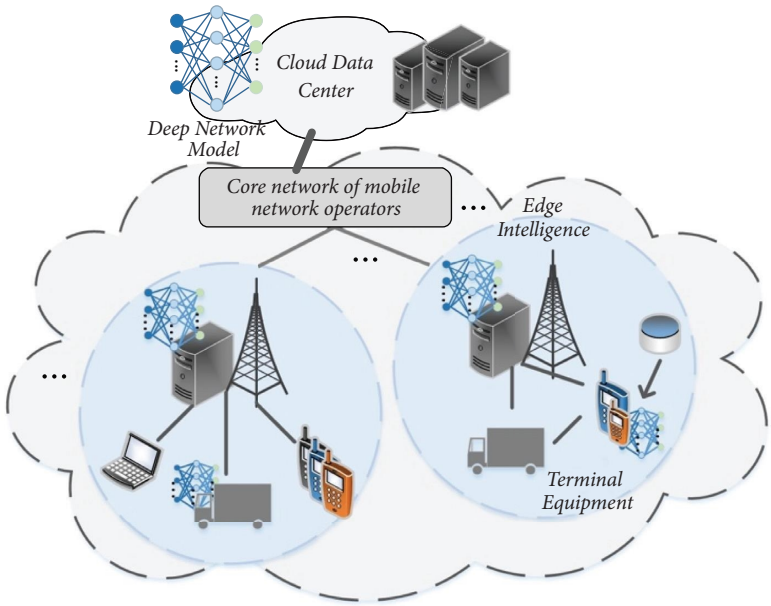

Figure 1: Motivation of the proposed approach FLOM.

without exposing private data and realizes multicategory tasks processing and global load balancing from the whole system level. At the same time, the deep feature learning method is used to fully analyze the features of the substrate network resources and the current task requests, construct the feature matrix as the input of the federated learning model based on the deep convolutional network, and obtain the probability distribution of a certain task assigned to each node in the edge computing centers and the cloud as the output of the model, so as to realize the large-scale tasks allocation.

The main contributions of this work are as follows:

(1) Based on the current situation that the number of user task requests is increasing rapidly, and the task categories are diversified, a large-scale task processing approach based on federated learning for big data in the edge computing environment is proposed;

(2) A federated task processing model based on federated learning framework and deep feature learning is proposed. Task requests are classified and identified according to the resource requirements of task requests and the substrate network resources, so as to realize the reasonable allocation of tasks and global load balancing, as well as to reduce the task execution overhead;

(3) Experiments have shown that FLOM can achieve large-scale task classification and load balancing. Compared with the many advanced approaches, it convincingly demonstrates the superiority of FLOM in processing large-scale tasks for big data in the edge computing environment.

The rest of this work is as follows: in Section 2, the related work of federated learning, deep learning, and edge computing technology is introduced. Section 3 describes the problems to be solved in this work. In Section 4, we formalize the proposed problem. Section 5 introduces the design and implementation of the proposed approach in detail. In Section 6, the experimental results are obtained through multiple sets of experiments, which validates the effectiveness of the proposed approach. And this work is discussed herein. At last, in Section 7, we draw the conclusion of this work and elaborate the future work.

2. Related Work

In this section, we introduce the related work in the following aspects: federated machine learning, deep learning, and the edge computing technology.

2.1. Federated Machine Learning. In the comprehensive technology survey [15] on the federated machine learning, researchers indicate that the optimization method based on federated learning framework is firstly presented by Google [16] in 2016. In [16], researchers introduced a new setting for the distributed optimization called *federated optimization*. The motivation of this idea is to keep the training data locally on users' mobile devices instead of logging it to the uniform data center for the training. It achieves the distributed mobile phone applications in a high-quality centralized model with user data protection. After that, Kaissis [17] et al. presented a secure, privacy-preserving and federated machine learning in medical imaging. It aims to improve patient care and address the demands for data protection, while the utilization is mandatory. Lu [18] et al. proposed a blockchain empowered secure data sharing framework for the distributed computing. They transform the problem of data sharing into a machine learning problem with privacy-preserved federated learning. At last, the computing jobs for the consensus in the permissioned blockchain can be employed for the federated learning. Yang [19] et al.

RETRACTED

proposed a novel over-the-air computation approach to achieve the efficient global model aggregation for on-device distributed federated machine learning in right of harnessing the signal superposition property of the wireless multiple-access channels. We can observe that the federated machine learning has the capability to overcome the problems of data islanding through privacy-preserving model training.

2.2. Deep Learning. An authoritative overview [20] on the recent advances in deep learning indicates that Deep Learning (DL) was first introduced into Machine Learning (ML) in 1986 and then used in Artificial Neural Networks (ANN) in 2000. Deep learning is composed of multiple hidden layers to learn data features with multiple abstraction layers [21]. Deep learning [22–24] belongs to a subfield of machine learning. It exploits multilevel nonlinear information processing and abstraction, which is used for feature learning, representation, classification, regression, and pattern recognition for supervised, unsupervised, semi-supervised, and self-supervised learning [25]. With the rise of deep learning, it is gradually studied and used in various fields. Li [26] et al. proposed a deep convolutional computing model (DCCM) to learn the hierarchical features in big data in right of the tensor representation model to extend the convolutional neural network from the vector space to the tensor space. Yuan [27] et al. presented a deep layer-wise supervised pretraining framework for the quality-relevant feature extraction and soft sensor modeling. It is based on the stacked supervised encoder-decoder. Lin [28] et al. proposed a graininess-aware deep feature learning approach to achieve the pedestrian detection. Different from the existing pedestrian detection approaches, the researchers incorporate the fine-grained information into the convolutional features to increase the discrimination for the parts of human body. In short, with the continuous development of deep learning, researchers have integrated the most cutting-edge technologies such as reinforcement learning and transfer learning to achieve multiple domain integration and create new computing paradigms.

2.3. Edge Computing Technology. Shi [29] published an authoritative review on edge computing technology in 2016, which indicates that edge computing is a kind of enabling technology allowing computation to be performed in the edge of networks. Herein, the upstream data represents the IoT services, and the downstream data represents the cloud services. Since edge computing was proposed, it has been customized and used in various fields. Yang [30] et al. presented a novel model of energy consumption of offloading from task computation and communication in consideration of the small-cell network architecture for task offloading. In [31], Mukherjee et al. took into consideration the cell-free massive MIMO framework with implementing the mobile edge computing functionalities. The successful edge computing probability for the target computation latency has been presented after deriving successful computing and communication probabilities via the stochastic geometry and queueing theory. Tang [32] et al. presented a novel location prediction approach an smart caching strategy based on ML for the user interests prediction, which can drive the content of user interests from the servers to the edge nodes. It can be seen that with the rapid development of edge computing technology and artificial intelligence, edge intelligence, as a new paradigm of edge computing and artificial intelligence enabling each other, will give birth to a large number of innovative research opportunities and has broad application prospects in many fields such as intelligent Internet of things, intelligent manufacturing, and smart city.

3. The Proposed Problem

In the edge computing center as well as the cloud data center, the system receives large-scale tasks from big data and assigns them onto the physical hosts in the resource pool. In general, the system assigns task requests according to a certain strategy. Task allocation is difficult to execute, or overload of the hosts occurs when task requests are assigned to hosts with resource surplus equal to or less than resource requests. The task request cannot be executed or consume additional computing resources and bandwidth resources to call data and interfaces from other servers for task execution when the task type does not match the processing task type of the processing node. It may cause the imbalance of system load and the failure of personalized service, thus reducing the resource utilization rate and even the service efficiency. On the other hand, there is an information isolated phenomenon among the processing units in the substrate network. It is difficult to share the information of users and processing nodes in the system reasonably without damaging the data privacy and security, so that the system cannot establish a unified model for efficient identification and processing of task requests. Figure 2 shows a brief process of task allocation in the federated cloud data center with edge computing. There is no doubt that an optimal large-scale task allocation approach should be able to make the whole system share users underlying data reasonably, classify tasks individually, and possess better load balancing effect. To sum up, in order to overcome the current problem of task requests with huge volume and various types of requirements, it is necessary to design an efficient task allocation approach to achieve personalized task classification and system load balancing in the edge computing environment.

4. Preliminaries

4.1. System Model. Consider a federated model (D, EN, TR, L_c, L_{mem}, L_{net}, L_{dsk}, R_c, R_{mem}, R_{net}, R_{dsk}) in every Δt time, where D represents the current set of physical hosts in the cloud data center, $D\ (l,\ t) = \{d_1,\ d_2,\ \ldots,\ d_l\}$, t represents the start time of task allocation, EN is the set of edge nodes for the task processing that is divided into v edge computing centers, EN $(m,\ t) = \{\text{en}_1,\ \text{en}_2,\ \ldots,\ \text{en}_m\}$, and TR is the set of task requests, TR $(n,\ \Delta t,\ t) = \{\text{tr}_1,\ \text{tr}_2,\ \ldots,\ \text{tr}_n\}$. L_c is the set of the current remaining CPU resource mount of m hosts in the set EN and l hosts in the set D, $L_c\ (m,\ l,\ t) = \{L_c^1,\ L_c^2, \ldots, L_c^m,\ L_c^m,\ L_c^{m+1}, \ldots, L_c^{m+l}\}$ after integrating D and

RETRACTED

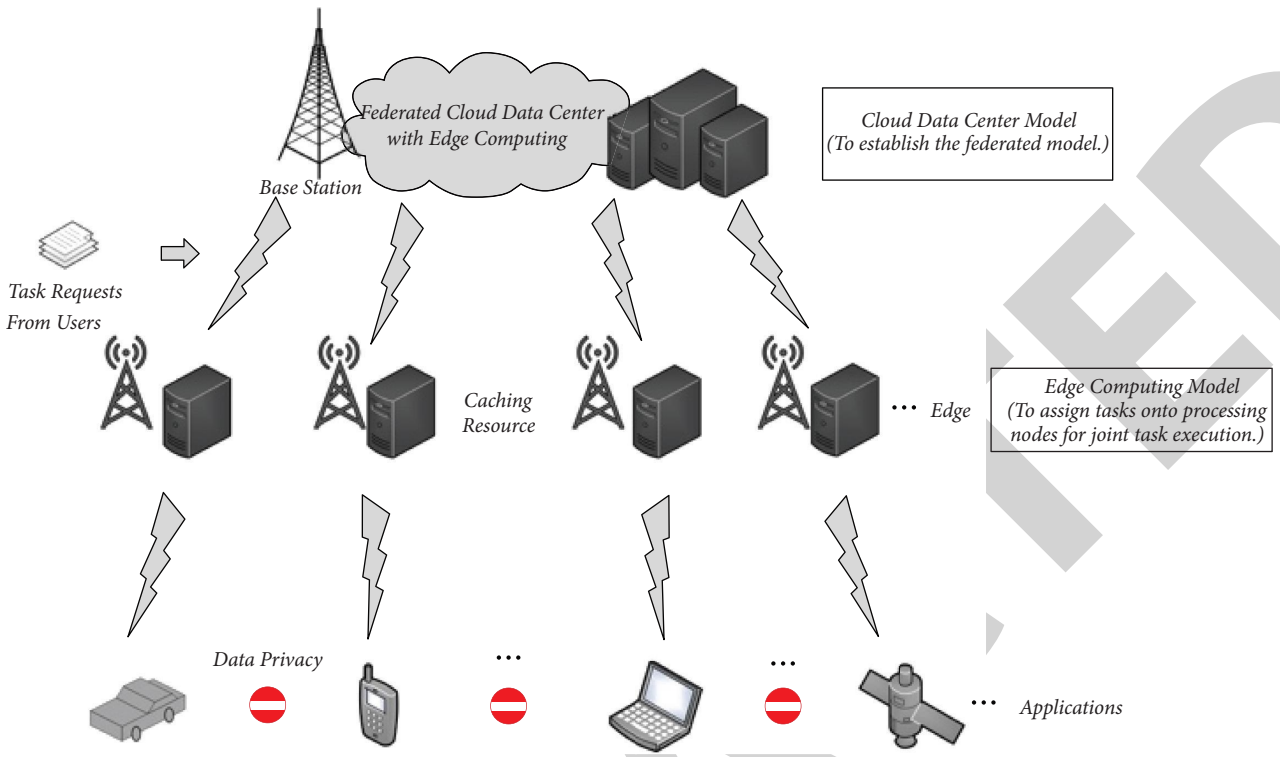

Figure 2: The proposed problem.

EN into a resource pool with $m + l$ physical hosts. L_{mem} is the set of the current remaining memory resource amount of the resource pool composed of physical hosts in the edge computing center and the cloud data center, $L_{\text{mem}}(m, l, t) = \{L^1_{\text{mem}}, L^2_{\text{mem}}, \ldots, L^m_{\text{mem}}, L^{m+1}_{\text{mem}}, \ldots, L^{m+l}_{\text{mem}}\}$. L_{net} is the set of the maximum available communication resource amount of the $m + l$ hosts, $L_{\text{net}}(m, l, t) = \{L^1_{\text{net}}, L^2_{\text{net}}, \ldots, L^m_{\text{net}}, L^{m+1}_{\text{net}}, \ldots, L^{m+l}_{\text{net}}\}$. L_{dsk} is used to represent the current remaining disk resource amount of the hosts, $L_{\text{dsk}}(m, l, t) = \{L^1_{\text{dsk}}, L^2_{\text{dsk}}, \ldots, L^m_{\text{dsk}}, L^{m+1}_{\text{dsk}}, \ldots, L^{m+l}_{\text{dsk}}\}$. Similar to the current remaining resource amount of the physical hosts, $R_c(n, t) = \{R^1_c, R^2_c, \ldots, R^n_c\}$ is the requested CPU resource amount of the n task requests in the set TR. R_{mem} is the requested memory resource amount of the tasks in set TR, $R_{\text{mem}}(n, t) = \{R^1_{\text{mem}}, R^2_{\text{mem}}, \ldots, R^n_{\text{mem}}\}$. $R_{\text{net}}(n, t) = \{R^1_{\text{net}}, R^2_{\text{net}}, \ldots, R^n_{\text{net}}\}$ is the requested maximum communication resource amount of the n task requests TR. $R_{\text{dsk}}(n, t) = \{R^1_{\text{dsk}}, R^2_{\text{dsk}}, \ldots, R^n_{\text{dsk}}\}$ is the requested disk resource amount of the n task requests in TR.

So as to achieve the load balancing of the edge computing center and the cloud data center, in this work, the standard deviation of all hosts' residual workload rate is employed to measure the load balancing degree of the processing nodes in the whole system. The available remaining resource amount of a physical host L_p can be defined as follows:

$$L_p = c_1 L^p_c + c_2 L^p_{\text{mem}}, \quad p \in \{1, 2, 3, \ldots, m + l\}, \quad c_1 + c_2 = 1. \tag{1}$$

Here, L_p is employed to represent the remaining computing capability of the physical host p. c_1 is the weight value of CPU, and c_2 is the weight value of memory. It is noted that L_c and L_{mem} are used to measure the amount of remaining resource and task requested resource in this work. The maximum requested resource amount in the set TR can be defined as follows:

$$L_{\text{mreq}} = \max^n_{q=1} c_1 R^q_c + c_2 R^q_{\text{mem}}. \tag{2}$$

The residual workload rate of a host can be defined as follows:

$$U_p = \frac{L_p}{W_p}, \quad p \in \{1, 2, 3, \ldots, m + l\}, \tag{3}$$

where W_p is the total computing capability of physical host p and it can be defined as follows:

$$W_p = c_1 W^p_c + c_2 W^p_{\text{mem}}, \quad p \in \{1, 2, 3, \ldots, m + l\}, \tag{4}$$

where W^p_c indicates the total CPU resource amount of host p and W^p_{mem} indicates the total memory resource amount of host p, respectively. The expectation and the standard deviation of all hosts' residual workload rates can be described as follows:

$$E(U) = \frac{\sum^{m+l}_{p=1} U_p}{m + l}. \tag{5}$$

On the basis of the above work, the proposed optimization objective for load balancing can be denoted as follows:

RETRACTED

$$\omega = \sqrt{\frac{1}{m+l}\sum_{p=1}^{m+l}\left(U_p - E(U)\right)^2}. \quad (6)$$

Minimize

$$\omega = \sqrt{\frac{1}{m+l}\sum_{p=1}^{m+l}\left(\frac{c_1 L_c^p + c_2 L_{\text{mem}}^p}{c_1 W_c^p + c_2 W_{\text{mem}}^p} - \frac{\sum_{r=1}^{m+l}\left(\left(c_1 L_c^r + c_2 L_{\text{mem}}^r\right)/\left(c_1 W_c^r + c_2 W_{\text{mem}}^r\right)\right)}{m+l}\right)^2}, \quad (7)$$

where r represents an auto-increment integer along with the task allocation that reflects the state of task allocation in a certain time.

4.2. Task Category. For the reasonable task allocation to improve the efficiency of task processing, we can define the task categories according to different hardware requirements before performing the task classification. Herein, load generation tools are used to construct the task training datasets, as well as to test and compare the performance data obtained from the operation of each server in the cloud data center and edge computing center. The intelligent algorithm is employed to establish the historical data training set with classification labels, and the specific results are described in Table 1.

It is noted that there is a certain relationship between the task type with its protocol and the requirements of processor, memory, disk, network bandwidth, etc. For instance, the tasks prone to document transmission are in need of more bandwidth resources, including application layer protocols such as file Transfer Protocol (FTP). The tasks in streaming media form starve for more resources of processors and bandwidth, including protocols such as Real-time Transport Protocol (RTP), as well as Real-time Streaming Protocol (RTSP). According to the above actual situation, this work achieves the task classification for allocation on the basis of resource types.

4.3. Definition of the Federated Model. Consider that we accumulate the data of n task requests from both the cloud data center and the edge computing center, which can be integrated as the task datasets denoted by $U = \{U_1, U_2, \ldots, U_{v+1}\}$. In many cases in the past, we can train a model by combining these task datasets in right of $U' = U_1 \cup U_2 \cup, \ldots, \cup U_{v+1}$ for the general machine learning model construction [33]. However, in this work, it is noted that we need to establish a federated cloud center with edge computing model DE_{fed} based on federal learning by collaborating the datasets without any leakage of task data to each other. If we formulate labels of task categories as a target-domain SD_n: $= \{x_n^q\}_{q=1^{Nn}}$ according to the hosts of physical resource pool, given the above setting, the objective is to establish a federated learning model to predict labels for the target-domain party as accurately as possible, where any models do not expose their datasets to each other [34].

4.4. Security Definition. In consideration of the privacy issue in many applications such as manufacturing, finance, and stocks, we know that all parties in the federal model can be dishonest. Assume a mapping $(Y_A, Y_B) = V(X_A, X_B)$, where V represents the two-party computation between the parties A and B, Y_A and Y_B are A's and B's private inputs, respectively, and Y_A and Y_B are their outputs, respectively. Herein, the protocol V can be secure against dishonest B if there is an infinite number of (X_A', Y_A') pairs such that $(Y_A', Y_B) = V(X_A', X_B)$. In the meanwhile, the protocol V can be secure against dishonest A if there is an infinite number of (X_B', Y_B') pairs, such that $(Y_A, Y_B') = V(X_A, X_B')$ [35]. The security definition has provided an effective control strategy for privacy-preserving in the process of establishing the federal model based federated cloud model with edge computing nodes.

5. The Proposed Approach

5.1. Design of the System Architecture. Figure 3 describes the framework details of the proposed approach in the edge computing environment. It intuitively shows the relationship between the cloud server and the local edge computing nodes. Firstly, the user data from different kinds of task requests in the substrate network is uploaded to the cloud data center and integrated with feature data of task processing nodes in the cloud data center and edge computing centers as the public dataset for initial model training. Then, each edge computing center downloads the initial model to the local server and updates the edge computing model with the real-time updated personalized user data. We upload the updated user model (edge computing model) to the cloud data center to update DE_{fed} to complete the encryption parameter sharing among the edge computing nodes that train and process different types of task data without affecting the privacy of user data and then realize the federated model training by the way of interaction between the federated model and the edge computing model. In the model application stage, the federated model classifies and identifies real-time task requests according to the federated model trained by historical dataset. It assigns task requests onto the optimal edge task computing node or directly sends them to the cloud data center for efficient processing. It can be seen that a "Lotus shaped" federated model for big data processing in the federal cloud-edge data center is established, which is named as "Lotus model."

RETRACTED

Table 1: Description of task categories.

Type of test	Task category	Tools
CPU	CPU bound	Whetstone
Memory	Memory bound	RunMemtestpro
CPU&Memory	Computation bound	Whetstone & Run Memtest pro
Network bandwidth	Communication bound	Small Bits
Writing files in the local system	Disk bound	DiskSpd
Comprehensive testing	Full-load bound	WebBench

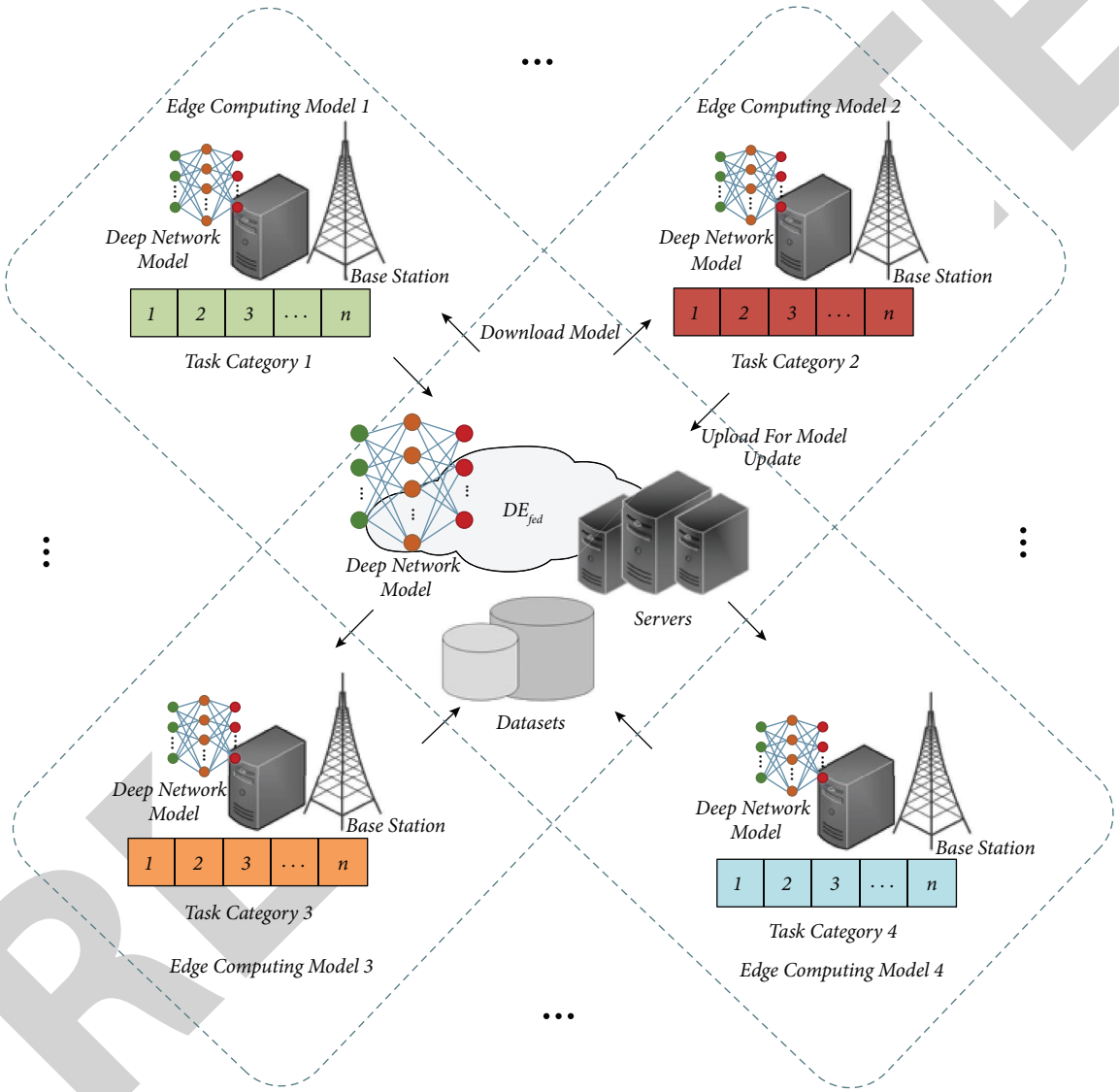

Figure 3: The view of FLOM's architecture "Lotus model."

Aiming at the problem of efficient large-scale task processing in big data environment, this work presents a novel task processing approach FLOM that can realize reasonable task allocation as well as the computation and bandwidth resources management through task classification and identification. According to the features of cloud data center with the surrounding edge computing nodes and the task requests, FLOM carries out deep network model training for local task requests processing. On the premise of ensuring the privacy of each user's information, in order to meet the task requirements, achieve reasonable task allocation, and overcome the problem of information island between processing units in the substrate network, this work exploits the federated learning method to train and update the federated model and edge computing model. In the meanwhile, this work designs the elaborate federated model "Lotus model." In the process of model training, each user model (edge computing model) and cloud data center model

are integrated to establish the federated model, which achieves the iterative update between the federated model and the edge computing models. In the application process of the model, the federated model calculates the optimal task allocation scheme according to the type and demand of task requests, so as to realize efficient task processing, achieve the system load balancing, and improve the resource utilization rate of cloud data center and edge computing nodes.

5.2. Feature Extraction. After the system and task category description, we need to extract features of task requests and physical hosts in the federated cloud center with edge computing to construct feature matrixes as the input of our deep network model. It is worth noting that there exist many features that can describe the cluster state and task requirements, and the inaccurate feature extraction cannot achieve accurate description. If the number of features is too small, the task cannot be described comprehensively, whereas if the number of features is too large, the load of data acquisition tools and the complexity of deep network model will be increased. Hereupon, according to resource management pattern in Alibaba Cloud Service and Google Cloud Service [36], we select more than twenty key features from the attributes of each task request and host, including CPU capacity provisioned (MHz), CPU usage (MHz), memory capacity provisioned (kB), memory usage (kB), disk read throughput (kB/s), disk write throughput (kB/s), network received throughput (kB/s), network transmitted throughput (kB/s), CPU capacity required (MHz), memory capacity required (kB), network resource request (kB/s), disk resource request (kB/s), the ratio of the available computing resource of the physical node to the amount demanded of computing capability of a task request, and the ratio of available CPU amount of the physical node to the total amount of available CPU resource in the cloud data center or edge computing center, etc. By extracting the features from current n task requests and $m+l$ hosts during a time window Δt, we can normalize and integrate the obtained features to construct a $(m+l) * 23$-dimensional feature matrix for the input of deep network model.

5.3. Federal Learning Process. Our designed federated cloud model with edge computing adopting the typical architecture and federal learning process aims to realize the encrypted model training and sharing for task classification and overall load balancing of system using large-scale masking data from kinds of task requests. In general, there are two main entities in the typical federal learning system, i.e., the data owners and the model owner. Herein, we can assume that the edge computing models run as the data owners and the cloud data center model operates as the model owner. We aim to collaboratively train a shared model (viz. the cloud data center model) while avoiding the user data exposing to the other users, which achieve the federal model training with privacy-preserving. The whole process differs from the traditional model training, which aggregates and integrates individual resource to complete model establishing in a centralized way.

As for the federated model training, the specific process of model training is as follows:

Step 1: since we have employed the deep network model to train the task classification model in this work, thus, we need to construct feature matrixes for input of the model. We elaborately select twenty-three key features from the attributes of the received task requests and physical hosts that include CPU capacity provisioned, CPU usage, and memory capacity provisioned via the distributed data acquisition tool Ganglia [37] during tons of tasks processing. Herein, aiming at reflecting the interaction between the current task requirement and available resources in the historical data set, as well as the interaction between the current resource surplus of a certain task processing node and the total resource surplus in the cloud data center or edge computing center, we design and calculate the following indicators in terms of CPU as two important elements in the input features, which are also applicable to memory, network, and disk resources.

$$\begin{aligned} \text{ratio}_{\text{CPU}}^{q/p} &= \frac{R_c^q}{L_c^p}, \\ \text{ratio}_{\text{CPU}}^{p/\text{total}} &= \frac{L_c^p}{\text{total}_c}. \end{aligned} \tag{8}$$

After extracting the features of physical hosts and task requests, we need to carry out normalization to accelerate the training process and enable the algorithm to converge quickly. In order to clearly reflect the probability distribution of each task assigned to each physical host in the output of deep network model, we use the normalized values to construct a feature matrix M_q composed of features from a certain task and all the available hosts, which can be allocated to each feature input process.

RETRACTED

$$M_q = \begin{bmatrix} \text{CPU cores}_1 & \text{CPU capacity provisioned}_1 & \text{CPU usage}_1 & \cdots & \text{ratio}_{\text{CPU}}^{q/1} & \cdots & \text{ratio}_{\text{dsk}(w)}^{1/\text{total}} \\ \text{CPU cores}_2 & \text{CPU capacity provisioned}_2 & \text{CPU usage}_2 & \cdots & \text{ratio}_{\text{CPU}}^{q/2} & \cdots & \text{ratio}_{\text{dsk}(w)}^{2/total} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \text{CPU cores}_p & \text{CPU capacity provisioned}_p & \text{CPU usage}_p & \cdots & \text{ratio}_{\text{CPU}}^{q/p} & \cdots & \text{ratio}_{\text{dsk}(w)}^{p/\text{total}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{CPU cores}_{m+l} & \text{CPU capacity provisioned}_{m+l} & \text{CPU usage}_{m+l} & \cdots & \text{ratio}_{\text{CPU}}^{q/(m+l)} & \cdots & \text{ratio}_{ds\ k(w)}^{(m+l)/\text{total}} \end{bmatrix}. \tag{9}$$

Step 2: as the public data set from task requests and physical hosts has been integrated, the cloud data center gets ready for the parameter initialization and training of the cloud data center model. Herein, cloud data center employs the deep network model constructed by Convolutional Neural Network (CNN), and it specifies the hyperparameters of the federated cloud center with edge computing model DE_{fed}, i.e., learning rate, batch size, and dropout rate. The cloud date center model is trained based on the public datasets. Subsequently, the cloud data center will distribute the initialized DE_{fed} among the edge computing centers to lead the edge computing center models' training for local models establishing.

Step 3: taking advantage of the obtained cloud data center model, each edge computing center model can exploit its local data from users' task requests and physical hosts in its own cluster to update the parameters of the deep network model. In the model DE_{fed}, we adopt deep neural model to learn both the cloud data center model and edge computing center models. Herein, the deep networks are employed to achieve feature learning and the task requests classification in right of inputting feature matrixes from the raw task request data. The goal of the model learning is to find the optimal parameters o_w^z that can minimize the loss function $L(o_w^z)$:

$$o_w^{z\,*} = \arg\min_{o_w^z} L(o_w^z), \tag{10}$$

where w is the serial number of the models in the clusters including v edge computing centers and the cloud data center. Hereupon, o_w^z represents parameters of the w th model in the iteration z. And then, the updated parameters of edge computing models will be sent to the cloud data center.

Step 4: in this step, we combine cloud data center model with the edge computing model to establish the federated model DE_{fed}. The trained cloud data center model aggregates the edge computing center models from clusters and then broadcasts updated federated model parameters to the v clusters. Herein, the cloud data center aims to minimize the global loss function $L(o_w^z)$ for the designed federated model.

$$L(o_G^z) = \frac{1}{v+1} \sum_{w=1}^{v+1} L(o_w^z). \tag{11}$$

Specifically, this process can be used to minimize the loss for our deep neural networks. We assume that f_{DE} () is the federated model to be learned. It also can be treated as a prediction function using deep network in the task classification. Thus, the learning objective is as follows:

$$\arg\min_{o_w^z} L = \sum_{w=1}^{v+1} l(y_w, f_{\text{DE}}(x_w)), \tag{12}$$

where l () is the loss for the deep neural network. $\{x_w, y_w\}_{w=1^{v+1}}$ are the task data samples from the training datasets. In this work, we employ the cross-entropy loss for the task requests classification.

$$\arg\min_{o_w^z} L = \sum_{w=1}^{v+1} l(y_w^z, f_{\text{DE}}(x_w^z)). \tag{13}$$

In the overall learning process, Steps 3-4 will be repeated until the global loss function $L(o_G^z)$ converges to an acceptable range or a desired task classification accuracy is obtained.

Note that we have employed CNN based deep neural model to train both cloud data center model and edge computing center model. We can take the feature matrix as the input of the deep model and the probabilities of mapping task requests onto physical hosts as the output. The basic components of the designed network for our proposal are shown in Figure 4. It consists of an input layer, convolutional layers, pooling layers, and a fully connected layer. We determine the number of hidden layers via sets of experimentations since the optimal number of hidden layers of deep neural network belongs to an open research question. Herein, we explore the performance of the designed model by changing the number of hidden layers from 1 to 5.

Generally, in the practical application, the result from full connection layer is transmitted to the output layer for the probability distribution of each task request that indicates the probability of obtaining a desired performance benefit after allocating the task request onto a certain physical host in the cluster. The output layer contains a softmax function, which is a generalization of the logistic regression. The probability Pr_p can be expressed as

RETRACTED

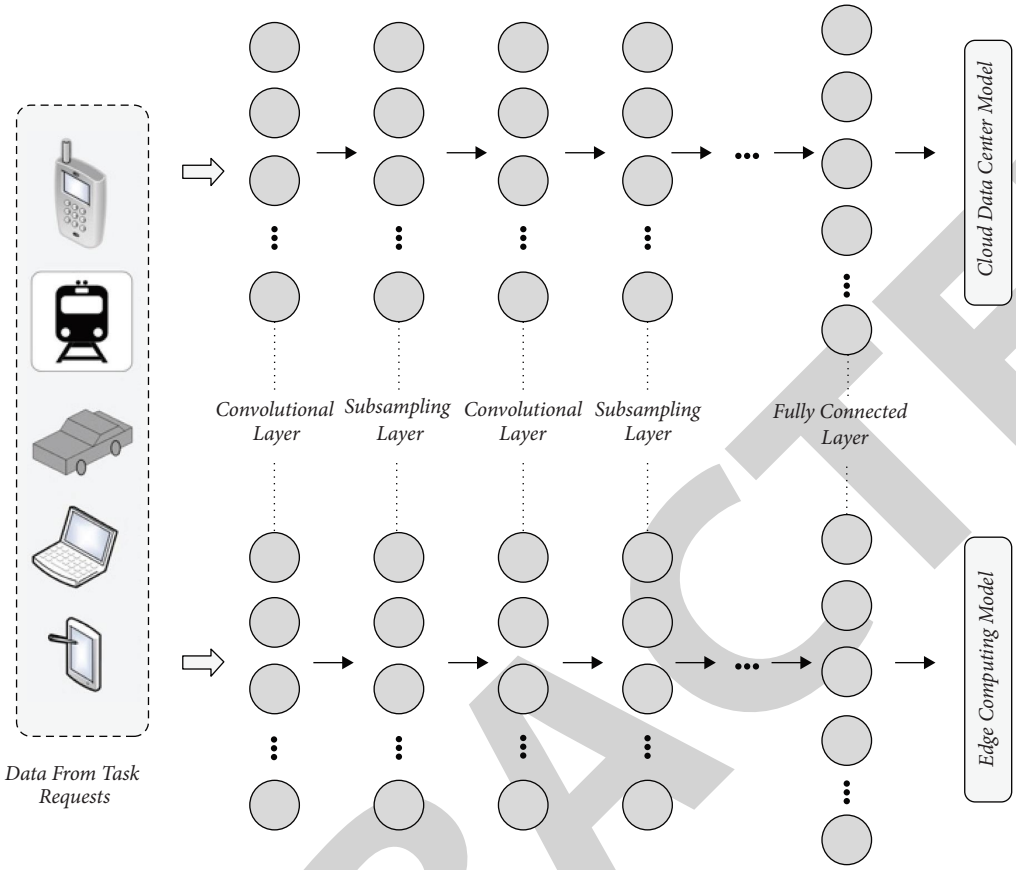

Figure 4: The deep learning process of FLOM.

$$\Pr_p = \frac{e^{v_p^l}}{\sum_K e^{v_K^l}}, \tag{14}$$

where v_p^l is the pth output of the lth convolutional layer. Using the softmax function, the multidimensional vector can be transformed into real values from 0 to 1 and their sum is 1.

As for the overfitting due to the lack of public available labeled training samples that limits the performance for big data feature learning, an adaptive distribution function is established to set the activating rate in each hidden layer with a probability ξ. It is always in accordance with the law of human cognition of things when we use the normal distribution to explain the law of nature within the widely used distribution functions. Herein, we cannot directly employ the normal distribution to set the activating rate of each hidden layer since it is a monotonically increasing function. Thus, we introduce a distribution model ξ in the lth layer in the deep neural network as follows:

$$\xi = \begin{cases} 1 - \dfrac{1}{\delta\sqrt{2\pi}} \displaystyle\int_{-\infty}^{l} \exp\left(-\frac{(l-(L/2))^2}{2\delta^2}\right) dl\,, & L = 2k; \\ 1 - \dfrac{1}{\delta\sqrt{2\pi}} \displaystyle\int_{-\infty}^{l} \exp\left(-\frac{(l-(L+1)/2)^2}{2\delta^2}\right) dl, & L = 2k+1. \end{cases} \tag{15}$$

where L presents the total number of the layers in the deep neural network, and δ is the parameter for decay velocity control. The parameter of the deep network model can be trained by the combination of higher-order backpropagation and the designed distribution function. The specific process can be found in [38].

5.4. Additively Homomorphic Encryption. In our proposed federated model, the direct information sharing of task data is forbidden. In this work, for the privacy security, we employ the additively homomorphic encryption. It can avoid the task information leakage. The arithmetic for elements of plaintext space can be provided by the additively homomorphic encryption scheme. An operation producing the encryption of the sum of two numbers as well as calculating the encryptions of the numbers can be provided herein. We can assume that the encryption of a real number a be $<a>$. Hereupon, for any two plaintexts a and b, we have $<a> + <b> = <a + b>$. Also, a ciphertext can be set to multiply with a plaintext instead of the repeated addition, $b \cdot <a> = <ba>$. Herein, b has not been encrypted. In this case, the sums and products of plaintexts and ciphertexts can be calculated without leaving the encrypted number space [33]. Similarly, we can introduce this operation into the vector and matrix manipulation. Hereupon, we can employ $b^T <a> = <b^T a>$ to calculate the inner product of two vectors of plaintexts a and b as well as $b \circ <a> = <b \circ a>$ to compute

RETRACTED

the component-wise product, respectively. Herein, we will not describe the details of matrix manipulation since matrix operations are similar to vector operations. The parameter sharing can be achieved without leaking any information from the edge computing centers. Taking advantages of our federated learning model, we can aggregate the task request data without impairing the privacy security.

6. Experiments

In this section, we will evaluate the proposed approach on the dataset composed by the task requests simulated by Whetstone [39]. Our approach is compared with the existing approach FIFO, Fair Scheduler, and Capacity Scheduler [40] based on the following indicators:

(1) Effectiveness of task requests classification
(2) Load balancing effect
(3) Performance under stress circumstances
(4) External service performance

6.1. Implementation Details. In order to validate the performance of our proposed federated model under data silos, we need to collect task requests from different users for classification and identification. We exploit the simulation tools to generate the certain kinds of task requests in parallel with different resource requirements continuously getting to cloud data center and edge computing centers to compose the synthetic task dataset. The task requests are divided into the training set and testing set herein. It is worth noting that they can be used to establish labeled dataset during the model training, while the partial labeled dataset during semisupervised learning is for the model testing. The hyperparameters of all the comparison approaches are tuned by using the cross-validation. In terms of building the deep network model mentioned in Section 5, the deep learning framework TensorFlow is exploited to achieve the modeling. We employ the Paillier additively homomorphic encryption [41] for the privacy security in python. The proposed federated model fetches the resource information and state of cloud data center and edge computing centers regularly.

During the model training, there are 32 convolution kernels in the first convolution layer. The size of each convolution kernel is set to 6×6 and the stride size is set to 1. We set the matrix size of the maximum pooling layer as 2×2 as well as the step size as 2. There are 64 kernels in the second convolutional layer, and other parameters are the same as the first layer. The training time of the designed model is usually 2-3 hours. We set the learning rate of model as 0.001. What is more, Adam optimization method is introduced to update the step size dynamically.

OpenStack [42] is employed to create virtualization scenarios for deployment of the big data processing framework Spark [43]. All the experiments are carried out in the same Linux workstation with an Intel Xeon with the configuration of three 3.4 GHz CPUs and 512 GB memory. In this work, 32 processing nodes with different configuration are created as the edge computing nodes and resource pool in the cloud data center. The latest version of Spark 2.4.6 is used in this work, which needs to be built on the basis of Hadoop.

6.2. Comparison in Effectiveness of Task Requests Classification. The indexes of classification accuracy during task requests identification for each subject are shown in Tables 2 and 3. The mean results obtained by multiple sets of experiments are shown in Table 4. The results indicate that our proposed federated model can achieve decent classification performance in terms of precision, recall, and F-measure on all the task requests from users. Compared to the other approaches, the proposed FLOM approach significantly improves the accuracy in case of 600, 1200, 1800, 2400, 3600, and 4500 task requests.

As for the precision, the proposed FLOM approach achieves 0.8857 in case of 3600 task requests that is the best result among all the datasets in different scales. We receive the best result of recall (0.8891) in the case of 600 task requests, while the best results of F-measure (0.8805) are in the case of 4500 task requests. Although there is no result obtained in the different data scales that possesses the absolute advantage in all three indicators, overall, we can still get a relatively decent result (0.8762, 0.8848, 0.8805) in the case of 4500 task requests. It significantly demonstrates the effectiveness of our proposed federated model for task processing in big data.

Figure 5 shows the comparison of four kinds of approaches in accuracy with 3-layer, 4-layer, and 5-layer deep neural network, respectively. Overall, the proposed FLOM approach shows its advantage of accuracy for task request classification in three kinds of deep network architectures herein except for the result in the 5-layer neural network with 2400 task requests. Specifically, as shown in Figure 5(a), in the 3-layer architecture, the accuracy of these approaches fluctuates with the change of task request scale. The results of Fair Scheduler and Capacity Scheduler are similar, but they are always better than FIFO. As for FLOM, its results are always better than the other three approaches. The maximum value of accuracy is obtained when the task scale is 1800, and the minimum value is obtained when the task scale is 4500. In Figure 5(b), overall, the results in terms of accuracy of all algorithms are almost stable from the beginning to the end in the 4-layer architecture deep network. In the initial stage, the results of Fair Scheduler are better than those of FIFO and Capacity Scheduler. However, the results of Capacity Scheduler begin to be better than Fair Scheduler from the task scale of 1800. The results of Fair Scheduler and Capacity Scheduler are always better than FIFO in the whole process. For FLOM, it shows the performance advantages over the other three approaches in the overall process. The maximum value of accuracy is obtained when the task scale is 2400, and the minimum value is obtained when the task scale is 1200. Figure 5(c) has shown the specific classification results of four approaches in the case of 5-layer neural network. FLOM still maintains the performance advantage in task request classification except for the task scale of 2400. Its result is only a little lower than Capacity Scheduler, but

RETRACTED

Table 2: Four types of task request classification results.

	Positive	Negative
The task has been correctly classified.	*tp*	*fp*
The task has not been correctly classified.	*fn*	*tn*

Table 3: Measuring accuracy of task request classification.

Indexes	Calculation formulas
Precision	$p = \text{Num}(tp)/(\text{Num}(tp) + \text{Num}(fp))$
Recall	$r = \text{Num}(tp)/(\text{Num}(tp) + \text{Num}(fn))$
F-measure	$F = 2 \times (p \times r)/(p + r)$
Accuracy	$\text{Accuracy} = (\text{Num}(tp) + \text{Num}(tn))/\text{Total}$

Num (*tp*) denotes the number of *tp*s, where *tp* denotes that tasks are successfully classified and assigned onto the target cluster.

Table 4: Comparison in effectiveness of task requests classification.

	Precision	Recall	*F*-measure
600 task requests	0.8692	0.8891	0.8790
1200 task requests	0.8592	0.8841	0.8715
1800 task requests	0.8694	0.8750	0.8722
2400 task requests	0.8665	0.8831	0.8747
3600 task requests	0.8857	0.8692	0.8773
4500 task requests	0.8762	0.8848	0.8805

still higher than the other two approaches. Therefore, in general, it can be observed that the proposed FLOM approach has certain performance advantages in tasks classification in big data. It is mainly because we choose the federated learning model to reduce the negative impact of data island on the accuracy of the model. Different from the traditional methods, which use hand-crafted feature learning, FLOM can obtain more accurate task request classification results by virtue of the powerful representation capability of deep neural network. On the other hand, we can carry out online update and incremental learning without retraining, while the general traditional methods still need to rely on other incremental learning algorithms other than the current model. Accurate task classification ensures the efficient processing of tons of task requests to a large extent, thus reducing resource costs.

6.3. Comparison in Load Balancing Effect. In this section, we analyze the load balancing effect of FLOM with the other task processing approaches. We compare their performance within 4-layer neural network since it possesses the potential to obtain relatively favorable results. And in the following experiments, we all employ this network structure by default. We employ the standard deviation value to measure the degree of load balancing in the experiments. The lower standard deviation value indicates that the load balancing effect of in the federated cloud center with edge computing is better at this time. As shown in Figures 6(a) and 6(b), in the two kinds of task request scales, the results have shown a trend of decreasing with the time changing. For the task scale of 1200, the standard deviation value of FLOM is always smaller than that of the other three approaches, while the results of Capacity Scheduler are always smaller than those of Fair Scheduler except for the time at 1750s. And their standard deviation values are both smaller than those of FIFO approach from 250s to 2750s. As for the scale of 3600 task requests, the proposed FLOM approach still maintains its advantages in terms of load balancing effect compared to the other three approach. Compared with Fair Scheduler, Capacity Scheduler just keeps its performance advantage before 750s, while its results become always higher than those of Fair Scheduler in the later stage. FIFO always has the higher standard deviation value compared to the other three approaches. The results of federated learning model significantly decrease the other three approaches by 5.98% at most.

From the above, we can observe that FLOM can achieve the desired load balancing of the whole cluster of the federated data center in the process of large-scale task processing in 4-layer neural network structure. It is mainly because that the federal learning model is used to realize the task and resource information sharing among cloud data center and edge computing centers, which makes the whole task allocation more reasonable. In addition, considering the overall load effect, the deep network model has been employed to achieve the deep feature learning of tasks requests and the task processing capability of each node, so as to realize the deep analysis. Tons of task requests can be reasonably assigned to avoid overload of some certain nodes and improve the overall resource utilization.

6.4. Performance under Stress Circumstances. In this section, we evaluate the capability of FLOM to assign task requests under the stress circumstances. In order to verify the effectiveness and robustness of FLOM in many application environments that often increase the demands due to large-scale task requests from users, we employ the stress tools to respectively inject two kinds of stress (CPU and bandwidth) to simulate the reduplicated task requirements into the model application. We still employ accuracy to measure the task processing capability of FLOM under the stress circumstances. It is clearly evident from Figure 7 that FLOM can achieve the promising results using the federated learning model. As for the reduplicated stress of CPU circumstance, Fair Scheduler and Capacity Scheduler obtain similar experimental results from the task scale of 600 to 4500. And both of them have achieved the expected results (0.801 and 0.811) in the end. FIFO has maintained a relatively low accuracy and did not exceed 0.8 from beginning to end. The accuracy of FIFO, Fair Scheduler, and Capacity Scheduler is always lower than that of FLOM. For the stress of bandwidth circumstance, we can observe that the bandwidth resource consumed by Capacity Scheduler is always more than that of Fair Scheduler since Capacity Scheduler generally does not assign network bandwidth resources for each subtask according to the global situation of system, which will lead to excessive resource consumption. The accuracy of FIFO is always lower than that of the other three approaches from start to finish, while FLOM still

RETRACTED

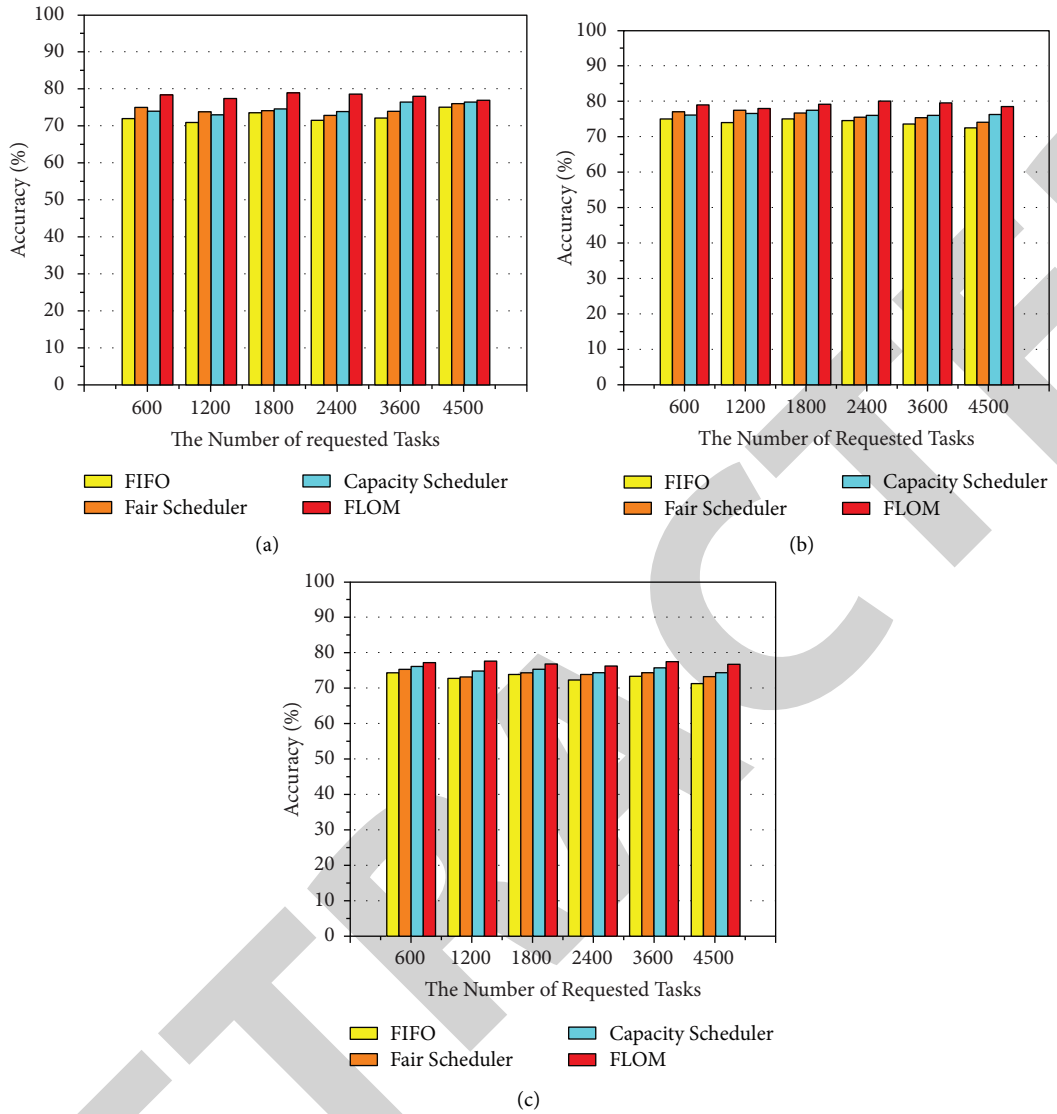

Figure 5: Comparison of FLOM with FIFO, Fair Scheduler, and Capacity Scheduler in task allocation effectiveness under different neural network structures. (a) 3-layer. (b) 4-layer. (c) 5-layer.

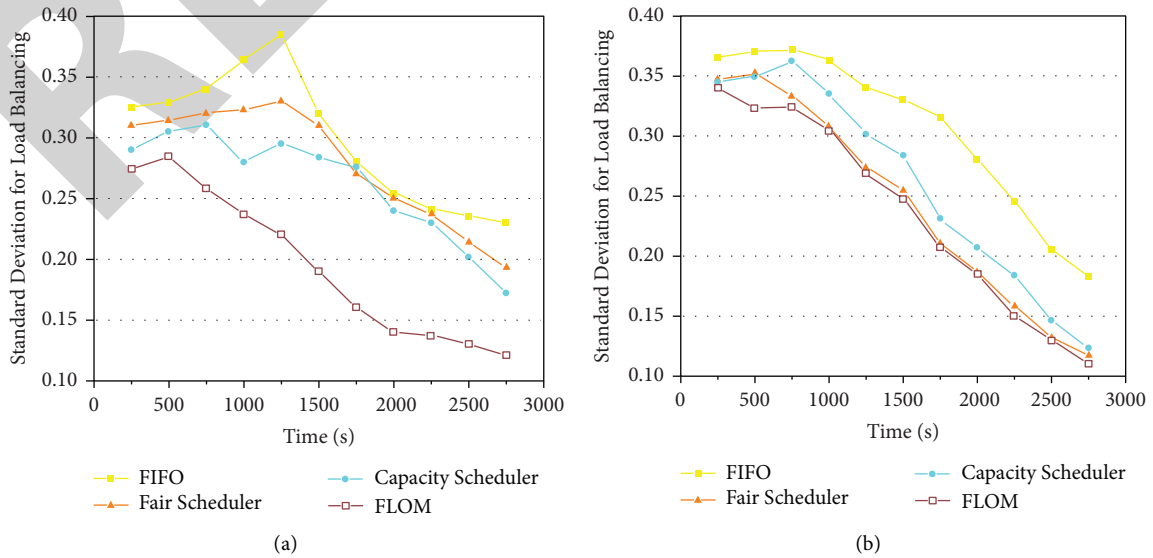

Figure 6: Comparison of FIFO, Fair Scheduler, Capacity Scheduler, and FLOM in load balancing with the task scale of (a) 1200 task requests and (b) 3600 task requests.

RETRACTED

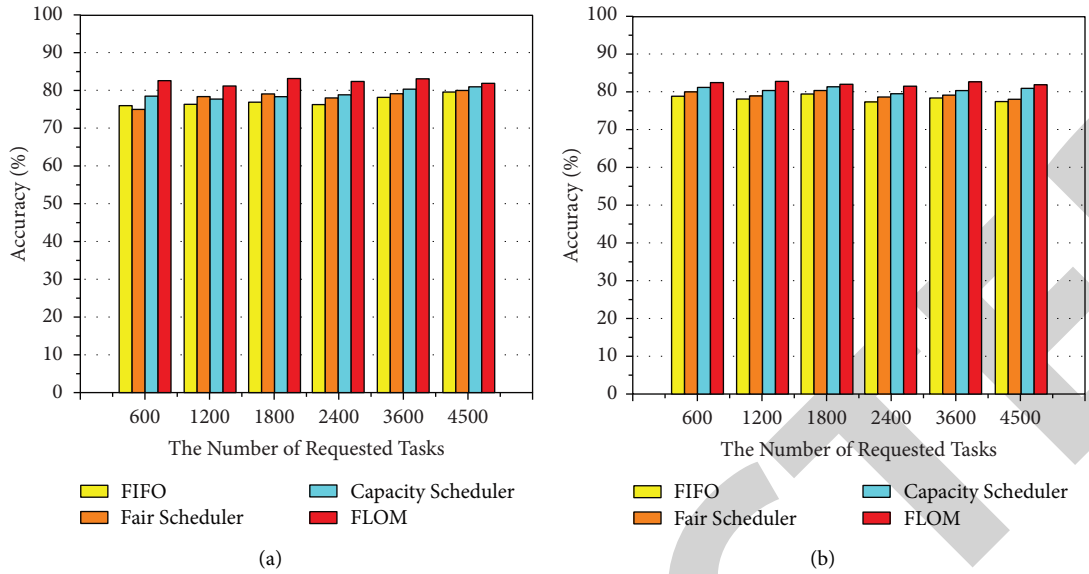

Figure 7: Comparison of FLOM with FIFO, Fair Scheduler and Capacity Scheduler in task allocation effectiveness under (a) CPU stress and (b) bandwidth stress.

keeps its superiority in task request classification under the bandwidth stress circumstance. This is mainly because we employ deep network model to achieve deep feature learning of various tasks and substrate resources and fully grasp the deep relationship in the resources and tasks matching. Thus, we can still maintain considerable task classification performance under stress circumstances and provide the guarantee for efficient processing of large-scale tasks.

6.5. Comparison in External Service Performance. In the last set of experiments, FLOM is verified by comparing the external performance of the whole system with the other three approaches. In this work, throughput is employed to measure the external service performance of the system since it is one of the specific embodiments of the comprehensive capability of the system in most of cloud computing environment. It is clearly evident from Figure 8 that the results of all approaches show a gradually decreasing trend. Specifically, for the scale of 1200 task requests, FIFO gets the decent results in the beginning stage but lower values than those of the other three approaches. Fair Scheduler and Capacity Scheduler get the similar results from start to finish, while the proposed FLOM approach gets the relatively lower results before 1250s and keeps its advantages from 1250s to 2750s.

For the scale of 2400 tasks, FLOM receives the relatively lower throughput before 1000s and gets the best results among four approaches from 1500s to 2750s. FIFO, Fair Scheduler, and Capacity Scheduler get expected results in the initial stage but lower values than those of FLOM in the latter stage. In the experiment for 4500 task requests, FLOM gets relatively lower results in the initial stage. But it exceeds the other three approaches in terms of throughput at 750s and keeps this advantage to the end. FIFO, Fair Scheduler, and Capacity Scheduler also receive expected results that are close to 2 req/s. In these three different scale experiments, FLOM has shown its performance advantages. It is mainly because FLOM fully considers the features of each task request and the substrate resource and employs the deep neural network for deep feature learning, which realizes the optimal mapping between task requests and resources. It exploits the limited resources to optimize external service performance and improves the capability of large-scale task processing in the edge environment.

In this work, our FLOM approach is a large-scale task processing approach for big data in the edge computing environment. This paper provides a detailed implementation process and experimental verification of this approach. It is suitable for the effective processing of multicategory tasks in the edge computing environment. Herein, we discuss its effectiveness in large-scale distributed tasks processing and its potential for future expansion.

FLOM with federal learning. In the traditional distributed environment, information islands always exist since each edge computing center needs to deal with different task requests. Information is relatively confidential among them, and information sharing needs to be realized at a certain cost. By using the idea of federated learning, we can realize the interaction of model parameters among the cloud data center and each edge computing center. It completes the iterative update of each submodel and cloud federated model without exposing their own private information, realizes the accurate classification, identification, and effective processing of large-scale tasks in big data environment by using unified model, and decreases unnecessary resource consumption.

FLOM with deep learning. The traditional classification approaches of cloud computing tasks tend to analyze the task request protocol and the file format related to

RETRACTED

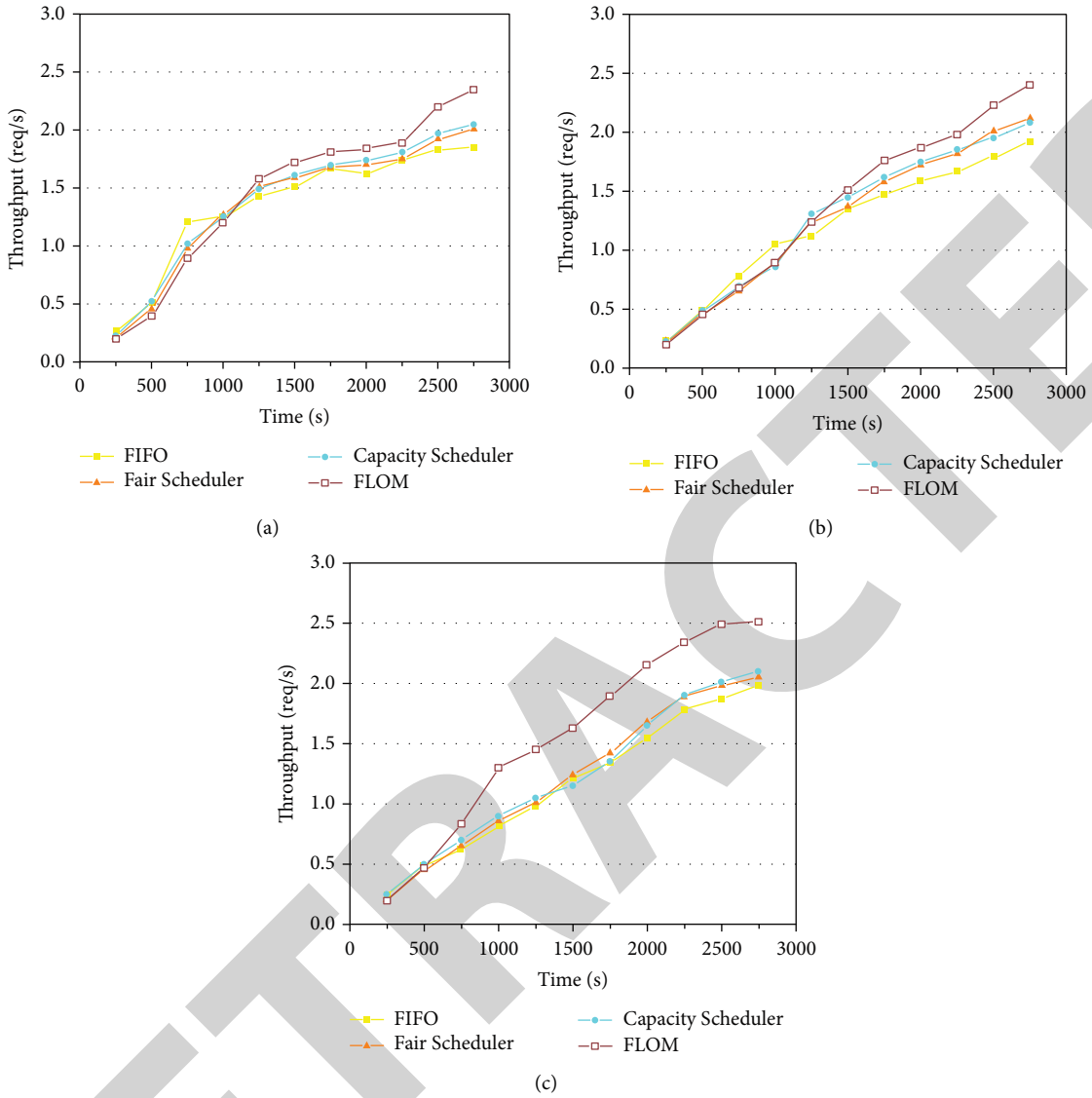

Figure 8: Comparison of FLOM with FIFO, Fair Scheduler, and Capacity Scheduler in external service with the task scale of (a) 1200 task requests, (b) 2400 task requests, and (c) 4500 task requests.

tasks from the characteristics of the task resource requirements. According to the demands of the task on CPU, network bandwidth, disk, and many other resources, the task requests are divided into CPU bound, communication bound, disk bound, etc. FLOM employs deep feature learning to analyze the collected task data and the features of the substrate network resources and forms a feature matrix composed of task request features and task processing node features in the edge cluster as the input of the deep neural network based federated model. The probability distribution of a certain task request assigned to each task processing node is taken as the output of the model to determine the optimal scheme of the current task allocation. The obtained scheme is considered from the perspective of global system benefit rather than just achieving local optimal after task classification. In addition, deep learning can be used for task classification in big data with the model updating through incremental learning. In the model application phase, it can avoid the excessive resource and time consumption caused by many traditional methods using a large number of iterative solutions.

FLOM with certain potential in big data analysis and processing. This work focuses on using the idea of federated learning to achieve large-scale tasks classification in cloud data center and edge computing centers to achieve efficient processing of big data. Herein, we constantly adjust number of layers of deep neural network (from 3 to 5) in the federated model to explore better experimental results. In the future research, we can continue to tune the number of layers of the deep neural network to explore and validate the performance of the federated model. In addition, in many real application scenarios, FLOM can be deployed in a larger scale edge computing environment to achieve task

RETRACTED

identification processing with more task types by tuning the network structure and parameter settings. We hope that, by proposing FLOM, federated learning will drive a new paradigm for big data processing in the future.

7. Conclusion

This work has studied the combination of federated learning and deep feature learning in the edge computing environment that supports the large-scale tasks processing for big data. Herein, (1) an optimized large-scale task processing methodology based on federated learning in the edge computing environment for big data is proposed; (2) a federated model for task processing based on the federated learning framework and deep feature learning is designed. We achieve the accurate task classification and overall load balancing by using the federated model trained through parameter updating among the cloud data center model and the edge computing models as well as the deep convolutional network with dropout; (3) the superiority of FLOM for large-scale tasks processing is evaluated in comparison with state-of-the-art approaches in the edge computing environment. Experimental results show that FLOM can accurately classify tasks, realize overall load balancing, and decrease consumption of task execution simultaneously.

In the future, it is necessary to adjust the number of layers in the deep neural network during exploring the optimal task allocation schemes since the selection of number of layers is an open problem. FLOM opens a new door for future research in big data processing. We plan to extend FLOM to the large-scale tasks processing in the larger scale experimental environment.

Data Availability

The simulation experiment data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant no. KJQN202103108), the Doctoral Initiation Project of Chongqing College of Electronic Engineering (120724), the Chongqing University Innovation Research Group Project (CXQT21031), the Research and Application of Industrial Internet Remote Sharing Training Technology (cstc2020jscx-msxmX0091), and the School Project of Chongqing College of Electronic Engineering (XJZK202104).

References

[1] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, 2019.

[2] Z. Luo, M. Li, L. Huang, X. Du, and M. Guizani, "Caching mechanism for mobile edge computing in V2I networks," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 8, 2019.

[3] Z. Chang, W. Guo, X. Guo, Z. Zhou, and T. Ristaniemi, "Incentive mechanism for edge computing-based blockchain," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7105–7114, 2020.

[4] D. Hu, D. Feng, Y. Xie, G. Xu, X. Gu, and D. Long, "Efficient provenance management via clustering and hybrid storage in big data environments," *IEEE Transactions on Big Data*, vol. 6, no. 4, pp. 792–803, 2019.

[5] X. Liu, Q. Zhu, S. Pramanik, C. Titus Brown, and G. Qian, "VA-store: a virtual approximate store approach to supporting repetitive big data in genome sequence analyses," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 3, pp. 602–616, 2020.

[6] M. M. Bersani, F. Marconi, D. A. Tamburri, A. Nodari, and P. Jamshidi, "Verifying big data topologies by-design: a semi-automated approach," *Journal of Big Data*, vol. 6, no. 1, 2019.

[7] W. Shu, K. Cai, and N. Xiong, "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing," *Future Generation Computer Systems*, vol. 124, pp. 12–20, 2021.

[8] W. Fang, X. Yao, X. Zhao, J. Yin, and N. Xiong, "A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms," *IEEE Transactions on Systems*, vol. 48, no. 4, pp. 522–534, 2016.

[9] Z. Liu, L. Lang, L. Li, Y. Zhao, and L. Shi, "Evolutionary game analysis on the recycling strategy of household medical device enterprises under government dynamic rewards and punishments," *Mathematical Biosciences and Engineering*, vol. 18, no. 5, 2021.

[10] T. Miller, "Explanation in artificial intelligence: insights from the social sciences," *Artificial Intelligence*, vol. 267, no. 2, 2019.

[11] T. Williams, D. Szafir, T. Chakraborti, and H. B. Amor, "Report on the first international workshop on virtual, augmented, and mixed reality for human-robot interaction," *AI Magazine*, vol. 39, no. 4, 2018.

[12] M. Hutson, "Artificial intelligence faces reproducibility crisis," *Science*, vol. 359, no. 726, p. 725, 2018.

[13] M. Adhikari and T. Amgoth, "An enhanced dynamic load balancing mechanism for task deployment in IaaS cloud," in *Proceedings of the IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, Kuala Lumpur, Malaysia, September 2019.

[14] R. Liu, S. H. Marakkalage, M. Padmal et al., "Collaborative SLAM based on WiFi fingerprint similarity and motion information," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1826–1840, 2020.

[15] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: concept and applications," *ACM Transactions on Intelligent Systemsand Technology*, vol. 10, no. 2, 2019.

[16] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: distributed machine learning for on-device intelligence," 2016, https://arxiv.org/pdf/1610.02527.

[17] G. A. Kaissis, M. R. Makowski, R. Daniel, and R. F. Braren, "Secure, privacy-preserving and federated machine learning

RETRACTED

in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[18] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[19] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, 2020.

[20] M. R. Minar and J. Naher, "Recent advances in deep learning: an overview," 2018, https://arxiv.org/abs/1807.08169.

[21] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015.

[22] T. Zhou, W. Wu, L. Peng et al., "Evaluation of urban bus service reliability on variable time horizons using a hybrid deep learning method," *Reliability Engineering and System Safety*, vol. 217, Article ID 108090, 2020.

[23] Y. Jing, H. Hu, S. Guo, X. Wang, and F. Chen, "Short-term prediction of urban rail transit passenger flow in external passenger Transport hub based on LSTM-LGB-DRS," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, 2021.

[24] L. Dong, M. N. Satpute, W. Wu, and D. Z. Du, "Two-phase multi-document summarization through content attention-based subtopic detection," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 6, pp. 1379–1392, 2021.

[25] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3, 2014.

[26] P. Li, Z. Chen, L. T. Yang, Q. Zhang, and M. Jamal Deen, "Deep convolutional computation model for feature learning on big data in Internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 790–798, 2017.

[27] X. Yuan, Y. Gu, Y. Wang, C. Yang, and W. Gui, "A deep supervised learning framework for data-driven soft sensor modeling of industrial processes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4737–4746, 2019.

[28] C. Lin, J. Lu, G. Wang, and J. Zhou, "Graininess-aware deep feature learning for pedestrian detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 3820–3834, 2020.

[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, 2016.

[30] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Transactions on Vehicular Technology*, vol. 99, p. 1, 2018.

[31] S. Mukherjee and J. Lee, "Edge computing-enabled cell-free massive MIMO systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, 2020.

[32] Y. Tang, K. Guo, J. Ma, Y. Shen, and T. Chi, "A smart caching mechanism for mobile multimedia in information centric networking with edge computing," *Future Generation Computer Systems*, vol. 91, no. 2, 2019.

[33] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao, "FedHealth: a federated transfer learning framework for wearable healthcare," *Intelligent Systems*, vol. 35, no. 4, pp. 83–93, 2020.

[34] Y. Liu, T. Chen, and Q. Yang, "Secure federated transfer learning," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, 2018.

[35] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)*Lake Buena Vista, FL, USA, April 2004.

[36] Y. Xue, H. Zhang, and H. Ma, "Performance evaluation of image and video cloud services," in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Exeter, UK, June 2019.

[37] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, 2004.

[38] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and F. Bu, "An adaptive dropout deep computation model for industrial IoT big data learning with crowdsourcing to cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2330–2337, 2018.

[39] P. Bala and S. Raju, "Performance improvement of multi-core architecture using Whetstone application in Linux," *International Journal of Computer Science and Network Security*, vol. 13, no. 10, 2013.

[40] R. Han, C. H. Liu, Z. Zong et al., "Workload-adaptive configuration tuning for hierarchical cloud schedulers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2879–2895, 2019.

[41] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[42] D. Cotroneo, R. Natella, and S. Rosiello, "Overload control for virtual network functions under CPU contention," *Future Generation Computer Systems*, vol. 99, pp. 164–176, 2019.

[43] D. Lunga, J. Gerrand, L. Yang, C. Layton, and R. Stewart, "Apache Spark accelerated deep learning inference for large scale satellite image analytics," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 271–283, 2020.

RETRACTED