Joint Task Offloading, CNN Layer Scheduling, and Resource Allocation in Cooperative Computing System

Rong Chai, *Senior Member, IEEE*, Xia Song, and Qianbin Chen, *Senior Member, IEEE*

***Abstract*—A two-tier cooperative computing system is considered in this article, where tier-1 consists of multiple mobile edge computing (MEC) servers and tier-2 consists of one mobile cloud computing (MCC) server. We assume that multiple mobile devices (MDs) are allowed to offload their tasks to the MEC and MCC servers. To facilitate task execution for the MDs, both types of servers are deployed with convolutional neural networks (CNNs). We assume that the cooperation between the MEC servers and the MCC server in task execution is enabled by scheduling different layers of the CNNs. To achieve efficient information interaction and task management, we first design a joint task management architecture. Stressing the importance of task execution latency, we formulate the joint task offloading, CNN layer scheduling and resource allocation optimization problem as an overall task latency minimization problem. To solve the formulated optimization problem, we transform it to three subproblems, i.e., CNN layer scheduling subproblem, task offloading subproblem and resource allocation subproblem, and solve the subproblems, respectively, by means of the extensive search algorithm, reformulation-linearization-technique and Lagrange dual algorithm. The effectiveness of the proposed algorithm is demonstrated via numerical simulations.**

***Index Terms*—Convolutional neural networks (CNN) layer scheduling, cooperative computing, mobile cloud computing (MCC) server, mobile edge computing (MEC) server, task offloading.**

I. Introduction

THE rapid advancements of mobile networks and smart terminals promote the emergence of advanced applications such as cloud gaming, graphics recognition, etc. However, the intensive computing requirements of these emerging applications pose challenges to mobile devices (MDs), which have relatively limited computation capability. While mobile cloud computing (MCC) can be applied to address these challenges, it suffers from long latency and low efficiency for transmitting huge amount of data from MDs to remote MCC servers [1]. To tackle the drawbacks of MCC technology, mobile edge computing (MEC) was proposed [2]. By deploying MEC servers at the edge of mobile networks, and allowing the tasks of MDs being offloaded to and executed at these servers, the task execution cost of the MDs can be reduced significantly.

While being capable of offering convenient information interaction with MDs, MEC servers may be subject to relatively limited computation capability especially compared to remote MCC servers. Hence, designing a two-tier cooperative computing system, which enables the cooperation between the MEC servers in the first tier and the remote MCC servers in the second tier for task execution will be highly desired as it may enhance the performance of task execution and achieve reasonable resource utilization. To further facilitate efficient task processing in computing systems, we may deploy convolutional neural networks (CNNs) on the computing servers [3]. As a typical CNN is composed of multiple layers with each layer having various data processing capability, it is possible to schedule a number of layers of the CNNs, process a portion of a task at one computing server, and execute the remaining task at other servers [4], [5].

In [6]–[8], the authors considered the task offloading problem in computing systems with neural networks. Lo *et al.* [6] introduced deep neural networks to MCC systems. To reduce the number of data transmissions, the authors designed an authentic operation unit to determine the task offloading strategy for the users. Li *et al.* [7] proposed a CNN-enabled fog computing system, which was capable of monitoring the manufacturing process in smart industry. They demonstrated that by offloading the computation burden to fog nodes, the data processing performance of the system could be improved. An MEC system with resource-constrained Internet-of-Things edge clusters was considered in [8]. By assuming that the MDs may cooperate with each other in task processing, a CNN-based computing framework was proposed to achieve efficient computation offloading.

While the cooperation between MEC servers and MCC servers in task processing is promising in CNN-enabled computing systems, the efficient CNN layer scheduling strategy should be designed by jointly taking into account the various computation capability of the servers and the task transmission cost between the MEC and MCC servers. Furthermore, in a practical computing system, the heterogeneous characteristics of MDs, especially in terms of local computation capability and task execution requirements, the various transmission performance between MDs and MEC servers, and the diverse computation capability of MEC servers pose challenges and difficulties to the design of task offloading and resource allocation strategies.

In this article, we consider a cooperative computing system consisting of multiple MDs, a number of MEC servers deployed

Manuscript received August 19, 2019; revised January 3, 2020 and March 22, 2020; accepted April 28, 2020. This work was supported by the National Natural Science Foundation of China under Grant 61831002. *(Corresponding author: Xia Song.)*

The authors are with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: chairong@cqupt.edu.cn; 2360321633@qq.com; cqb@cqupt.edu.cn).

Digital Object Identifier 10.1109/JSYST.2020.2991814

at network edge and one remote MCC server. Suppose both the MEC and MCC servers are deployed with CNNs for task processing. To facilitate the cooperation in task execution between two types of servers, we assume that the MEC servers may schedule a certain number of CNN layers for task execution and send the remaining amount of data to the MCC server for further task processing. More specifically, to execute the task for one MD, the CNN deployed on the associated MEC server of the MD may process the task by employing the first several layers. Then, the intermediate data of the task, which is indeed the output of the hidden layer of the CNN will be sent to the MCC server. Upon receiving the task from the MEC server, the CNN deployed on the MCC server schedules its corresponding layers and processes the remaining task for the MD.

We jointly study task offloading, CNN layer scheduling and resource allocation problem in the considered cooperative computing system.

Stressing the importance of the overall task latency of all the MDs, we formulate the joint task offloading, CNN layer scheduling and resource allocation problem as an overall task latency minimization problem. As the formulated problem is NP-hard, which cannot be solved conveniently, we transform the optimization problem to three subproblems, i.e., CNN layer scheduling subproblem, task offloading subproblem, and resource allocation subproblem, and solve the three subproblems by applying extensive search algorithm, reformulation-linearization-technique (RLT) [9], and Lagrange dual algorithm, respectively.

The major contributions of this article are summarized as follows.

1) In this article, we design a CNN-based cooperative computing system to facilitate the cooperation in task execution between a number of MEC servers and one MCC server. To achieve efficient information interaction and task management, we present a joint task management architecture based on which the proposed algorithm (PA) can be conducted.
2) Unlike previous work, which mainly stressed task offloading or resource allocation, respectively, we jointly consider the task offloading, CNN layer scheduling, and resource allocation problem in the CNN-based cooperative computing system, where the cooperation between the MEC and MCC servers in task execution is enabled.
3) Addressing the importance of overall task latency, we formulate an overall task latency minimization problem. To solve the optimization problem, we transform it to three subproblems, i.e., CNN layer scheduling subproblem, task offloading subproblem, and resource allocation subproblem, and then solve the three subproblems with extensive search algorithm, RLT and Lagrange dual algorithm successively.

The remainder of this article is organized as follows. An overview of related work is detailed in Section II. The system scenario, CNN-based cooperative computing scheme, and the joint task management architecture are presented in Section III. In Section IV, we present the formulation of the overall task latency minimization problem. The solution and the pseudocodes of the PA are detailed in Section V. Complexity analysis is presented in Section VI. Section VII examines the performance results of the PA via simulations. Finally, Section VIII concludes this article.

II. Related Work

An overview of the related work on task offloading and resource allocation for single-tier computing systems and multitier computing systems is presented in this section.

A. Task Offloading and Resource Allocation for Single-Tier Computing Systems

The task offloading and resource allocation problems in single-tier computing systems, such as MEC systems, fog computing systems, cloudlet systems, etc., were studied in [10]–[28].

The task offloading issue of one MD in a single-tier computing system was considered in [10]–[14]. Yu *et al.* [10] developed a dynamic task offloading framework for single-cell MEC system. The task offloading problem of the MD was formulated as a multiple label classification problem, which aimed to minimize the computational and offloading overhead and a deep supervised learning method was proposed to obtain the suboptimal offloading strategy.

In [11], the task offloading and power allocation problem was studied for unpredictable tasks in an MEC system. Under the constraints on the task execution capability of MEC servers and the transmit power of MDs, a joint optimal strategy was proposed, which aimed to achieve the maximal task processing capacity of the system. Kuange *et al.* [12] stressed the joint task scheduling and power allocation problem in an MEC system and proposed an alternation method to minimize the energy consumption required for task execution.

Janatian *et al.* [13] formulated the joint task offloading and power allocation problem of a fog computing system as an energy consumption minimization problem and presented a time-switching scheme to obtain the optimal computing strategy. In [14], the task offloading problem was studied for a single MD in an MCC system, where the task of the MD was characterized by an arbitrary component dependency graph. To minimize the energy consumption for task execution, the authors proposed an energy-efficient joint task scheduling and computation offloading scheme.

As an extension of single-MD computing systems, computing systems consisting of multiple MDs were considered in [15]–[23]. In [15]–[18], the authors stressed the energy consumption required for task execution. The task offloading and resource allocation problem in a multi-MD MEC system was formulated as an energy consumption minimization problem in [15]. To solve the optimization problem, the authors proposed an energy-efficient joint scheme (EEJS), which decoupled the original optimization problem to two-layer problems, i.e., lower-layer resource allocation problem and upper-layer computation offloading, and solved the two problems, respectively, by using the Lagrange dual method and the Hungarian method. You *et al.* [16] studied the joint task offloading and energy harvesting problem in a multi-MD MEC system employing time division multiple access (TDMA). By formulating the joint optimization problem as an energy consumption minimization problem, and deriving an offloading priority function for the MDs, a joint offloading scheduling and energy harvesting strategy was obtained.

In a green MEC system with the MDs being capable of harvesting renewable energy, the task offloading problem was formulated as an energy consumption minimization problem in [17]. The authors proposed a centralized greedy scheduling

algorithm and a distributed greedy scheduling algorithm for solving the optimization problem, respectively. In [18], the multi-MD task offloading problem was considered as a competitive game, and a Gauss–Seidel-like method was designed to acquire the Nash equilibrium of the task offloading decisions, which reduced the energy consumption of the MDs under the hard deadline constraint of task execution.

The task offloading problem of a multi-MD TDMA MEC system was investigated in [19]. The task offloading problem was formulated as a piecewise optimization problem, which minimized task execution latency, and a closed-form solution was obtained by using the Lagrange dual method. Leveraging the idea of software-defined networking (SDN) for ultradense networks (UDNs), Chen and Hao [20] proposed a task offloading scheme, which minimized task execution latency and maximized the battery lifetime of the MDs as well.

In [21], an edge assisted offloading system was designed for both infrastructure-based networks and ad-hoc networks. By applying sort-enumerate algorithm, the task split and offloading strategy was obtained to minimize task execution latency. Dinh *et al.* [22] investigated the tradeoff between energy consumption and task execution latency in a multitask multi-MEC system. By applying semidefinite relaxation algorithm, the optimal offloading strategy was obtained. In [23], aiming to minimize the computational overhead, the authors formulated the problem of computation offloading among multiple users as a game model, and demonstrated the existence of Nash equilibrium point.

In order to enhance the task execution performance of MEC systems, cooperation in task offloading among computing entities can be applied [24]–[28]. The cooperation among MEC servers was studied in [24]–[26]. In [24], the cooperative MEC offloading problem in UDN was studied, the authors presented a greedy approximation scheme so as to minimize the overall computational overhead. To minimize the task execution latency, Yang *et al.* [25] presented a cooperative MEC architecture for UDNs, and proposed a bottom-up dynamic programming algorithm to acquire the optimal offloading strategy. In [26], the collaborative task offloading problem for multiaccess MEC systems was studied, and a heuristic greedy offloading scheme was proposed to minimize system energy consumption.

Wang *et al.* [27] and Zhang *et al.* [28] considered the task offloading problem in vehicle networks. In [27], the mobile task offloading problem was studied for a cloudlet system, where the vehicle cloudlet servers were deployed to execute the tasks of vehicles, and a cloudlet relaying scheme was proposed to minimize the energy consumption of the vehicles. In [28], an MEC-based vehicular network framework was designed and a predictive combination-mode relegation method was proposed to achieve the cooperation of the MEC servers for reducing task offloading cost.

B. Task Offloading and Resource Allocation for Multitier Computing Systems

Multitier computing systems can be applied to further enhance the performance of task offloading systems [29]–[35].

Meng *et al.* [29] studied the task offloading problem in a three-tier mixed edge/cloud system and proposed a suboptimal strategy, which minimized system energy consumption. In [30], the task offloading problem was considered for a cooperative two-tier computing system serving one MD at certain time period, and a greedy algorithm was proposed to maximize the percentage of the tasks being offloaded successfully. To minimize task execution latency while guaranteeing the fairness among the MDs in a cooperative fog/cloud system, Du *et al.* [31] designed a low-complexity suboptimal method to determine the joint optimal offloading and resource allocation strategy.

In [32], in order to maximize system benefit while satisfying the tolerable execution latency in a mixed fog/cloud system, a matching game framework was designed, and a student project algorithm was proposed to obtain a user-oriented cooperation offloading strategy. The cooperation in task offloading in a mixed fog/cloud system was considered in [33]. To minimize the task execution latency of all the MDs, a branch and bound method was presented to solve the single-MD offloading problem and an iterative heuristic algorithm (IHRA) was proposed to determine the multi-MD resource allocation strategy. Although cooperation in task offloading was considered in [33], the authors assumed that the fog servers and the MCC server could only process individual tasks independently, resulting in a coarse-grained cooperation. In addition, as multiple MDs were not allowed to share the resources of one MEC server, the task execution performance was highly limited due to insufficient resource utilization.

In [34] and [35], D2D offloading scheme was jointly considered with MEC offloading to enable two-tier cooperative computing, where the tasks of MDs could be offloaded to their D2D peers or the associated MEC servers for execution. In [34], to maximize the percentage of the tasks being offloaded, the Karmarkars algorithm was proposed to determine the optimal task offloading strategy. Xu *et al.* [35] formulated the task offloading problem as a stackelberg cooperative game and designed a security-aware incentive mechanism to maximize network utility. In [36], traffic offloading in hybrid satellite-terrestrial systems was considered. An SDN-based spectrum sharing and traffic offloading mechanism was designed to achieve the cooperation and competition between ground base stations and the beam groups of satellite-terrestrial communication systems.

While the issues of task offloading and resource allocation in mobile computing systems have been addressed in the aforementioned research work, the cooperation mechanisms between the MEC and MCC servers, and the joint task offloading and resource allocation problem in CNN-enabled cooperative computing systems have not been studied extensively. In this article, we consider a practical cooperative computing system where both the MEC servers and the MCC server are deployed with CNNs. By scheduling various CNN layers at the two types of servers in a cooperative manner, desired task execution performance can be obtained. Stressing the performance of all MDs in terms of task execution latency, we formulate the joint task offloading, CNN layer scheduling and resource allocation problem as an overall task latency minimization problem. To solve the formulated problem, we decompose the problem into three subproblems, which are solved, respectively, to obtain the near optimal strategy.

III. System Model and Joint Task Management Architecture

A. System Scenario

In this article, we consider a two-tier cooperative computing system, which offers task offloading service to multiple MDs. The tier-1 of the system consists of N MEC servers and the tier-2 consists of one MCC server. To conduct task execution for the MDs, we assume that both the MCC server and the MEC

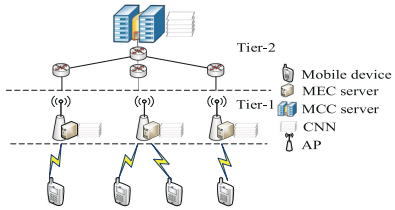


Fig. 1. System model.

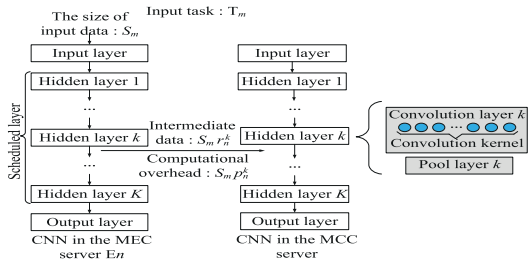


Fig. 2. CNN model.

servers are deployed with CNNs. To utilize computing resources efficiently, suppose the MDs may conduct local computing or offload their tasks to the associated MEC servers.

Let MD_m denote the mth MD, $1 \le m \le M$, where M is the number of MDs. Let $\mathrm{T} = \{\mathrm{T}_1, \ldots, \mathrm{T}_M\}$ denote the set of tasks, where T_m denotes the task of MD_m. While various types of heterogeneous tasks can be considered, without loss of generality, we assume that each task can be characterized by a 3-tuple. More specifically, T_m can be characterized by $\langle S_m, R_m^{\min}, D_m^{\max} \rangle$, where S_m denotes the size of the input data of T_m, $R_m^{\min}$, and $D_m^{\max}$ denote, respectively, the minimum transmission rate and the maximum tolerable execution latency of T_m. We denote $\mathrm{E} = \{\mathrm{E}_1, \ldots, \mathrm{E}_N\}$ as the set of MEC servers, where E_n denotes the nth MEC server, $1 \le n \le N$. Let B_n denote the bandwidth of the wireless link between E_n and the MDs, F_n denote the computation capability of E_n, and C_n denote the transmission rate of the link between E_n and the MCC server. Notice transmission rate here is also referred to as transmission capacity.

To improve the resource utilization of the MEC servers and the MCC server, we assume that the tasks of multiple MDs can be offloaded to one MEC server, in which case the resource sharing between these MDs has to be considered. In particular, the bandwidth of the wireless links between the MEC servers and the MDs, the computation resource of the MEC servers, and the transmission capacity of the link between the MEC servers and the MCC server should be allocated to the MDs in an optimal manner. The system model considered in this article is shown in Fig. 1.

B. CNN-Based Cooperative Computing Scheme

Aiming to achieve the performance advantages of both tiers in task execution, we propose a CNN-based cooperative computing scheme, which allows the tasks of MDs being processed jointly by the CNNs deployed on the MEC servers and that on the MCC server.

The CNNs deployed on the MEC and MCC servers consist of one input layer, one output layer, and a number of hidden layers. It should be mentioned that the CNN model considered in this article is indeed a logical model, which omits the practical realization details of the CNN and focuses on its functionality and performance especially in terms of task execution. In practice, besides input and output layers, one CNN model may also consist of multiple convolutional layers and multiple pool layers, and each convolutional layer is composed of one or multiple convolution kernels, which act as filters. In this article, for simplicity, we combine the function of convolution layers and pool layers, and define hidden layers in the CNN model with each hidden layer consisting of one convolution layer and one pool layer.

TABLE I
PARAMETERS OF CNNS

Parameter \ Value \ Layer	k=1	k=2	k=3	k=4	k=5
Reduction ratio (r_n^k)	0.4	0.2	0.16	0.128	0.1152
Computational overhead (p_n^k)	0.5	1.1	1.8	2.84	3.992

While the number of filters in convolution layers can be quite large, the amount of the output data of the filters can be reduced compared to that of the input data. After being processed by the pool layer, the size of the intermediate data will be further reduced. Hence, in general, the input data size will decrease at the output of the hidden layers or the output layer [5]–[8]. In this article, it is assumed that through being processed by a particular CNN layer, the input data size of the task and the computational overhead for processing the task will both be reduced. Let r_n^k denote the reduction ratio of the data size after being processed by the first kth hidden layers of the CNN on E_n, p_n^k denote the computational resource required for processing unit input data by the first kth hidden layers of the CNN on E_n, $n \in [1, N]$, $k \in [1, K]$, where K is the number of the hidden layers of the CNNs deployed on the MEC and MCC servers.

In Fig. 2, we consider the scenario that task T_m is jointly processed by the CNNs deployed on E_n and that deployed on the MCC server. The two CNNs are both composed of one input layer, one output layer and K hidden layers. As a simple example, we set the number of hidden layers as 5, i.e., $K = 5$, and set the CNN parameters including the reduction ratio r_n^k and the computational overhead p_n^k referring to [37], [38], which are shown in Table I.

Suppose the input data size of T_m is 5 Mb, i.e., $S_m = 5$ Mb, and E_n conducts task execution by scheduling the first and second hidden layers of its CNN model, i.e., $k^* = 2$. Accordingly, the resulted computational overhead of T_m can be computed as $S_m p_n^2 = 5.5$ GHz and the intermediate data size will reduce to $S_m r_n^2 = 1$ Mb. The intermediate data of T_m will, then, be sent to the CNN model deployed on the MCC server for further processing.

In Table II, we list some important notations used in this article.

C. Joint Task Management Architecture

To achieve efficient information interaction and task management in the cooperative computing system, we propose a joint task management architecture, as shown in Fig. 3. In the proposed architecture, we introduce three functional entities, i.e., global task management entity (GTME), local task management entity (LTME), and device task management entity (DTME).

TABLE II
SUMMARY OF KEY NOTATIONS

Notation	Description
S_m	Input data size of T_m
P_m	Transmit power of MD_m
F_m^0	Local computation capability of MD_m
p_m^0	Local computational overhead of MD_m task T_m
D_m	Local task execution latency of T_m
B_n	Wireless link bandwidth between E_n and MDs
F_n	Computation capability of E_n
C_n	Transmission capacity of the link between E_n and MCC server
r_n^k	Size reduction ratio of task being processed by the first k layers of the CNN model at E_n
p_n^k	Computational overhead of unit task data being processed by the first k layers of the CNN model at E_n
δ_{mn}^k	Number of CNN layers scheduled for T_m at E_n
D_{mn}	Total task latency of T_m when offloaded to E_n
D_{mn}^{t}	Transmission latency for offloading T_m to E_n
D_{mn}^{p}	Task execution latency of T_m at E_n
D_{mn}^{tc}	Transmission latency of T_m when transmitted from E_n to MCC server
D_{mn}^{pc}	Executing time of T_m at MCC server
α_{mn}	Bandwidth allocation ratio of T_m at E_n
β_{mn}	Computation capability allocation ratio of T_m at E_n
λ_{mn}	Link capacity allocation ratio of T_m at E_n when transmitted from E_n to MCC server

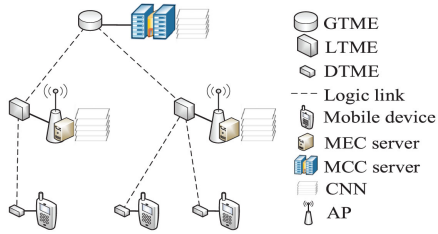


Fig. 3. Proposed joint task management architecture.

The main functions of the GTME, LTME, and DTME can be summarized as follows.

1) Global Task Management Entity: Being deployed on the MCC server, GTME acts as the centralized controller of the system. By interacting with LTMEs, GTME receives the status information of MDs and MEC servers, and conducts the proposed joint task offloading, CNN layer scheduling and resource allocation algorithm. The obtained joint strategy will be sent back to LTMEs.

2) Local Task Management Entity: By deploying on individual MEC servers, each LTME acts as a local controller of each server. Through interacting with its associated DTMEs and GTME, the LTME sends the state information of MDs to GTME, and forwards the received the optimal strategies to DTMEs.

3) Device Task Management Entity: Embedded inside each MD, DTME is responsible for collecting and storing channel information, the characteristics and the task requirements of the MD. By interacting with the associated LTME, DTMEs upload their collected information to LTME and receive the optimal strategy.

IV. Optimization Problem Formulation

In this section, the overall task latency of all MDs is examined and the joint task offloading, CNN layer scheduling and resource allocation problem is formulated as an overall task latency minimization problem.

A. Objective Function

Stressing the task execution performance of all MDs in the cooperative computing system, we define the overall task latency D as

$$D = \sum_{m=1}^{M} x_m D_m + \sum_{m=1}^{M} \sum_{n=1}^{N} x_{mn} D_{mn} \tag{1}$$

where x_m denotes the local task processing variable of T_m, i.e., if T_m is processed at MD_m locally, $x_m = 1$, otherwise, $x_m = 0$; D_m denotes the local task execution latency of T_m, x_{mn} denotes the task offloading variable of T_m, i.e., if T_m is offloaded to E_n, $x_{mn} = 1$, otherwise, $x_{mn} = 0$, D_{mn} denotes the total execution latency of T_m when offloaded to E_n.

The local task execution latency of T_m, denoted by D_m can be computed as

$$D_m = \frac{S_m p_m^0}{F_m^0} \tag{2}$$

where F_m^0 is the local computation capability of MD_m, and p_m^0 denotes the local computational overhead of T_m.

Jointly considering the time required to transmit tasks from MDs to MEC and MCC servers, as well as the task processing time at the servers, we formulate D_{mn} as

$$D_{mn} = D_{mn}^{\mathrm{t}} + D_{mn}^{\mathrm{p}} + D_{mn}^{\mathrm{tc}} + D_{mn}^{\mathrm{pc}} \tag{3}$$

where D_{mn}^{t} denotes the transmission latency for offloading T_m to E_n, D_{mn}^{p} denotes the task execution latency of T_m at E_n, D_{mn}^{tc} denotes the transmission latency of T_m when transmitted from E_n to the MCC server, and D_{mn}^{pc} denotes the executing time of T_m at the MCC server.

D_{mn}^{t} in (3) can be formulated as

$$D_{mn}^{\mathrm{t}} = \frac{S_m}{R_{mn}} \tag{4}$$

where R_{mn} is the transmission rate of T_m when transmitted from MD_m to E_n, which is given by

$$R_{mn} = \alpha_{mn} r_{mn} \tag{5}$$

where $\alpha_{mn} \in [0, 1]$ is the bandwidth allocation ratio of E_n assigned for transmitting T_m, r_{mn} is the achievable transmission rate with the whole bandwidth resource being assigned, i.e.,

$$r_{mn} = B_n \log_2 \left(1 + \frac{P_m g_{mn}}{\sigma^2}\right) \tag{6}$$

where P_m is the transmit power of MD_m, g_{mn} is the channel gain between MD_m and E_n, and σ^2 denotes noise power.

D_{mn}^{p} in (3) can be calculated as

$$D_{mn}^{\mathrm{p}} = \sum_{k=1}^{K} \delta_{mn}^k p_n^k \frac{S_m}{\beta_{mn} F_n} \tag{7}$$

where δ_{mn}^k denotes CNN layer scheduling variable. If the first k hidden layers of the CNN on E_n are scheduled to process T_m, $\delta_{mn}^k = 1$, otherwise, $\delta_{mn}^k = 0$, and $\beta_{mn} \in [0, 1]$ is the computation capability allocation ratio of E_n assigned to T_m.

D_{mn}^{tc} in (3) can be calculated as the ratio of the intermediate data size of T_m to the transmission rate of the link between E_n and the MCC server. In the case that the first kth hidden layers of the CNN are scheduled, the transmission latency can

be calculated as $\frac{r_n^k S_m}{\lambda_{mn} C_n}$. Given the constraints on the layer scheduling strategy δ_{mn}^k, we can express D_{mn}^{tc} as

$$D_{mn}^{\mathrm{tc}} = \sum_{k=1}^{K} \delta_{mn}^k r_n^k \frac{S_m}{\lambda_{mn} C_n} \tag{8}$$

where $\lambda_{mn} \in [0, 1]$ is the link capacity allocation ratio when E_n transmitting the intermediate data of T_m to the MCC server.

In this article, it is assumed that the MCC server has relatively high computation capability, thus, the latency term D_{mn}^{pc} in (3) is negligible.

B. Optimization Constraints

To design the joint task offloading, CNN layer scheduling and resource allocation strategy, which minimizes the overall task latency of the cooperative computing system, we consider the following constraints.

1) CNN Layer Scheduling Constraints: In this article, it is assumed that one or multiple layers of the CNNs on the MEC servers can be scheduled for executing the task of MDs, thus, we can express the CNN layer scheduling constraints as

$$\mathrm{C1}: \ \delta_{mn}^k \in \{0, 1\} \tag{9}$$

$$\mathrm{C2}: \sum_{k=1}^{K} \delta_{mn}^k \leq 1. \tag{10}$$

The above-mentioned constraints ensure that each task can only be processed by the first k layers of the CNN model on one MEC server.

2) Task Offloading Constraints: Assuming that one task can be executed locally or offloaded to at most one MEC server, the task offloading constraints can be expressed as

$$\mathrm{C3}: x_m \in \{0, 1\}, \ x_{mn} \in \{0, 1\} \tag{11}$$

$$\mathrm{C4}: x_m + \sum_{n=1}^{N} x_{mn} = 1, \ 1 \leq m \leq M. \tag{12}$$

It is apparent that C3 and C4 ensure that all the tasks should be executed either in local computing mode or MEC offloading mode.

3) Resource Allocation Constraints: In the case that multiple MDs choose to offload their tasks to one MEC server, the resource allocation constraints should be satisfied, which can be expressed as

$$\mathrm{C5}: \alpha_{mn}, \ \beta_{mn}, \ \lambda_{mn} \in [0, 1] \tag{13}$$

$$\mathrm{C6}: \sum_{m=1}^{M} \alpha_{mn} \leq 1 \tag{14}$$

$$\mathrm{C7}: \sum_{m=1}^{M} \beta_{mn} \leq 1 \tag{15}$$

$$\mathrm{C8}: \sum_{m=1}^{M} \lambda_{mn} \leq 1 \tag{16}$$

where C5 ensures that the fractions of resources, which are allocated to the tasks should be varying from 0 and 1. C6–C8 ensure that for multiple MDs, which share the same MEC server, the communication and computing resources allocated to the MDs cannot exceed the total amount of the resources on the server.

4) Data Rate and Latency Requirements: Stressing task offloading requirements, we assume that the wireless link between MD_m and E_n should meet the minimum transmission rate constraint when offloading T_m to E_n, i.e.,

$$\mathrm{C9}: \sum_{n=1}^{N} x_{mn} R_{mn} \geq R_m^{\min}, \text{ if} \sum_{n=1}^{N} x_{mn} = 1. \tag{17}$$

As the task execution latency of T_m should meet a tolerable maximum latency requirement, we express the task execution latency constraint as

$$\mathrm{C10}: x_m D_m + \sum_{n=1}^{N} x_{mn} D_{mn} \leq D_m^{\max}. \tag{18}$$

C9 and C10 ensure that task transmission and execution should meet the constraints on transmission rate and task execution latency. Since different tasks may pose different requirements on transmission rate and task execution latency, constraints C9 and C10 indeed partially characterize the heterogeneous features of the tasks.

C. Optimization Problem

Combining the aforementioned objective function with optimization constraints, we formulate the overall task latency minimization problem as

$$\min_{\substack{x_m, x_{mn}, \delta_{mn}^k \\ \alpha_{mn}, \beta_{mn}, \lambda_{mn}}} D$$
$$\text{s.t.} \quad \mathrm{C1} - \mathrm{C10}. \tag{19}$$

By solving the above-mentioned optimization problem, the joint task offloading, CNN layer scheduling and resource allocation strategy can be obtained.

V. Solution of the Optimization Problem

The formulated optimization problem in (19) is a nonconvex mixed integer nonlinear programming problem, which cannot be solved conveniently. In this section, we decompose the problem into three subproblems, i.e., CNN layer scheduling subproblem, task offloading subproblem, and resource allocation subproblem, and solve the three subproblems successively to obtain the joint strategy.

A. CNN Layer Scheduling Subproblem

In this section, we assume that the task offloading strategy of MDs is given, i.e., $x_{mn} = 1$, and no resource sharing among tasks is required, i.e., $\alpha_{mn} = 1$, $\beta_{mn} = 1$, $\lambda_{mn} = 1$. Substituting α_{mn}, β_{mn}, λ_{mn} into (3)–(8), we may rewrite D_{mn} as D_{mn}^0, i.e.,

$$D_{mn}^0 = \frac{S_m}{r_{mn}} + \sum_{k=1}^{K} \delta_{mn}^k \left(p_n^k \frac{S_m}{F_n} + r_n^k \frac{S_m}{C_n} \right). \tag{20}$$

As the only optimization variable contained in D_{mn}^0 is the CNN layer scheduling variable, i.e., δ_{mn}^k, we may design the optimal CNN layer scheduling strategy by minimizing D_{mn}^0. Hence, the optimization problem formulated in (19) is now

reduced to the CNN layer scheduling subproblem, which is formulated as

$$\min_{\delta_{mn}^k} D_{mn}^0 \quad \text{s.t. C1, C2, C10 in (19).} \tag{21}$$

The problem formulated in (21) is a one-variable optimization problem, we may solve it based on extensive search algorithm and obtain the optimal CNN layer scheduling strategy, denoted by $\delta_{mn}^{k,*}$.

B. Task Offloading Subproblem

By substituting $\delta_{mn}^{k,*}$ into the optimization problem formulated in (19), the optimization problem can be reduced to a joint task offloading and resource allocation problem. For simplicity, we define $\hat{D}_{mn}^{\mathrm{t}} = \frac{S_m}{r_{mn}}$, $\hat{D}_{mn}^{\mathrm{p}} = \sum_{k=1}^{K} \delta_{mn}^{k,*} \frac{p_n^k S_m}{F_n}$, $\hat{D}_{mn}^{\mathrm{tc}} = \sum_{k=1}^{K} \delta_{mn}^{k,*} \frac{r_n^k S_m}{C_n}$, and replace x_m by $1 - \sum_{n=1}^{N} x_{mn}$ according to constraint C4. Denoting the objective function D in (19) as D^0, we obtain

$$D^0 = \sum_{m=1}^{M} \left(1 - \sum_{n=1}^{N} x_{mn}\right) D_m + \sum_{m=1}^{M} \sum_{n=1}^{N} x_{mn} \left(\frac{\hat{D}_{mn}^{\mathrm{t}}}{\alpha_{mn}} + \frac{\hat{D}_{mn}^{\mathrm{p}}}{\beta_{mn}} + \frac{\hat{D}_{mn}^{\mathrm{tc}}}{\lambda_{mn}}\right). \tag{22}$$

It is apparent that D^0 is a second-order function of optimization variables x_{mn}, α_{mn}, β_{mn}, and λ_{mn}, which is notoriously hard to determine the minimum value. To address the difficulties, we first transform and relax variable x_{mn}, α_{mn}, β_{mn}, and λ_{mn}, then apply the RLT to solve the optimization problem.

To tackle the problem of fraction optimization, we define $\iota_{mn} = \frac{1}{\alpha_{mn}+\varepsilon_b}$, $\nu_{mn} = \frac{1}{\beta_{mn}+\varepsilon_c}$, and $\oslash_{mn} = \frac{1}{\lambda_{mn}+\varepsilon_f}$ where ε_b, ε_c, and ε_f are microscales introduced to avoid divide-by-zero error, then transform the original problem in (19) as follows:

$$\begin{aligned} \min_{\substack{x_{mn},\iota_{mn},\\ \nu_{mn},\oslash_{mn}}} & \sum_{m=1}^{M} \left(1 - \sum_{n=1}^{N} x_{mn}\right) D_m \\ & + \sum_{m=1}^{M} \sum_{n=1}^{N} x_{mn}(\iota_{mn} \hat{D}_{mn}^{\mathrm{t}} + \nu_{mn} \hat{D}_{mn}^{p} + \oslash_{mn} \hat{D}_{mn}^{tc}) \\ \text{s.t.} \quad & \mathrm{C3, C4, C9, C10 \, in \, (19)} \\ & \mathrm{C11}: \sum_{m=1}^{M} \frac{1}{\iota_{mn}} \le 1 + M\varepsilon_b, \iota_{mn} \in \left[\frac{1}{1+\varepsilon_b}, \frac{1}{\varepsilon_b}\right] \\ & \mathrm{C12}: \sum_{m=1}^{M} \frac{1}{\nu_{mn}} \le 1 + M\varepsilon_c, \nu_{mn} \in \left[\frac{1}{1+\varepsilon_c}, \frac{1}{\varepsilon_c}\right] \\ & \mathrm{C13}: \sum_{m=1}^{M} \frac{1}{\oslash_{mn}} \le 1 + M\varepsilon_f, \oslash_{mn} \in \left[\frac{1}{1+\varepsilon_f}, \frac{1}{\varepsilon_f}\right]. \end{aligned} \tag{23}$$

Problem (23) is still a nonconvex problem because of discrete variable x_{mn} and the second-order form of the optimization variables. We, now, employ discrete variable relaxation method to convert $x_{mn} \in \{0, 1\}$ to $0 \le x_{mn} \le 1$, then apply the RLT to linearize the objective function and constraints in (23). To linearize the second order terms $x_{mn}\iota_{mn}$, $x_{mn}\nu_{mn}$, and $x_{mn}\oslash_{mn}$, we define $\varphi_{mn} = x_{mn}\iota_{mn}$, $\varsigma_{mn} = x_{mn}\nu_{mn}$, and $\vartheta_{mn} = x_{mn}\oslash_{mn}$. Considering the constraints on x_{mn}, ι_{mn}, ν_{mn}, and $\oslash_{mn}$, we express the RLT bound-factor product constraints for φ_{mn}, ς_{mn}, and ϑ_{mn} as

$$\Xi_{mn}^{\varphi} = \begin{cases} \varphi_{mn} - \frac{1}{1+\varepsilon_b} x_{mn} \ge 0 \\ \iota_{mn} - \frac{1}{1+\varepsilon_b} - \varphi_{mn} + \frac{1}{1+\varepsilon_b} x_{mn} \ge 0 \\ \frac{1}{\varepsilon_b} x_{mn} - \varphi_{mn} \ge 0 \\ \frac{1}{\varepsilon_b} - \iota_{mn} - \frac{1}{\varepsilon_b} x_{mn} + \varphi_{mn} \ge 0 \end{cases} \tag{24}$$

$$\Xi_{mn}^{\varsigma} = \begin{cases} \varsigma_{mn} - \frac{1}{1+\varepsilon_c} x_{mn} \ge 0 \\ \nu_{mn} - \frac{1}{1+\varepsilon_c} - \varsigma_{mn} + \frac{1}{1+\varepsilon_c} x_{mn} \ge 0 \\ \frac{1}{\varepsilon_c} x_{mn} - \varsigma_{mn} \ge 0 \\ \frac{1}{\varepsilon_c} - \nu_{mn} - \frac{1}{\varepsilon_c} x_{mn} + \varsigma_{mn} \ge 0 \end{cases} \tag{25}$$

$$\Xi_{mn}^{\vartheta} = \begin{cases} \vartheta_{mn} - \frac{1}{1+\varepsilon_f} x_{mn} \ge 0 \\ \oslash_{mn} - \frac{1}{1+\varepsilon_f} - \vartheta_{mn} + \frac{1}{1+\varepsilon_f} x_{mn} \ge 0 \\ \frac{1}{\varepsilon_f} x_{mn} - \vartheta_{mn} \ge 0 \\ \frac{1}{\varepsilon_f} - \oslash_{mn} - \frac{1}{\varepsilon_f} x_{mn} + \vartheta_{mn} \ge 0. \end{cases} \tag{26}$$

After substituting φ_{mn}, ς_{mn}, and ϑ_{mn} into (23), we obtain the following optimization problem:

$$\begin{aligned} \min_{\substack{x_{mn} \\ \iota_{mn},\nu_{mn},\oslash_{mn} \\ \varphi_{mn},\varsigma_{mn},\vartheta_{mn}}} & \sum_{m=1}^{M} \left(1 - \sum_{n=1}^{N} x_{mn}\right) D_m \\ & + \sum_{m=1}^{M} \sum_{n=1}^{N} (\varphi_{mn} \hat{D}_{mn}^{t} + \varsigma_{mn} \hat{D}_{mn}^{p} + \vartheta_{mn} \hat{D}_{mn}^{tc}) \\ \text{s.t.} \quad & \mathrm{C4, C9, C10 \, in \, (19)} \\ & \mathrm{C11{-}C13 \, in \, (23)} \\ & \mathrm{C14}: 0 \le x_{mn} \le 1 \\ & \mathrm{C15}: \varphi_{mn} \in \Xi_{mn}^{\varphi} \\ & \mathrm{C16}: \varsigma_{mn} \in \Xi_{mn}^{\varsigma} \\ & \mathrm{C17}: \vartheta_{mn} \in \Xi_{mn}^{\vartheta}. \end{aligned} \tag{27}$$

It can be demonstrated that the optimization problem formulated in (27) is a convex optimization problem, therefore, we can

Algorithm 1: RLT-Based Task Offloading Algorithm.

1: Define $\hat{D}_{mn}^{\mathrm{t}} = \frac{S_m}{r_{mn}}$, $\hat{D}_{mn}^{\mathrm{p}} = \sum_{k=1}^{K} \delta_{mn}^{k,*} \frac{p_n^k S_m}{F_n}$, $\hat{D}_{mn}^{\mathrm{tc}} = \sum_{k=1}^{K} \delta_{mn}^{k,*} \frac{r_n^k S_m}{C_n}$
2: Replace x_m by $1 - \sum_{n=1}^{N} x_{mn}$
3: Define $\iota_{mn} = \frac{1}{\alpha_{mn}+\varepsilon_b}$, $\nu_{mn} = \frac{1}{\beta_{mn}+\varepsilon_c}$, and $\oslash_{mn} = \frac{1}{\lambda_{mn}+\varepsilon_f}$
4: Convert $x_{mn} \in \{0, 1\}$ to $0 \le x_{mn} \le 1$
5: Formulate the RLT bound-factor product constraints (24)-(26) for φ_{mn}, ς_{mn} and ϑ_{mn}
6: Solve the optimization problem formulated in (27) based on CVX
7: Obtain the optimal solution of (27), denoted by $\{\hat{x}_{mn}, \hat{\iota}_{mn}, \hat{\nu}_{mn}, \hat{\oslash}_{mn}, \hat{\varphi}_{mn}, \hat{\varsigma}_{mn}, \hat{\vartheta}_{mn}\}$
8: Determine x_{mn}^*, x_m^* based on (28)

solve it in polynomial time by using standard software tools, e.g., CVX [39].

Let $\{\hat{x}_{mn}, \hat{\iota}_{mn}, \hat{\nu}_{mn}, \hat{\oslash}_{mn}, \hat{\varphi}_{mn}, \hat{\varsigma}_{mn}, \hat{\vartheta}_{mn}\}$ denote the optimal solution of (27). As $\hat{x}_{mn}$ is a continuous approximation of x_{mn}, to obtain the binary task offloading strategy x^*_{mn} and local task processing variable x^*_m of (19), we define

$$\begin{cases} x^*_{mn} = 1, \ \text{if} \sum_{n=1}^{N} \hat{x}_{mn} \geq 0.5 \,\text{and}\, \dfrac{\hat{x}_{mn}}{\sum_{n=1}^{N} \hat{x}_{mn}} \geq 0.5 \\ x^*_{mn} = 0, \ \text{if} \sum_{n=1}^{N} \hat{x}_{mn} \geq 0.5 \,\text{and}\, \dfrac{\hat{x}_{mn}}{\sum_{n=1}^{N} \hat{x}_{mn}} < 0.5 \\ x^*_{m} = 1, \ \text{if} \sum_{n=1}^{N} \hat{x}_{mn} < 0.5. \end{cases} \quad (28)$$

Algorithm 1 summarizes the RLT-based task offloading algorithm.

C. Resource Allocation Subproblem

While the resource allocation strategy denoted by $\hat{\alpha}_{mn}$, $\hat{\beta}_{mn}$, and $\hat{\lambda}_{mn}$ can be obtained from $\hat{\iota}_{mn}$, $\hat{\nu}_{mn}$, $\hat{\oslash}_{mn}$, the suboptimality may occur due to the approximation of x_{mn}. In this section, given task offloading strategy x^*_{mn}, we further formulate the resource allocation subproblem and calculate the optimal resource allocation strategy by means of Lagrange dual method.

Based on the obtained task offloading strategy x^*_{mn}, the number of tasks, which are offloaded to E_n can be calculated. In the case that $\sum_{m=1}^{M} x^*_{mn} > 1$ for any E_n, i.e., more than one task is offloaded to E_n, resource sharing occurs at E_n, and the optimal resource allocation strategy should be designed. Let $\Phi_n = \{\mathrm{T}_m | x^*_{mn} = 1, 1 \leq m \leq M\}$ denote the set of tasks which are offloaded to E_n. We denote $\bar{D}_{mn}$ as the task latency of T_m when offloaded to E_n, i.e., $x^*_{mn} = 1$, we obtain

$$\bar{D}_{mn} = \frac{S_m}{R_{mn}} + \sum_{k=1}^{K} \delta^{k,*}_{mn} \left(\frac{p^k_n S_m}{\beta_{mn} F_n} + \frac{r^k_n S_m}{\lambda_{mn} C_n} \right). \quad (29)$$

The resource allocation subproblem can be expressed as

$$\min_{\alpha_{mn}, \beta_{mn}, \lambda_{mn}} \sum_{\mathrm{T}_m \in \Phi_n} \bar{D}_{mn} \quad \text{s.t.} \quad \mathrm{C5} - \mathrm{C10}. \quad (30)$$

The optimization problem (30) is a convex problem, which can be solved by applying the Lagrange dual algorithm. The Lagrangian of (30) is given by

$$\begin{aligned} & L(\alpha_{mn}, \beta_{mn}, \lambda_{mn}, \eta_{mn}, \mu_{mn}, \gamma_{mn}, \theta_{mn}, \omega_{mn}, \varepsilon, \tau, \psi) \\ = & \sum_{\mathrm{T}_m \in \Phi_n} \bar{D}_{mn} + \sum_{\mathrm{T}_m \in \Phi_n} \eta_{mn} \left(\bar{D}_{mn} - D^{\max}_m \right) \\ & + \sum_{\mathrm{T}_m \in \Phi_n} \mu_{mn} \left(R^{\min}_m - R_{mn} \right) + \sum_{\mathrm{T}_m \in \Phi_n} \gamma_{mn} \left(\alpha_{mn} - 1 \right) \\ & + \sum_{\mathrm{T}_m \in \Phi_n} \theta_{mn} \left(\beta_{mn} - 1 \right) + \sum_{\mathrm{T}_m \in \Phi_n} \omega_{mn} \left(\lambda_{mn} - 1 \right) \\ & + \varepsilon \left(\sum_{\mathrm{T}_m \in \Phi_n} \alpha_{mn} - 1 \right) + \tau \left(\sum_{\mathrm{T}_m \in \Phi_n} \beta_{mn} - 1 \right) \\ & + \psi \left(\sum_{\mathrm{T}_m \in \Phi_n} \lambda_{mn} - 1 \right) \end{aligned} \quad (31)$$

where $\eta_{mn}, \mu_{mn}, \gamma_{mn}, \theta_{mn}, \omega_{mn}, \varepsilon, \tau, \psi$ are Lagrange multipliers. The Lagrange dual problem of (30) can be formulated as

$$\max_{\substack{\eta_{mn}, \mu_{mn} \\ \gamma_{mn}, \theta_{mn}, \omega_{mn} \\ \varepsilon, \tau, \psi}} \min_{\alpha_{mn}, \beta_{mn}, \lambda_{mn}} L \quad \text{s.t.} \quad \eta_{mn}, \mu_{mn}, \gamma_{mn}, \theta_{mn}, \omega_{mn}, \varepsilon, \tau, \psi \geq 0. \quad (32)$$

By assuming the Lagrange multipliers are given, the optimal resource allocation strategy can be obtained as

$$\alpha^*_{mn} = \left[\sqrt{\frac{(1 + \eta_{mn}) S_m}{(\gamma_{mn} + \varepsilon - \mu_{mn}) r_{mn}}} \right]^+ \quad (33)$$

$$\beta^*_{mn} = \left[\sqrt{\frac{(1 + \eta_{mn}) S_m p^k_n}{(\theta_{mn} + \tau) F_n}} \right]^+ \quad (34)$$

$$\lambda^*_{mn} = \left[\sqrt{\frac{(1 + \eta_{mn}) S_m r^k_n}{(\omega_{mn} + \psi) C_n}} \right]^+ \quad (35)$$

where $[x]^+ = \max\{x, 0\}$.

Updating the Lagrange multipliers based on the gradient method, i.e.,

$$\eta_{mn}(t+1) = \left[\eta_{mn}(t) - \pi_1(t) \left(\bar{D}_{mn} - D^{\max}_m \right) \right]^+ \quad (36)$$

$$\mu_{mn}(t+1) = \left[\mu_{mn}(t) - \pi_2(t) \left(R^{\min}_m - R_{mn} \right) \right]^+ \quad (37)$$

$$\gamma_{mn}(t+1) = \left[\gamma_{mn}(t) - \pi_3(t) \left(\alpha_{mn} - 1 \right) \right]^+ \quad (38)$$

$$\theta_{mn}(t+1) = \left[\theta_{mn}(t) - \pi_4(t) \left(\beta_{mn} - 1 \right) \right]^+ \quad (39)$$

$$\omega_{mn}(t+1) = \left[\omega_{mn}(t) - \pi_5(t) \left(\lambda_{mn} - 1 \right) \right]^+ \quad (40)$$

$$\varepsilon(t+1) = \left[\varepsilon(t) - \pi_6(t) \left(\sum_{\mathrm{T}_m \in \Phi_n} \alpha_{mn} - 1 \right) \right]^+ \quad (41)$$

$$\tau(t+1) = \left[\tau(t) - \pi_7(t) \left(\sum_{\mathrm{T}_m \in \Phi_n} \beta_{mn} - 1 \right) \right]^+ \quad (42)$$

$$\psi(t+1) = \left[\psi(t) - \pi_8(t) \left(\sum_{\mathrm{T}_m \in \Phi_n} \lambda_{mn} - 1 \right) \right]^+ \quad (43)$$

where t is the iteration index, $\pi_q(t)$ is the step size, $1 \leq q \leq 8$. The convergence condition of the algorithm is set as

$$\begin{aligned} \sum_{T_m \in \Phi_n} & \left(|\eta_{mn}(t+1) - \eta_{mn}(t)| + |\mu_{mn}(t+1) - \mu_{mn}(t)| \right. \\ & + |\gamma_{mn}(t+1) - \gamma_{mn}(t)| + |\theta_{mn}(t+1) - \theta_{mn}(t)| \\ & + |\omega_{mn}(t+1) - \omega_{mn}(t)| + |\varepsilon(t+1) - \varepsilon(t)| \\ & \left. + |\tau(t+1) - \tau(t)| + |\psi(t+1) - \psi(t)| \right) \leq \varpi \end{aligned} \quad (44)$$

Algorithm 2: Lagrange Dual Method-Based Resource Allocation Algorithm.

Set the maximum iteration number $l_{\max}$ and maximum convergence condition ϖ
Initialize $\eta_{mn}, \mu_{mn}, \gamma_{mn}, \theta_{mn}, \omega_{mn}, \varepsilon, \tau$ and ψ
Set $l = 0$
repeat
  for $\mathrm{T}_m \in \Phi_n$ **do**
    Determine α_{mn}, β_{mn}, λ_{mn} according to (33)–(35)
    Update $\eta_{mn}, \mu_{mn}, \gamma_{mn}, \theta_{mn}, \omega_{mn}, \varepsilon, \tau$ and ψ according to (36)-(43)
    if (44) achieves **then**
      Convergence = true
      return $\alpha^*_{mn} = \alpha_{mn}$, $\beta^*_{mn} = \beta_{mn}$, $\lambda^*_{mn} = \lambda_{mn}$
    else
      $l = l + 1$
    end if
  end for
until Convergence = true or $l = l_{\max}$

Algorithm 3: Proposed Joint Task Offloading, CNN Layer Scheduling and Resource Allocation Algorithm.

for $m = 1$ to M **do**
  for $n = 1$ to N **do**
    Set $x_{mn} = 1$, $\alpha_{mn} = 1$, $\beta_{mn} = 1$, $\lambda_{mn} = 1$
    Solve CNN layer scheduling subproblem (21) by applying extensive search method, and obtain $\delta_n^{k,*}$
  end for
end for
Substitute δ_{mn}^k by $\delta_{mn}^{k,*}$ in the optimization problem formulated in (19), formulate task offloading subproblem (27)
Solve (27) based on RLT-based task offloading algorithm, and obtain x^*_m, x^*_{mn}
if $\sum_{m=1}^{M} x^*_{mn} > 1$ **then**
  Formulate resource allocation subproblem (30)
  Solve (30) by means of Lagrange dual method and obtain α^*_{mn}, β^*_{mn} and λ^*_{mn}
end if

where ϖ denotes the maximum tolerance.

Algorithm 2 summarizes the Lagrange dual method-based resource allocation algorithm. Replacing α_{mn}, β_{mn}, and λ_{mn} by α^*_{mn}, β^*_{mn}, and λ^*_{mn}, respectively, in D_{mn}, we will be able to obtain the optimal task latency of T_m at E_n when sharing bandwidth, computation, and link resources with other tasks.

Algorithm 3 presents the pseudocodes of the proposed joint task offloading, CNN layer scheduling, and resource allocation algorithm.

VI. Complexity Analysis

In this section, we analyze the computational complexity of the PA. As three subproblems, i.e., CNN layer scheduling subproblem, task offloading subproblem, and resource allocation subproblem are successively solved, we examine the complexity for solving the three subprolbems, respectively.

For the CNN layer scheduling subproblem, as the total number of scheduled CNN layers is set as K in this article, by using extensive search method to obtain the optimal CNN layer scheduling strategy, the number of operations for all MDs can be calculated as NMK, hence, the computational complexity is $O(NMK)$.

For the task offloading subproblem, since original optimization problem is transformed to a convex optimization problem in (27) by applying RLT and is, then, solved by using CVX, the computational complexity is polynomial complexity $O(n^3)$, where n is the number of RLT variables. Since the number of RLT variables involved in the task offloading subproblem is $(4(4+1))NM/2 = 10NM$, the computational complexity of the task offloading subproblem is $O((NM)^3)$.

To solve the formulated resource allocation subproblem, at each iteration, we need to perform $|\Phi_{\mathrm{n}}|$ operations, where $|\cdot|$ denotes the cardinality of one set. As each MEC server should update Lagrange multipliers according to (36)−(43), the number of operations for updating the Lagrange multiplier is $8|\Phi_{\mathrm{n}}| + 3$. Denote L as the number of iterations required for the algorithm to achieve convergence, the number of operations for conducting resource allocation is $8NL|\Phi_{\mathrm{n}}| + 3NL$. The corresponding computational complexity is $O(NL|\Phi_{\mathrm{n}}|)$.

Accordingly, the overall complexity for conducting task offloading, CNN layer scheduling, and resource allocation is $O(NMK)$+$O((NM)^3)$+$O(NL|\Phi_{\mathrm{n}}|)$=$O((NM)^3)$.

TABLE III
Simulation Parameters

Parameter	Value
Input data size (S_m)	$[1, 10]$Mb
Minimum transmission rate ($R_m^{\min}$)	1Mbps
Maximum tolerable task latency ($D_m^{\max}$)	$[1, 2]$s
Transmit power of MD_m (P_m)	0.6W
Computation capability of MD_m (F_m^0)	$\{2, 4\}$GHz
Local computation overhead of T_m (p_m^0)	$[2, 4]$ GHz/M
Noise power (σ^2)	-110dBm
Channel loss	128.1 + 27log(d)dB
Bandwidth of E_n (B_n)	10MHz
Computation capability of E_n (F_n)	$\{600, 800\}$GHz
Link capacity of E_n (C_n)	100Mbps
Reduction ratio (r_n^k)	$[0.4, 0.2, 0.16, 0.128, 0.1152]$
Computational overhead (p_n^k)	$[0.5, 1.1, 1.8, 2.84, 3.992]$GHz/M

VII. Simulation Results

In this section, the effectiveness of the PA is examined by MATLAB simulations. We consider a rectangular region with the area being 100 m×100 m, where the MEC servers and the MDs are randomly located. The number of MEC servers varies from 2 to 8, and the number of tasks is set from 25 to 35. In [40]–[42], we set the simulation parameters including the characteristics of tasks, MEC servers, the transmission channels and the MCC server, as listed in Table III. In the simulation, the number of hidden layers of CNN models is set as 5, i.e., $K = 5$, and the parameters r_n^k and p_n^k are chosen according to Table I.

We examine and compare the performance of the following algorithms, i.e., the PA, the RLT algorithm without Lagrange dual algorithm (RLT-0), the IHRA proposed in [33], the EEJS proposed in [15], average resource allocation (ARA) algorithm, which allocates the resource of the MEC servers equally to the associated MDs and a benchmark algorithm, i.e., the minimum distance based task offloading (MDO) algorithm, which allows the MDs to offload their tasks to the nearest MEC server.

The overall task latency versus the number of tasks obtained from above six algorithms is shown in Fig. 4. The number of

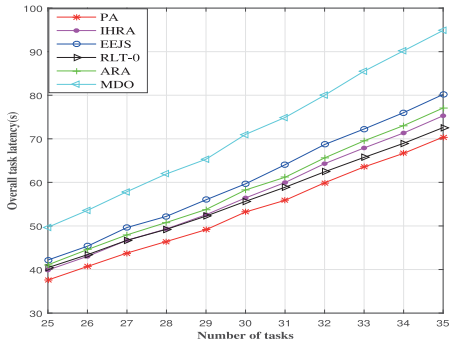


Fig. 4. Overall task latency versus number of tasks (different algorithms).

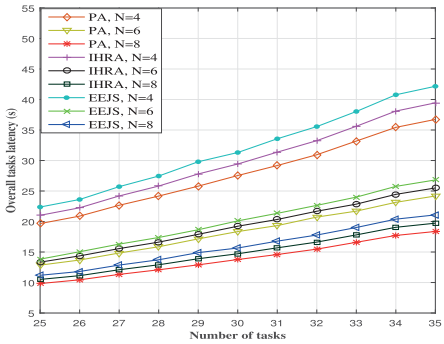


Fig. 5. Overall task latency versus number of tasks (different number of MEC servers).

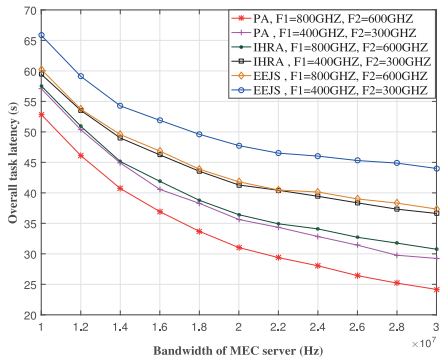


Fig. 6. Overall task latency versus bandwidth of MEC servers.

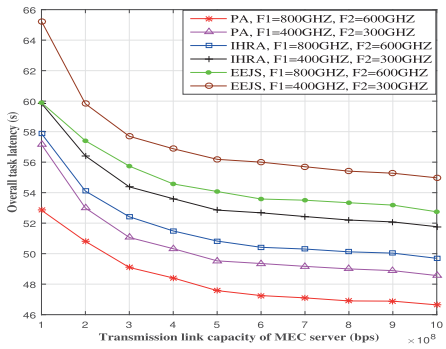


Fig. 7. Overall task latency versus link capacity between the MEC servers and the MCC server.

MEC servers is chosen as two in plotting the curves. While the overall task latency increases when the number of tasks of MDs increases for all the algorithms, the PA offers the minimum overall task latency compared with other algorithms. This is mainly benefited from the joint optimization of task offloading, CNN layer scheduling and resource allocation, and the combination of the RLT and Lagrange dual method in the cooperative computing system. Furthermore, it can be seen from the figure that the task latency obtained from the MDO algorithm and the EEJS are undesired. The reason is that latency minimization is not the major concern of both algorithms. Although the IHRA, ARA, and RLT-0 algorithms outperform the MDO algorithm and EEJS, they achieve high latency than the PA due to highly limited cooperation between the MEC and MCC servers and the insufficient utilization of system resources.

We examine the overall task latency performance of three algorithms, i.e., the PA, IHRA, and EEJS in Fig. 5. Different numbers of MEC servers, i.e., $N = 4, 6, 8$ are considered in the simulation. From the figure, we can see that the overall task latency can be reduced as the number of MEC servers increases. This is because the increase of the number of MEC servers results in the increase of the available communication and computing resources, thus offering higher degree of flexibility and efficiency in task offloading and resource allocation, and lower task latency in turn. Comparing the three algorithms, we can observe that the EEJS exhibits the worst latency performance. This is because it aims to achieve optimum energy efficiency, thus, results in undesired latency performance. The IHRA offers better latency performance than the EEJS, however, its performance is inferior to the PA, which is mainly because fine-grained resource sharing is not applied in the IHRA.

To examine the impacts of bandwidth and link capacity on the overall task latency, we plot Figs. 6 and 7, where the bandwidth is varying within the range between 10 and 30 MHz, and the capacity is varying from 100 to 1000 Mb/s. In plotting both figures, we set the number of MEC servers and tasks as 2 and 30, respectively. From Fig. 6, we can see that overall task latency decreases as the bandwidth of MEC servers increases. This is because higher bandwidth offers better transmission performance when offloading tasks to the MEC servers, hence resulting in lower overall task latency. Similarly, from Fig. 7, we can see that higher transmission capacity of the links between the MEC servers and the MCC server results in lower task latency.

In Figs. 6 and 7, we also examine the performance of the PA, EEJS, and IHRA obtained from different computation capability of MEC servers, where F1 and F2 denote, respectively, the computation capability of the MEC servers E1 and E_2. It can be observed from both figures that higher computation capability of MEC servers offers better task latency performance, which is benefited from reduced task execution latency. In addition, comparing the performance of the three algorithms in both figures, we can see that the PA offers better performance than the EEJS and IHRA.

In Fig. 8, we examine the performance of two baseline algorithms and the PA with and without load distribution constraint on MEC servers. To consider the load distribution of the MEC servers, we set a maximum number of tasks that can be executed at each MEC server. For baseline Algorithm 1, we assume that

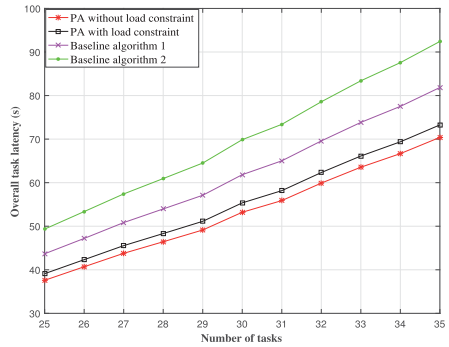


Fig. 8. Overall task latency versus number of tasks (different task offloading schemes).

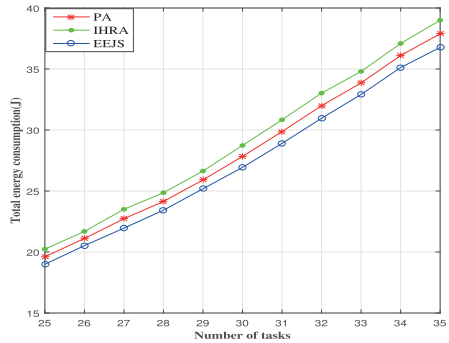


Fig. 9. Total energy consumption versus number of tasks (different algorithms).

MDs can only offload their tasks to the MEC servers and no MCC server is available. For baseline Algorithm 2, we assume that no MEC servers are deployed and MDs can only offload their tasks to the MCC server. From the figure, we can observe that the PA offers better performance than the two baseline algorithms, indicating that task execution performance enhancement can be acquired from the cooperative computing between the MEC and MCC servers. We can also see that the performance of the PA with load constraint is slightly weaker than that of the PA without load constraint. The reason is that load distribution condition limits the number of MDs offloading task to individual MEC servers, and hence, the resource of the MEC server may not be utilized efficiently.

In Fig. 9, the total energy consumption of MDs versus the number of tasks obtained from the PA, IHRA, and EEJS is plotted. The number of MEC servers is chosen as two in plotting the curves. We observe that as the number of tasks increases, the total energy consumption obtained from three algorithms increases accordingly. Comparing the performance obtained from three algorithms, we can see that the PA requires slightly higher energy consumption compared with the EEJS, this is because the EEJS aims to achieve the optimal energy consumption. However, we also notice that the gap between the two algorithms is relatively small. Furthermore, comparing the PA with the IHRA, we can see that the PA offers better energy consumption performance. The reason is the IHRA fails to consider the fine-grained resource sharing at the MEC and MCC servers, thus, may result in undesired task execution performance.

VIII. Conclusion

In this article, we proposed a cooperative computing system based on CNN models. To achieve efficient information interaction and task management, a joint task management architecture was designed. Based on the architecture, we formulated an overall task latency minimization problem, and solved the problem to obtain the joint task offloading, CNN layer scheduling and resource allocation strategy. The performance of the PA, previously proposed algorithms and baseline algorithms was examined via simulation, and the effectiveness of the PA was demonstrated.

While performance improvement in task execution can be seen clearly by applying the PA in CNN-based cooperative computing systems, it should be noticed that the PA is a centralized scheme, which conducts information collection and algorithm realization under the management of a central controller, i.e., GTME in Fig. 3, and, therefore, additional signaling interaction between various task management entities is required. In addition, in this article, we consider a quasi-static scenario of which one MD only associates with one MEC server and does not switch from one MEC server to another during task execution, hence, the task migration among various MEC servers is not taken into account. In our future work, we may extend current system model to a dynamic scenario and consider the possible task migration among MEC servers, in which case, the cooperation between the MEC servers and the additional costs due to task caching and task transmission should be considered.

References

[1] X. Fu, S. Secci, D. Huang, and R. Jana, "Mobile cloud computing," *IEEE Commun. Mag.*, vol. 53, pp. 61–62, Mar. 2015.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Comm. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.

[3] M. U. Yaseen, A. Anjum, and N. Antonopoulos, "Modeling and analysis of a deep learning pipeline for cloud based video analytics," in *Proc. IEEE/ACM Int. Conf. Big Data Comput. Appl. Technol.*, 2017, pp. 121-130.

[4] L. Li, K. Ota, and M. Dong, "Eyes in the dark: Distributed scene understanding for disaster management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3458–3471, Dec. 2017.

[5] J. Chen and X. Ran, "Deep learning with edge computing: a review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[6] C. Lo, Y. Y. Su, C. Y. Lee, and S. C. Chang, "A dynamic deep neural network design for efficient workload allocation in edge computing," in *Proc. IEEE Int. Conf. Comput. Des.*, Nov. 2017, pp. 273–280.

[7] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: efficient manufacture inspection system with fog computing," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4665–4673, Oct. 2018.

[8] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "Deepthings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.

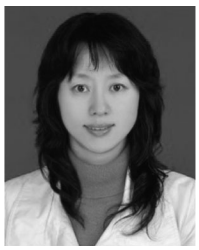
[9] E. Dalkiran and H. D. Sherali, "RLT-POS: Reformulation-Linearization-Technique-based optimization software for solving polynomial programming problems," *Math. Program. Comput.*, vol. 8, no. 3, pp. 337–375, Feb. 2016.

[10] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. IEEE 28th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2017, pp. 1–6.

[11] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Power-constrained edge computing with maximum processing capacity for IoT networks," *IEEE Int. Things J.*, vol. 6, no. 3, pp. 4330–4343, Jun. 2019.

[12] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Int. Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019, doi: 10.1109/JIOT.2019.2911455.

[13] N. Janatian, I. Stupia, and L. Vandendorpe, "Optimal resource allocation in ultra-low power fog-computing SWIPT-based networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2018, pp. 1–6.
[14] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 301–313, Apr. 2019.
[15] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems," in *Proc. IEEE Int. Conf. Commun. (ICC'18)*, Jul. 2018, pp. 1–6.
[16] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[17] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep./Oct. 2019, doi: 10.1109/TSC.2018.2826544
[18] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 87–96, Jan. 2017.
[19] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.
[20] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[21] H. Harvey, Y. Mao, Y. Hou, and B. Sheng, "Edos: Edge assisted offloading system for mobile devices," in *Proc. 26th Int. Conf. Comput. Commun. Netw.*, Sept. 2017, pp. 1–9.
[22] T. Q. Dinh, J. Tang, Q. D. La, and Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 4798–4810, Aug. 2017.
[23] X. Chen, J. Lei, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[24] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultra-dense IoT networks," *IEEE Int. Things J.*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.
[25] T. Yang, H. Zhang, H. Ji, and X. Li, "Computation collaboration in ultra dense network integrated with mobile edge computing," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, 2017, pp. 1–5.
[26] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.
[27] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11181–11191, Nov. 2018.
[28] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile edge computing for vehicular networks: A promising network paradigm with predictive offloading," *IEEE Veh. Tech. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
[29] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, Oct. 2017.
[30] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
[31] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.
[32] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, "Joint radio and computational resource allocation in IoT fog computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7475–7484, Aug. 2018.
[33] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Int. Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019, doi: 10.1109/JIOT.2018.2868616.
[34] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.
[35] J. Xu, L. Chen, K. Liu, and C. shen, "Designing security-aware incentives for computation offloading via device-to-device communication," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 6053–6066, Sep. 2018.
[36] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2202–2217, Oct. 2018.
[37] A. G. Howard, M. Zhu, B. Chen, and D. Kalenichenko, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *Comput. Sci.*, *arXiv: 1704.04861*.
[38] Y. Jia and S. Evan, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, Nov. 2014, pp. 675–678.
[39] M. Grant and S. Boyd, "CVX: MATLAB software for disciplined convex programming, version 2.1," Mar. 2014. [Online]. Available: http://cvxr.com/cvx
[40] 3GPP TS36.331, "Evolved Universal Terrestrial Radio Access (E-UTRA), Radio Resource Control(RRC), Protocol specification," Tech. Rep. 3GPP TS36.331, Dec. 2011.
[41] Y. Zhai, E. Mbarushim, W. Li, J. Zhang, and Y. Guo, "Lit: A high performance massive data computing framework based on CPU/GPU cluster," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sept. 2013, pp. 1–8.
[42] T. Thinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.



Rong Chai (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from McMaster University, Hamilton, ON, Canada, in 2008.

She then joined the School of Communication and Information Engineering, Chongqing University of Posts and Technology, Chongqing, China, where she is currently a Professor. Her research interests include wireless communication and network theory.

Xia Song received the M.S. degree in information and communication engineering from Chongqing University of Posts and Technology, Chongqing, China, in 2020.

His research interests include resource management of wireless communication networks, and mobile edge computing.

Qianbin Chen (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2006.

In 1988, he was with the School of Communication and Information Engineering, Chongqing University of Posts and Technology, Chongqing, China, where he is currently a Professor. His research interests include wireless communications and network theory.