Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9784720, 1 page
https://doi.org/10.1155/2023/9784720

Retraction

Retracted: QoC-Driven MEC Transfer System Framework in Wireless Networks

Wireless Communications and Mobile Computing

Received 8 August 2023; Accepted 8 August 2023; Published 9 August 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1].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] P. Yu, H. Zhao, M. Hu et al., "QoC-Driven MEC Transfer System Framework in Wireless Networks," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 7863972, 14 pages, 2022.

Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7863972, 14 pages
https://doi.org/10.1155/2022/7863972

Research Article

QoC-Driven MEC Transfer System Framework in Wireless Networks

Ping Yu,[1,2] Hongwei Zhao,[1] Ming Hu,[2] Hui Yan,[3] Xiaozhong Geng,[2] Hanlin Chen,[2] and Dejin Chu[2]

[1]*College of Computer Science and Technology, Jilin University, Jilin 130000, China*
[2]*Changchun Institute of Technology, Jilin 130012, China*
[3]*Suqian University, Suqian 223800, China*

Correspondence should be addressed to Hui Yan; yanhui@ccit.edu.cn

Received 23 June 2022; Revised 14 July 2022; Accepted 20 July 2022; Published 19 August 2022

Academic Editor: Hangjun Che

In this paper, we propose a Heterogeneous *MEC* System Framework based on Transfer Learning (*HMECSF-TL*), which uses convolutional neural network (*CNN*) to process few training samples. In view of the time-varying network environment and the limited end devices resources, the *HMECSF-TL* framework uses transfer learning (*TL*) technology to optimize the *CNN* model and jointly optimizes the allocation of computing resources and communication resources, which is beneficial to achieve the dual goals of extending the use time of end devices and improving the speed and the accuracy of image classification. We first introduce the Quality of Content (*QoC*)-driven *MEC* transfer system architecture of *cloud-edge-end*. The cloud server uses the existing image dataset to train the general neural network model in advance and transfer the general model to the edge servers, and then the edge servers deploy the local models to the end devices to form the personalized models. Then, considering the time-varying situation of the network environment, in order to get the updated model faster and better, we present the process of collaborative optimization of model between the edge sever and multiple end devices, using an edge server as an example. Considering the limited resources of the end devices, we propose a joint optimization of energy and latency with the goal of minimizing offloading cost, in order to rapidly improve the speed and the accuracy of image classification with few training samples under the premise of rational resource allocation and verify the performance of the framework experimentally. Simulation results show that the proposed *HMECSF-TL* framework outperforms the benchmark strategy without *TL* in terms of reducing the model training time and improving the image classification accuracy, as well as reducing the offloading cost.

1. Introduction

The rapid development of *IoT* technology and the emergence of massive data have led to a shift in the data processing model from a centralized cloud data center to distributed network edge sides, resulting in the emergence of multiaccess edge computing (*MEC*). Distributed *MEC* servers provide computing resources and *IT* services with low-latency, low energy consumption, and high bandwidth and security. *MEC* technology has become an important technology for *5G*. At the same time, with the popularization of *IoT* devices and the intelligentization of mobile terminals, more and more mobile applications have become resource-scarce and latency-sensitive. Only with the continuous enhancement of the data processing capabilities of the end intelligent devices can they better satisfy the needs. *MEC* technology makes better use of data through distributed model training and intelligent knowledge inference, enabling deeper integration of *IoT* and *AI* and further realizing the intelligent connection of all things.

With the help of *MEC*, end devices can offload computing tasks from computing-intensive applications such as image classification to nearby edge servers to save energy and improve computing power [1]. *CNN* is a key technology to achieve image classification, and it is an influential deep learning framework. The training process of *CNN* models based on big data introduces a relatively high computational complexity, and *MEC* is beneficial in reducing the computational burden of the devices. In addition, in practical applications, the training data and the data in the target domain may not be in the same feature space or may follow different data distributions. *TL* is suitable for solving the above problems, and *TL* can well reduce the convergence time of models in the target domain [2]. Although *TL* has been shown to improve data-starved situations, the huge demand for personalized samples and unstable transmission latency limit the performance of real-time applications. Although *MEC*-assisted deep learning can provide responsive services and sufficient computing resources, related research is still in its infancy [3]. Especially for computing-intensive real-time applications requiring ultralow latency, it is important to explore the application of *TL* in *MEC* systems. This paper is based on this background to carry out research work.

In order to alleviate the contradiction between the limited resources of the end devices and efficient image classification, we explore the relationship between energy consumption, latency, and *QoC*; then we consider offline training of the general model, online collaborative optimization of local models, and fine-tuning of personalized models; and also consider the requirements of energy consumption and latency; and finally, we propose a *cloud-side-end MEC* transfer system framework. We further study the training and inference work of the deep learning model by implementing model transfer through *TL* for energy-constrained end devices under the condition of limited wireless bandwidth. The advantages of the framework and the main contributions of this paper are summarized as follows:

(1) Taking full advantage of the abundant resources on the *cloud-edge* and the latest data provided by end devices, we propose *HMECSF-TL* based on the *cloud-edge-end* architecture. The proposed framework completes the model transfer through *TL* and localizes and personalizes the general model of deep neural network in order to achieve the effect of reducing model training time

(2) The framework adopts an online collaborative way to train the local model on the edge server, which can not only respond quickly on the edge server, but also improve the accuracy of image classification. Specifically, the end devices that satisfy the optimal resource allocation strategy on the edge server offload few training samples to participate in the collaborative training of the local model on the edge server. After training to obtain the optimized model, the personalized model of the end devices are fine-tuned through *TL* to update it to the optimized model, and then the inference work of the end devices that do not participate in the collaborative work is completed

(3) The *HMECSF-TL* framework considers the balance between improving image classification accuracy and rational allocation of resources. Specifically, considering the limited energy of end devices and limited wireless bandwidth in wireless networks, we propose a joint optimization energy and latency problem to implement resource management and find optimal strategy for bandwidth allocation and task offloading to minimize the weighted sum of costs

(4) Simulation experiments show that the proposed *HMECSF-TL* framework can achieve better performance, which is embodied in the advantages of cost-saving and the model performance related to the image classification accuracy and the training time

The rest of the paper is organized as follows. *Part 2* describes the related work. *Part 3* describes the proposed framework and the system model in detail. *Part 4* presents the problem formulation of optimizing resources. Part *5* presents the problem of image classification. *Part 6* gives the simulation results and the conclusions drawn. *Part 7* concludes.

2. Related Work

MEC is an evolution of cloud computing that sinks computing power to the edge of the network. In [4], *MEC* enables end devices to offload intensive computing task to nearby edge server to save energy consumption and improve computing power. The *edge-end* computing framework can provide better low-latency and stronger computing power, which is suitable for solving various applications with low-latency and computing-intensive. For example, image classification using *CNN* is one of the common applications, which has a large impact on the performance improvement in terms of classification accuracy. In [5], an image classification using *CNN* is provided by mobile devices for various image and video analysis applications. However, due to the limited computing resources of end devices, the devices are insufficient to provide accurate and fast image classification capabilities. In [6], an image classification method based on the *MEC* architecture was proposed to solve the problem of limited computing resources of devices. *5G* provides ultra-low-latency, high-reliability connections for the end devices. In [7], Kim et al. leveraged the low transmission latency of *5G* by combining *5G* and *MEC* to bring the computing power closer to the end devices and reduce the applications latency. Thus, the combination of *5G* and *MEC* makes it excellent for various image and video analysis applications.

It is worth noting that most machine learning algorithms usually assume that the training data and test data lie in the same feature space and have the same distribution [8]. However, in [9], it is proposed that the input space may differ between the source tasks and the target tasks, the situation that is more realistic. And *TL* can be a good solution to the problems such as different input spaces. The deep transfer learning method, which uses deep learning technology to build transfer learning model, has become an important method to solve the problems of deep learning data dependency and insufficient training data. The deep learning model can greatly reduce the interference of human experience and improve accuracy under certain conditions [10]. In [11], a deep transfer network model based on a joint distribution adaptation method was proposed. In [12], a sparse encoder was combined with a deep transfer learning method to predict device lifetime. In [13], the use of deep *GANs* was proposed to solve the domain adaption problem. In this paper, *TL* is used in a *cloud-side-end* computing framework to reduce the model training latency and further improve the performance of the framework.

3. Heterogeneous MEC System Framework Based on Transfer Learning (*HMECSF-TL*)

3.1. System Architecture. Although the image processing work such as image classification based on deep learning has high recognition accuracy, it usually requires powerful computing resources to achieve the desired results, such as using cloud computing to process massive data. It is worth noting that cloud computing brings the problem of excessive communication latency and the risk of leaking user privacy. In recent years, with the rapid development of *MEC*, *AI*, *5G*, and other technologies, edge intelligence has emerged as the times require, communication latency and privacy risks can be well resolved, and network functions have shifted from interconnected devices to interconnected intelligence, but there are still some problems to be solved. Using the *cloud-edge-end* architecture to deploy the general model to edge servers and end devices can achieve edge intelligence, and latency requirement and privacy protection are guaranteed to a certain extent. However, due to the limited data collected by various end devices, the general model established by the deep learning algorithm on the cloud server usually has low accuracy in processing few samples, resulting in poor generalization of the model. Therefore, it is necessary to optimize the model by combining real-time data in order to achieve the expected effect. We perform model transfer on *cloud-side-end* through *TL* to improve the model training speed. The accuracy of model can be quickly improved by adopting two stages: the *edge-end* online collaborative optimization model method and the *cloud-edge* offline optimization model method. Considering the abundant resources of cloud server and edge servers but limited resources of end devices, this paper focuses on the *edge-end* online collaborative optimization model stage. The system model of our proposed *HMECSF-TL* framework is shown in Figure 1.

3.1.1. The Description of the HMECSF-TL Framework. This system includes the cloud server, the edge servers, and the end devices. Through *TL*, we transfer the general model on the cloud to the edge servers as the local models and then transfer the local model to the end devices as the personalized models [3]. It is worth noting that each time the upper model is optimized, the lower models are then updated by fine-tuning actions. We use massive data to train the general model on the cloud and use the data offloaded by multiple end devices to collaboratively train the local model on the edge server. The end devices that have not participated in the collaborative training use the personalized model to complete the image inference on the local few samples after fine-tuning. The general flow of the system optimization model is further described next.

(a) Each end device collects few samples, and then some of the end devices that satisfy the allocation strategy of jointly optimizing energy consumption and latency offload few samples to the edge server, and then the *edge-end* online cooperation completes the training of the local model

(b) The edge server uses the trained local model to update the personalized models of end devices through fine-tuning operations, and the devices that have not participated in the collaborative training of the local model use the updated personalized model to continue to complete the inference work

(c) Each edge server asynchronously offloads the dataset formed by the data offload by the end devices to the cloud server at regular intervals in order to optimize the general model. The optimized general model on the cloud also updates the local model of each edge server through fine-tuning operation, in order to improve the performance of the local model faster

3.1.2. Online Collaborative Training Stage of Edge-End. In this paper, we focus on the *edge-end* collaborative training stage which is closely related to the end devices, and the model optimization process is shown in Figure 2.

The optimization process includes the following four steps [14].

(a) Offload resource parameters to get the offload strategy and bandwidth allocation. The edge server calculates the optimal offload strategy and bandwidth allocation according to some key parameters such as the required computing resources of the processing task, preprocessed data size, transmission power, and channel conditions and then notifies the end devices

(b) Few samples offload from the end devices. The end devices that satisfy the offload strategy use the allocated bandwidth to offload tasks to the edge server

(c) Optimize the local model. The local model is trained on the edge server using few samples offloaded from multiple end devices at a small learning rate until convergence to further optimize the local model

(d) Broadcast the optimized local model. The edge server broadcasts the well-trained local model to the end devices, so that each device can independently perform inference work such as image classification using the updated personalized model and those end devices that do not participate in the collaborative training continue to complete the inference work

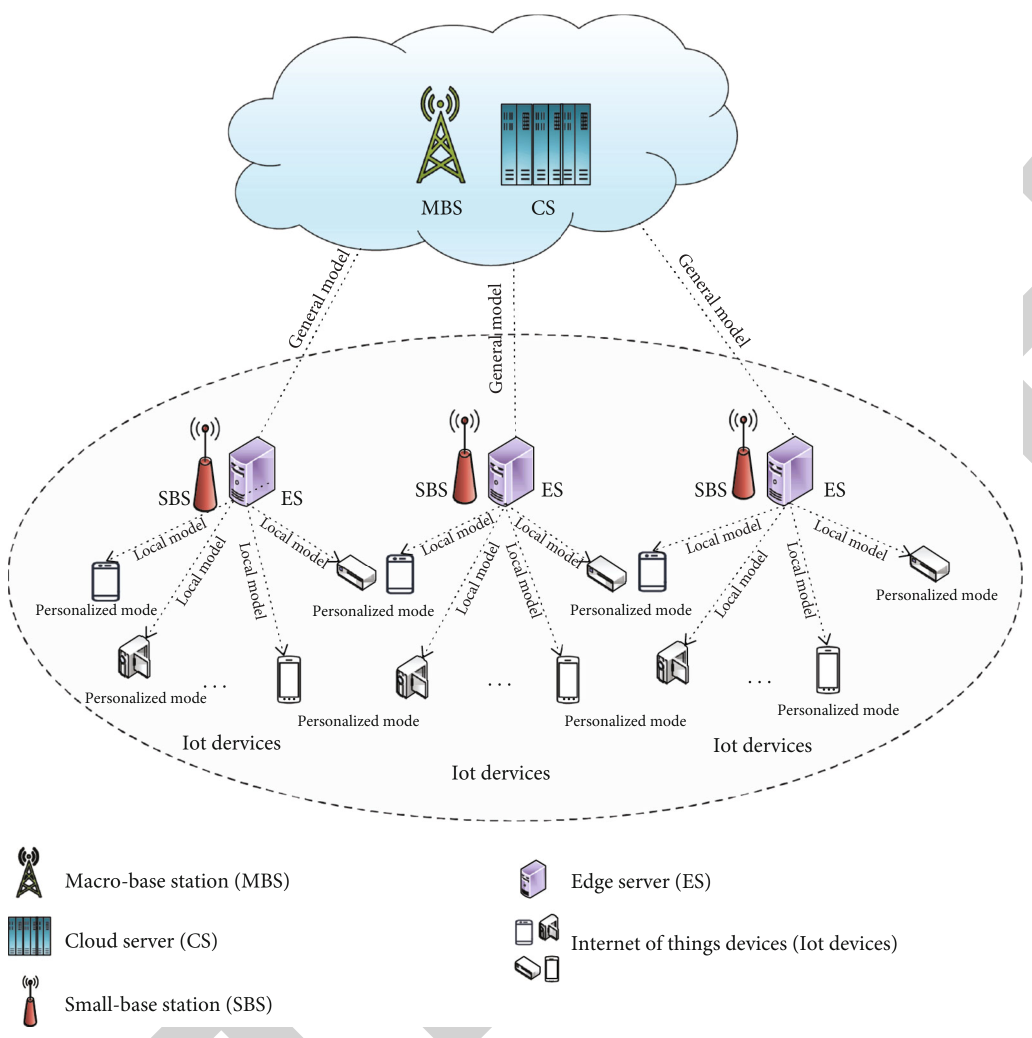


Figure 1: System model.

At this stage, because each device may have unique training samples, the edge server usually tends to include more samples from the devices. However, considering that the end devices have limited energy and limited wireless bandwidth, only a part of the devices can actually offload their few samples to the edge server. Therefore, we consider which devices to offload and how to allocate the appropriate wireless bandwidth.

4. Problem Formulation of Optimizing Resources

The *HMECSF-TL* framework improves the model training speed through *TL* and uses collaborative optimization to improve the model accuracy. However, considering the limited energy of the end devices and the limited wireless bandwidth, only some of the end devices can actually offload tasks to participate in the collaborative training local model. So we focus on the *edge-end* collaborative training model phase and obtain the optimal strategy of bandwidth allocation and task unloading by minimizing the weighted sum of costs and then perform *edge-end* collaborative training to complete the image classification work. Minimizing the weighted sum of costs is a problem of jointly optimizing energy consumption and latency, which is described further below.

We take an edge server as an example to focus on the *edge-end* collaborative training model stage. There are M terminal devices within the coverage of the edge server, and the edge server has sufficient computing resources and storage resources, and each end device contains embedded sensor and embedded computing chip, and each end device is involved in only one task in the same time slot.

The task for the m *-th* device is represented as

$$U_m = (F_m, D_m, T^{\text{req}}), m \in M, \tag{1}$$

where F_m indicates the computing resources required by the task, D_m indicates the size of the data to be processed, and T^{req} is the latency constraint.

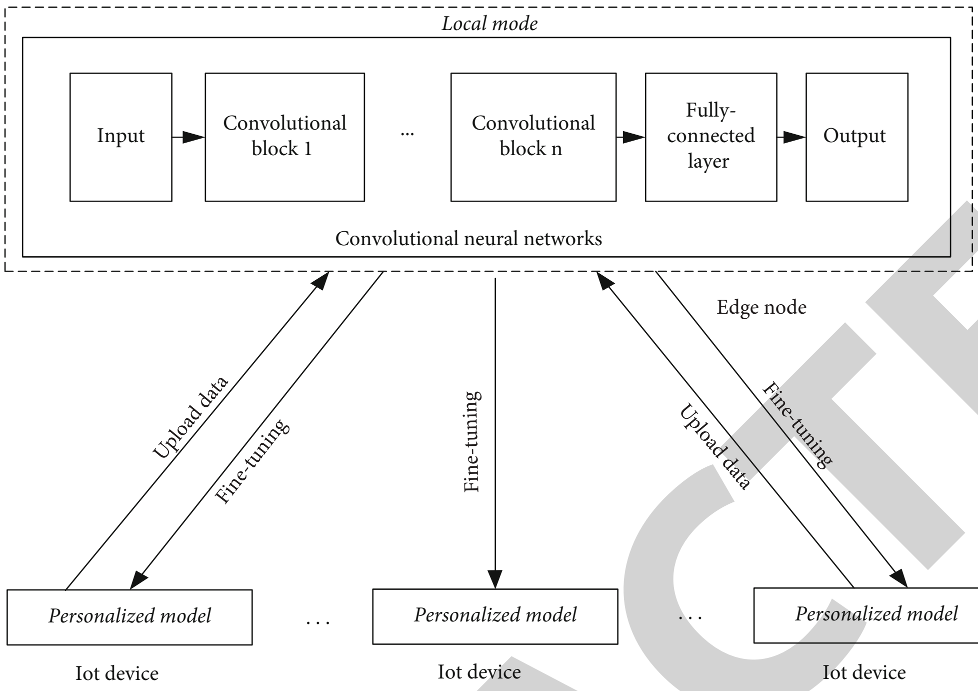


Figure 2: Model online optimization.

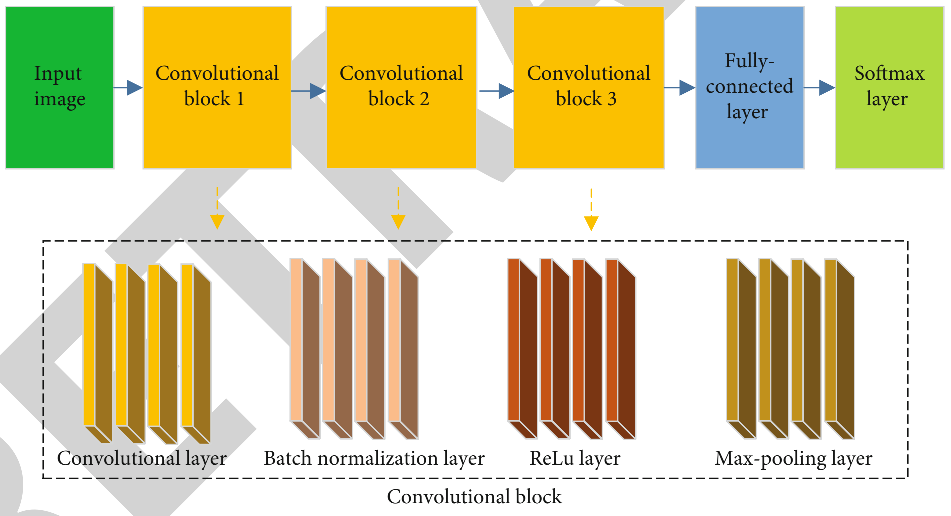


Figure 3: Illustrations of convolutional neural network

Whether the m -*th* device offloads the task to the edge server can be indicated by the variable a_m,

$$a_m = \{0, 1\}, m \in M. \tag{2}$$

If the task is processed on the device side, then $a_m = 0$. We use f_m^L to represent the computing execution capacity of the m -*th* device for processing the task U_m. $F_{m,\max}^L$ represents the maximum computing resource of the m -*th* device, which should satisfy the following constraint.

$$f_m^L \le F_{m,\max}^L, m \in M. \tag{3}$$

If the m -*th* device offloads the task U_m to the edge server, then $a_m = 1$. Here we assume that the device can only select at most one server for offloading and that it performs the full offload operation.

We use f_m^E to represent the computing execution capacity assigned to the m -*th* device by the edge server, which is used to process task U_m. $F_{\max}$ represents the maximum computing resources of the edge server, which should satisfy the following constraint.

$$\sum_{m=1}^{M} a_m f_m^E \le F_{\max}, m \in M. \tag{4}$$

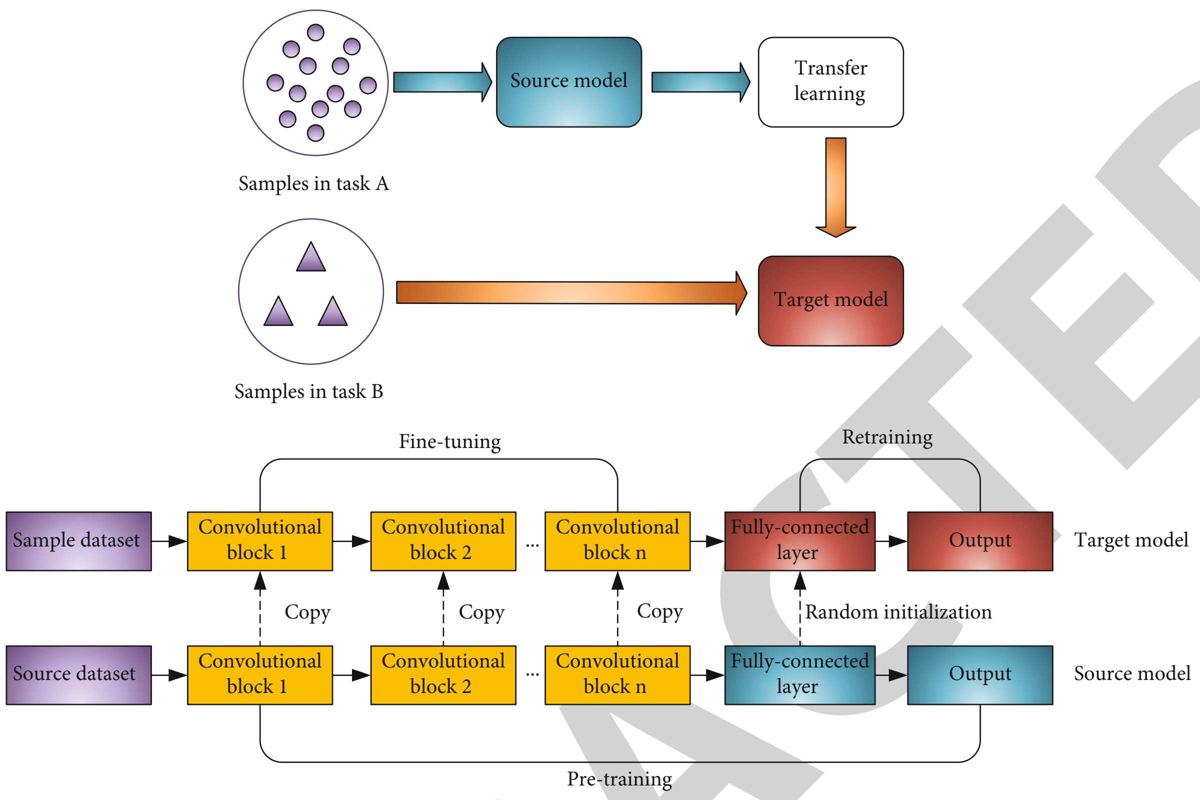


Figure 4: Illustrations of transfer learning.

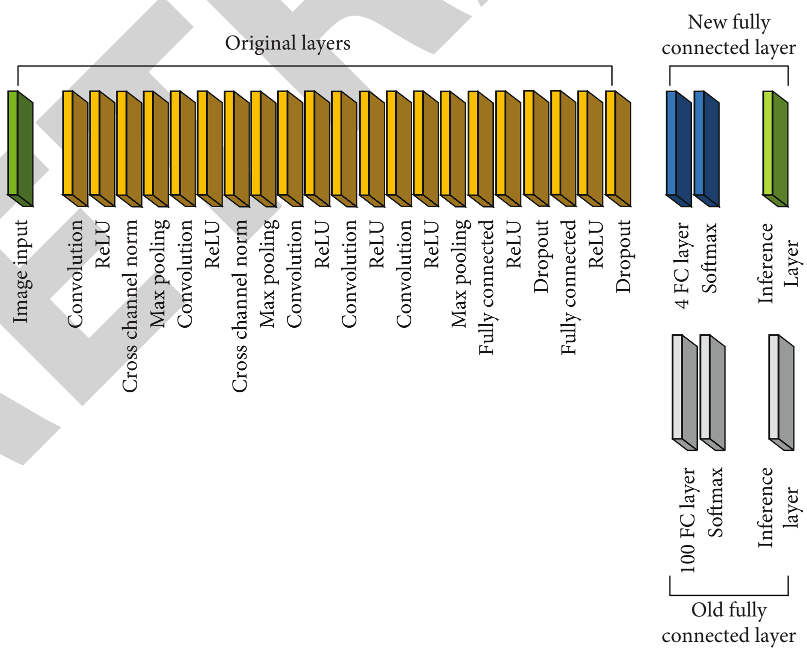


Figure 5: Illustrations of Transferred AlexNet CNN

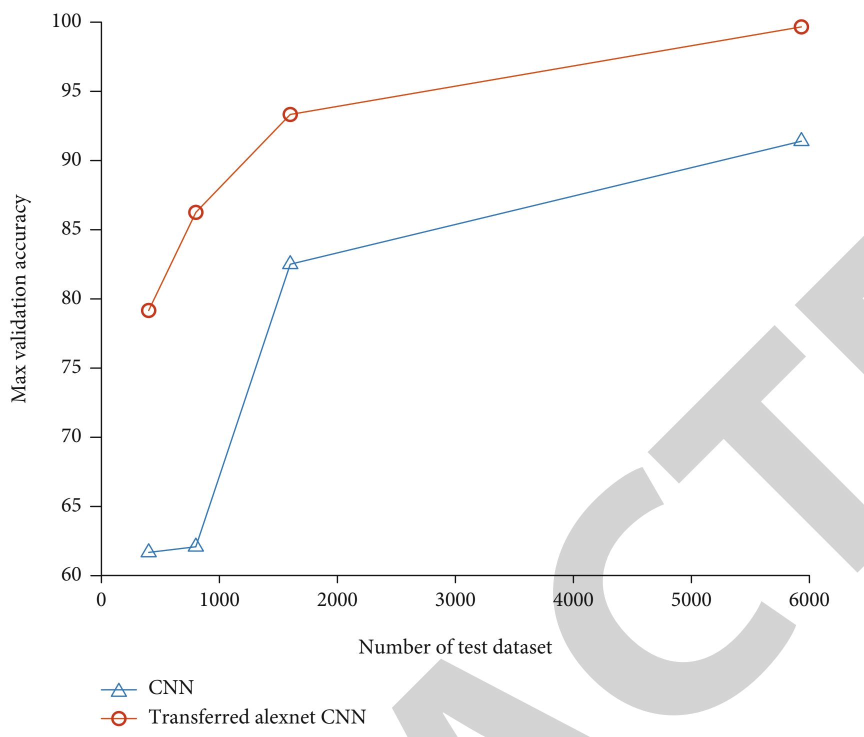


Figure 6: Max validation accuracy.

4.1. Latency Model. If the m *-th* device D_m processes the task U_m on the device side, then the latency only involves the calculated latency T_m^L which is represented as

$$T_m^L = \frac{F_m}{f_m^L}, m \in M, \tag{5}$$

and there is

$$T_m^L \leq T^{\text{req}}, m \in M. \tag{6}$$

The m *-th* device offloads the task U_m to the edge server and then participates in the online collaborative training model. So the latency T_m consists of two components, the transmission latency T_m^{tr} and the computing latency T_m^{com}; there are

$$\begin{gathered} T_m^{tr} = \frac{D_m}{r_m}, m \in M, \\ T_m^{\text{com}} = \frac{F_m}{f_m^E}, m \in M, \\ T_m = T_m^{tr} + T_m^{\text{com}} = \frac{D_m}{r_m} + \frac{F_m}{f_m^E}, m \in M, \end{gathered} \tag{7}$$

and there is

$$T_m \leq T^{\text{req}}, m \in M. \tag{8}$$

The transmitted data rate r_m for the m *-th* device offloaded the task U_m is represented as

$$r_m = w_m B \log_2\left(1 + \frac{P_m^{tr} h_m}{\sigma^2}\right), m \in M, \tag{9}$$

where B represents the channel bandwidth, P_m^{tr} represents the transmission power of the m *-th* device, h_m represents the channel gain, and σ^2 represents the noise power. w_m represents the proportion of subcarriers assigned to the m *-th* device by the edge server [1]. J represents the set of devices that perform task offloading.

$$\begin{gathered} W = [w_1, \cdots, w_m, \cdots w_J], m \in J, \\ J \leq M, \end{gathered} \tag{10}$$

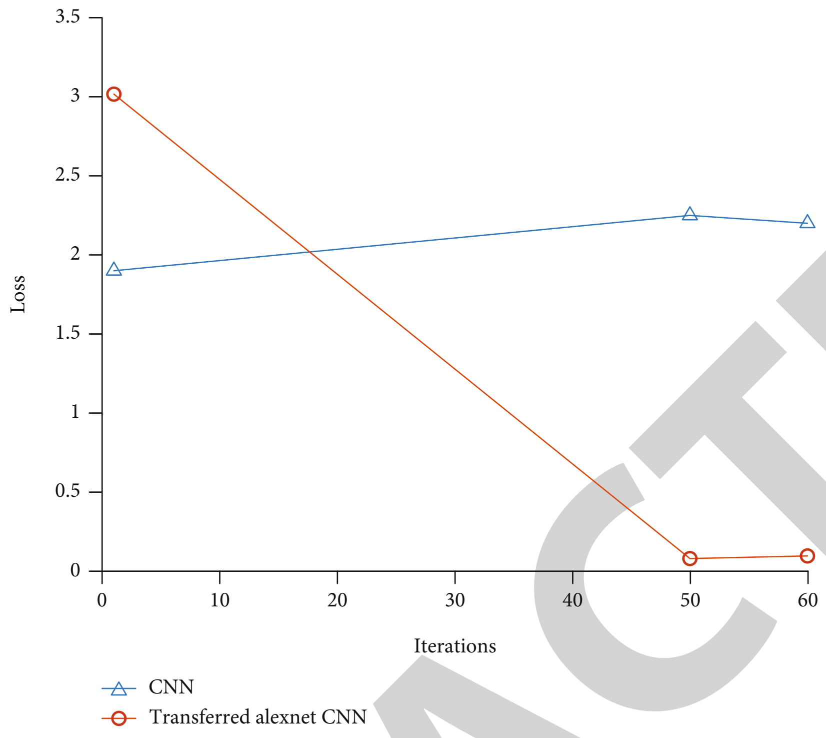
where the allocated subcarrier ratio takes a real number in the range of 0 to 1.

$$w_m \in [0, 1], \tag{11}$$

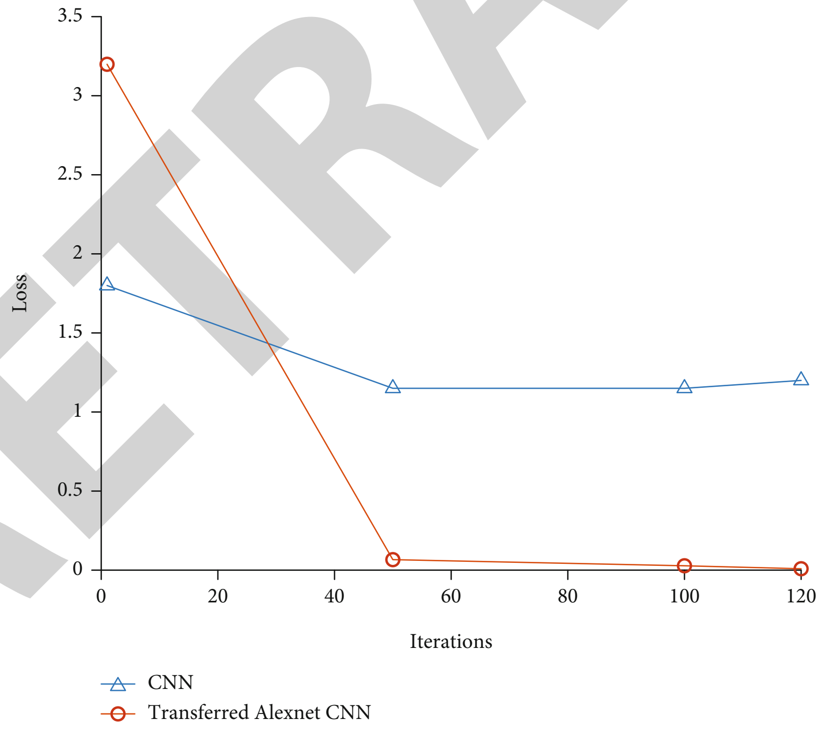
and in which the sum of the subcarriers ratios allocated by the edge server to the end devices participating in the offloading task cannot exceed 1.

$$\sum_{m=1}^{J} w_m \leq 1. \tag{12}$$

In this framework, considering the *edge-end* online collaborative training model time is affected by the offloading latency of each end device, the slowest end device will limit

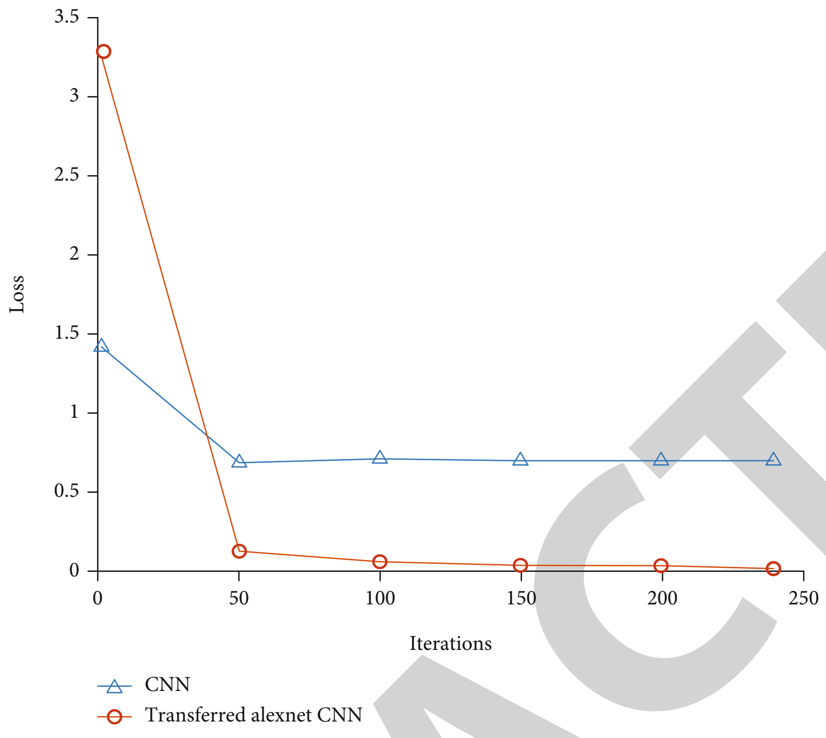


(a)

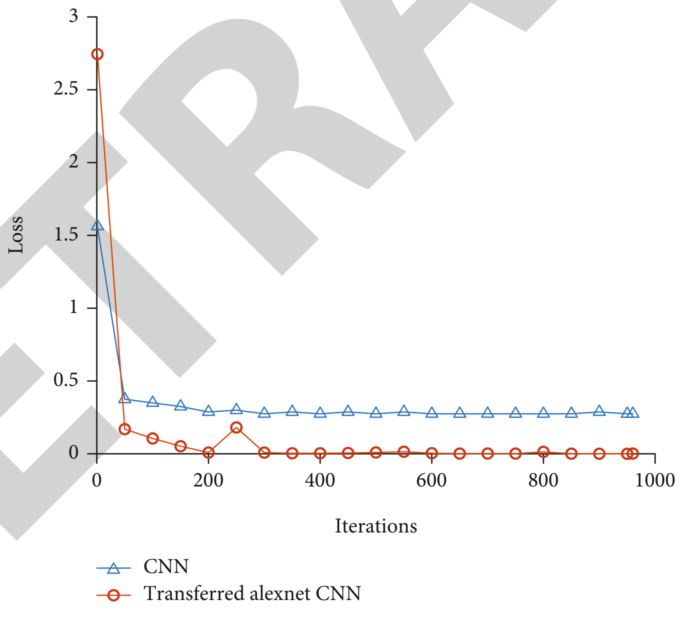


(b)

FIGURE 7: Continued.



(c)



(d)

Figure 7: Loss.

the total latency, leading to the "dropout effect" problem [14]. To reduce the latency, we can allocate more subcarriers to the end device that is originally slow to reduce their transmission latency. For better online collaborative training, it is better if the transmission latency of each device is as close as possible. The relevant constraints are as follows.

$$\min\max\left\{T_m^{tr} \mid m \in J\right\}. \quad (13)$$

Therefore, the final transmission latency $T(w)$ of the end devices participating in the *edge-end* collaborative training should take the maximum value of the transmission latency of the offloaded devices,

$$T(w) = \max_{m\in J}\left\{T_{\mathrm{m}}^{tr}\right\}. \quad (14)$$

Then, the latency T'_{mn} of the m -*th* device participating in

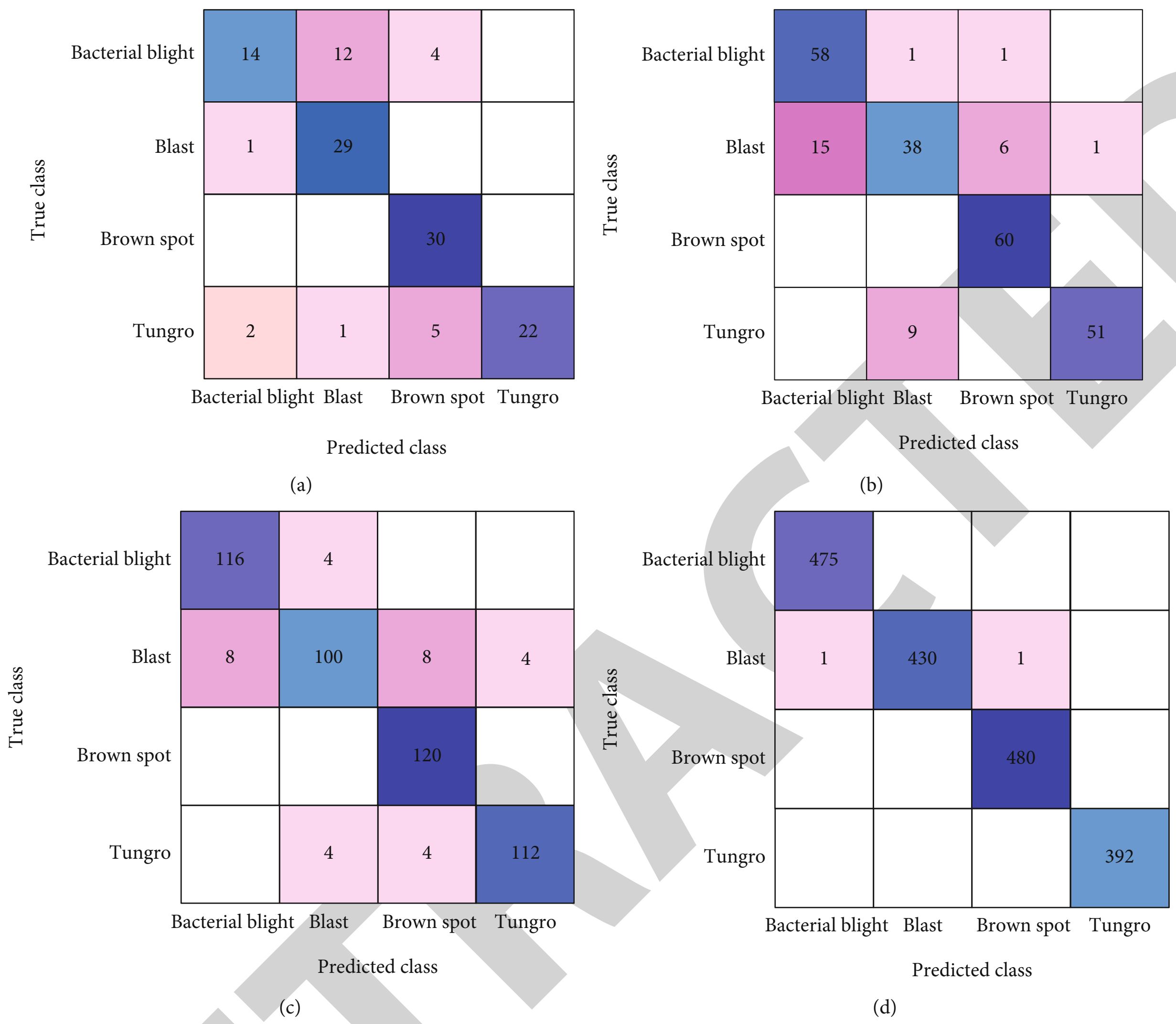


FIGURE 8: Confusion matrix of Transferred AlexNet *CNN*.

the *edge-end* online collaborative training to complete the task U_m processing after the correction is represented as

$$T'_m = T(w) + T_m^{\text{com}}, m \in M. \quad (15)$$

The total latency is

$$T = \sum_{m=1}^{M} \left((1 - a_m) T_m^L + a_m T'_m \right), m \in M. \quad (16)$$

4.2. Energy Model. If the m *-th* device processes the task U_m on the end device, then the energy consumption E_m^L of the device is represented as

$$E_m^L = k_m \left(f_m^L \right)^{v_m}, m \in M, \quad (17)$$

where k_m represents the effective switched capacitance and v_m represents a positive constant [1].

The m *-th* device offloads the task to the edge server and then participates in the *edge-end* online collaborative optimization training. The energy consumption of the m *-th* device is mainly related to the transmission energy consumption, there is that,

$$E_m = p_m^{tr} \frac{D_m}{r_m}, m \in M. \quad (18)$$

The total energy consumption of the device is

$$E = \sum_{m=1}^{M} \left((1 - a_m) E_m^L + a_m E_m \right), m \in M. \quad (19)$$

4.3. Problem Formulation. Since each edge server allocates resources individually, here we only analyze the online collaborative training model of a certain edge server. Specifically, the resource allocation is mainly to determine the offloading strategy and bandwidth allocation, so we describe the offloading problem and bandwidth allocation problem as

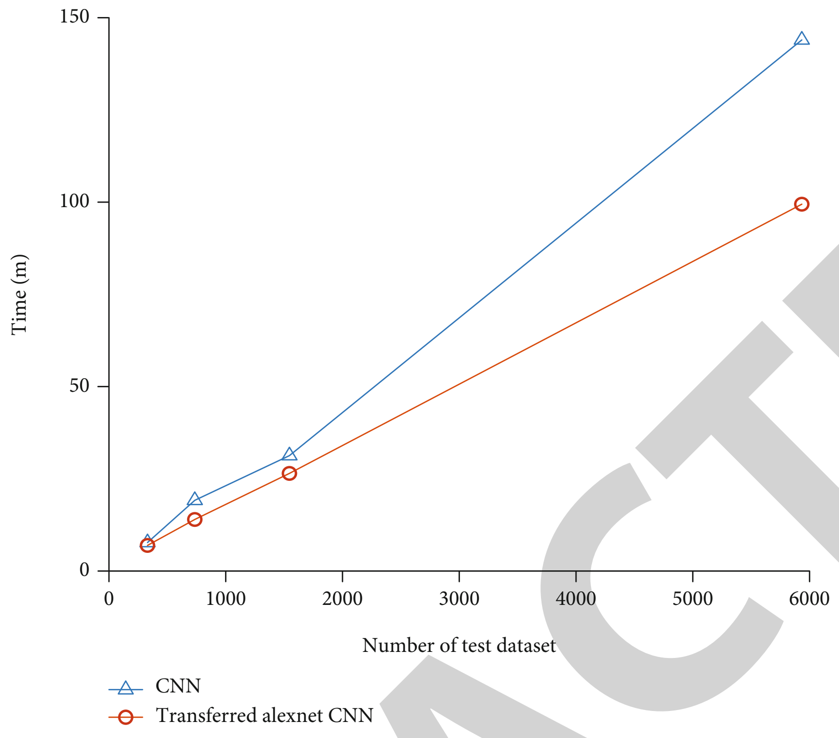


Figure 9: Processing time.

a problem of minimizing the sum of cost weights. For a given offload strategy vector a_m and uplink bandwidth allocation vector w_m, we define the weighted sum of costs during *edge-end* collaborative training as

$$O = \alpha E + (1 - \alpha) T. \tag{20}$$

In which,

$$\alpha \in [0, 1], \tag{21}$$

where α represents the weighting factor, and we can adjust α to achieve the focus of resource allocation. That is, if the task is latency-sensitive, then increase the value of $(1 - \alpha)$, and if the focus is more on device energy consumption issues, then increase the value of α.

Assuming that $A = \{a_1, \cdots, a_m, \cdots, a_J\}$ represents the offloading strategy vector, we describe the problem as a problem of minimizing the sum of cost weights [14–17], in which

$$(P1): \min_{\{A,W\}} O, \tag{22}$$

$$\text{s.t.C1} : a_m = \{0, 1\}, \tag{23a}$$

$$\text{C2} : J \le M, \tag{23b}$$

$$\text{C3} : \sum_{m=1}^{J} w_m \le 1, \tag{23c}$$

$$\text{C4} : 0 \le w_m \le 1. \tag{23d}$$

The constraints in *P1* are explained as follows: Constraint (23a) gives the binary choice of the offloading strategy. Constraint (23b) states that the number of devices participating in offloading cannot exceed the total number of devices. Constraint (23c) states that the allocated bandwidth cannot exceed the total bandwidth due to fading or loss in practice. Constraint (23d) gives a range of bandwidth allocation ratios for each device.

P1 is a mixed integer nonlinear program (*MINLP*). *P1* can be decomposed into multiple subproblems with separated objective and constraints. We can employ the Tammer decomposition method [18] to transform the original problem with high complexity into an equivalent master problem and a subproblem. *P1* can be transformed into the equivalent problem P2:

$$(P2): \min_{W} \left(\min_{A} O \right), \quad \text{s.t.C1-C4.} \tag{24}$$

Therefore, solving the problem of *P2* is equivalent to solving the following task offloading problem *P2.1*:

$$(P2.1): \min_{A} J^*(A), \quad \text{s.t.C1-C2.} \tag{25}$$

where $J^*(A)$ is the suboptimal value function corresponding to the bandwidth allocation problem *P2.2*, written as

$$(\text{P2.2}): J^*(A) = \min_{W} O, \quad \text{s.t.C2-C4.} \tag{26}$$

Note that the decomposition from problem of *P1* to problems of *P2.1* and *P2.2* does not change the optimality of the solution [18].

We refer to the algorithm and related parameters in [19] to solve the suboptimal solution of problem *P2*. And based on this strategy, we further study the training and inference performance of deep learning models by implementing model transfer through *TL*, in order to verify the performance of the proposed *HMECSF-TL* framework in terms of cost savings and in terms of image classification accuracy and training time.

5. Problem Formulation of Image Classification

The image classification task based on *HMECSF-TL* framework can not only improve the accuracy of image classification, but also improve the speed of image classification. This paper completes image classification based on model transfer through *TL* and achieves the purpose of efficient image classification by *QoC*-driven.

5.1. Convolutional Neural Network. CNN is a typical feedforward neural network for presentation learning and an influential deep learning framework for high-dimensional big data processing [3]. We first design a traditional *CNN* model, as shown in Figure 3.

5.2. Transfer Learning. The general model requires massive data to train its classifier to achieve high accuracy, but requires massive data to be provided to the personalized models of the end devices or to the local model of the edge server, which is obviously not feasible. To localize the general model and reduce the requirement of training samples, *TL* is increasingly adopted by model training research [3]. *TL* can tackle the problem with few additional training samples. This is achieved by transferring the knowledge from the old task into the new task, as shown in Figure 4. *TL* includes model transfer, feature transfer, relationship transfer, and sample transfer [8]. In *HMECSF-TL* framework, model transfer is used to transfer the parameters of the general model trained with a large number of high-quality data samples to the local model, which uses local data samples for further personalized training.

As shown in Figure 4, the parameters of the deep learning algorithm trained on the cloud and on the edge nodes are transferred to the lower layers step by step, and the model in the target domain is only retrained with a limited data samples for the fully connected layer without training from scratch, which can significantly shorten the training time. In *HMECSF-TL* framework, the source domain is the general model, and the target domain includes both the local model and the personalized model. Since the quality and quantity of data samples available on the cloud are higher than those in the target domain, the cloud has enough knowledge to perform more finer-grained image classification to better perform tasks such as image classification.

Although the traditional *CNN* model we designed earlier has realized the function of image classification, it is not satisfactory through experimental verification. *AlexNet* is a convolutional neural network that is 8 layers deep. The pretrained network can classify images into 1000 object categories. As a result, the network has learned rich feature representations for a wide range of images. So we designed an improved *Transferred AlexNet CNN* model using a pretrained "AlexNet" though *TL* under the *HMECSF-TL* framework, as shown in Figure 5.

And the above two neural network models are compared to verify whether the image classification performance of the *Transferred AlexNet CNN* model under the *HMECSF-TL* framework is better.

6. Evaluation Results

6.1. Datasets and Simulation Settings. In this paper, we use the dataset disclosed in [20], which contains 5932 images of contaminated rice leaves, involving four types of rice leaf diseases, namely bacterial blight, blast, brown spot, and tungro. *AlexNet* has an image input size of 227-by-227. So we processed the images according to 227-by-227 to adapt to the *Transferred AlexNet CNN* model validation; and set the first layer in the earliest designed traditional *CNN* model that is the image size adjustment layer, which can convert the dataset 227-by-227 images to 224-by-224 images. We set the learning rate to 0.0001 and the validation frequency to 50 iterations.

In the experiment, single processor *Intel i5-8250* U *CPU @1.60GHz*, *RAM 20.0GB*, *64-bit Windows 10 OS*, and *MATLAB R2020a* were used for simulation experiments.

6.2. Performance Evaluation. We perform *edge-end* collaborative training on two different *CNN* with the same initialization parameters to complete image classification according to the allocation strategy obtained by minimizing the weighted sum of costs. We conduct experimental verification on 400 images, 800 images, 1600 images, and 5932 images, respectively. The performance of the *Transferred AlexNet CNN* model is verified by experiments, and the performance of our proposed *HMECSF-TL* is evaluated.

As shown in Figure 6, the *HMECSF-TL* framework can effectively improve the image classification accuracy. Specifically, for image classification of 400 images, the max validation accuracy of the traditional *CNN* model is only 61.67%, while the max validation accuracy of the *Transferred AlexNet CNN* model is 79.16%. As the number of images increases, the max validation accuracy of both models improves. When 5932 images are processed, the max validation accuracy of traditional *CNN* model rises to 91.4%, while the max validation accuracy of the *Transferred AlexNet CNN* model is 99.66%. The max validation accuracy of the *Transferred AlexNet CNN* model based on the *HMECSF-TL* framework is consistently higher than that of the traditional *CNN* model.

As shown in Figure 7, the *Transferred AlexNet CNN* model based on the *HMECSF-TL* framework converges faster than the traditional *CNN* model, and the loss of the *Transferred AlexNet CNN* model is smaller after convergence.

As shown in Figure 8, the classification accuracy was evaluated on the validation set. We use the confusion matrix in the form of N rows and N columns to express the accuracy evaluation. These accuracy indicators reflect the accuracy of image classification from different aspects. The confusion matrix is calculated by comparing the position and classification of each measured pixel with the corresponding position and classification in the classified image. The resulting confusion matrix gives us additional insights on which categories are misclassified more frequently by the model.

As shown in Figure 9, the *HMECSF-TL* framework not only effectively improves the image classification accuracy, but also improves the image classification speed. Specifically, for image classification of 400 images, the processing time of the traditional *CNN* model is 7 minutes and 56 seconds, while the processing time of the *Transferred AlexNet CNN* model is 7 minutes and 7 seconds. As the number of images increases, the time required for the two models increases. When processing 5932 images, the processing time of the traditional *CNN* model is 148 minutes and 13 seconds, while the processing time of the *Transferred AlexNet CNN* model is only 102 minutes and 22 seconds. The processing time of the *Transferred AlexNet CNN* model based on the *HMECSF-TL* framework is always lower than that of the conventional *CNN*.

7. Conclusion

The *HMECSF-TL* framework studied in this paper improves the model training speed though *TL* and uses collaborative optimization to improve the accuracy of the model. We verified the effectiveness of the *Transferred AlexNet CNN* model inference work under the *HMECSF-TL* framework through simulation experiments and also verified the performance improvement of image classification under the *HMECSF-TL* framework. It can be seen that the proposed *HMECSF-TL* framework outperforms the benchmark strategy without *TL* in terms of reducing the model training time and improving the image classification accuracy, as well as reducing the offloading cost. In the next research, we plan to apply *HMECSF-TL* framework to more aspects, such as object detection and target tracking and expect the framework to play a greater role.

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.

Authors' Contributions

The author listed has made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Acknowledgments

The authors are grateful to the anonymous reviewers for their insightful comments which have certainly improved this paper. This work was supported in part by the Provincial Science and Technology Innovation Special Fund Project of Jilin Province (grant no. 20190302026GX); the Natural Science Foundation of Jilin Province (grant no. 20200201037JC); the Platform of Jilin Province Science and Technology (S&T) Department (grant no. 20190902011TC and YDZJ202202CXJD036); the Digital Transformation and Innovation Platform Construction Project of Jilin Province Development and Reform Commission (grant no. 2021C049); the Youth Foundation Project of Changchun Institute of Technology (grant no. 320200010); the Industrial Technology Research and Development Project of Jilin Province Development and Reform Commission (grant no. 2020C020-1); and the Jilin Province S&T Development Optimized S&T Resource Sharing Service Platform Project (Big data analysis and visual simulation application service platform for smart agriculture).

References

[1] F. B. Jiang, K. Z. Wang, L. Dong, C. H. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6252–6265, 2019.

[2] Y. Hao, J. Yang, M. Chen, M. S. Hossain, and M. F. Alhamid, "Emotion-aware video QoE assessment via transfer learning," *IEEE Multimedia*, vol. 26, no. 1, pp. 31–40, 2019.

[3] D. P. Wu, X. J. Han, Z. G. Yang, and R. Y. Wang, "Exploiting transfer learning for emotion recognition under cloud-edge-client collaborations," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 479–490, 2021.

[4] J. Wang, J. Hu, G. Y. Min et al., "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 64–69, 2019.

[5] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. D. Wu, "Object detection with deep learning: a review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[6] J. Chen and X. K. Ran, "Deep learning with edge computing: a review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[7] G. Y. Kim, R. Kim, S. C. Kim, K. D. Nam, S. U. Rha, and J. H. Yoon, "DNN inference offloading for object detection in 5G multi-access edge computing," in *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 389–392, Jeju Island, Korea, October 2021.

[8] S. J. L. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[9] J. Yosinski, J. Clune, and Y. S. Bengio, "How transferable are features in deep neural networks," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014.

[10] C. Li, S. H. Zhang, Y. Qin, and E. Estupinan, "A systematic review of deep transfer learning for machinery fault diagnosis," *Neurocomputing (Amsterdam)*, vol. 407, pp. 121–135, 2020.

[11] T. Han, C. Liu, W. G. Yang, and D. X. Jiang, "Deep transfer network with joint distribution adaptation: a new intelligent fault diagnosis framework for industry application," *ISA Transactions*, vol. 97, pp. 269–281, 2020.

[12] C. Sun, M. Ma, Z. B. Zhao, S. H. Tian, R. Q. Yan, and X. F. Chen, "Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2416–2425, 2019.

[13] X. Li, W. Zhang, and Q. Ding, "Cross-domain fault diagnosis of rolling element bearings using deep generative neural networks," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5525–5534, 2019.

[14] B. Yang, O. Fagbohungbe, X. L. Cao et al., "A joint energy and latency framework for transfer learning over 5G industrial edge networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 531–541, 2022.

[15] M. F. Leung and J. Wang, "Cardinality-constrained portfolio selection based on collaborative neurodynamic optimization," *Neural Networks*, vol. 145, pp. 68–79, 2022.

[16] M. F. Leung and J. Wang, "Minimax and bi-objective portfolio selection based on collaborative neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2825–2836, 2021.

[17] M. C. Yuen, S. C. Ng, and M. F. Leung, "A competitive mechanism multi-objective particle swarm optimization algorithm and its application to signalized traffic problem," *Cybernetics and Systems*, vol. 52, no. 1, pp. 73–104, 2021.

[18] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach," *Mathematical Research*, vol. 35, pp. 376–386, 1987.

[19] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[20] P. K. Sethy, N. K. Barpanda, A. K. Rath, and S. K. Behera, "Deep feature based rice leaf disease identification using support vector machine," *Computers and Electronics in Agriculture*, vol. 175, pp. 105527–105535, 2020.