Task Partitioning and Offloading in DNN-Task Enabled Mobile Edge Computing Networks

Mingjin Gao, Rujing Shen, Long Shi, Wen Qi, Jun Li, Yonghui Li

Abstract—Deep neural network (DNN)-task enabled mobile edge computing (MEC) is gaining ubiquity due to outstanding performance of artificial intelligence. By virtue of characteristics of DNN, this paper develops a joint design of task partitioning and offloading for a DNN-task enabled MEC network that consists of a single server and multiple mobile devices (MDs), where the server and each MD employ the well-trained DNNs for task computation. The main contributions of this paper are as follows: First, we propose a layer-level computation partitioning strategy for DNN to partition each MD's task into the subtasks that are either locally computed at the MD or offloaded to the server. Second, we develop a delay prediction model for DNN to characterize the computation delay of each subtask at the MD and the server. Third, we design a slot model and a dynamic pricing strategy for the server to efficiently schedule the offloaded subtasks. Fourth, we jointly optimize the design of task partitioning and offloading to minimize each MD's cost that includes the computation delay, the energy consumption, and the price paid to the server. In particular, we propose two distributed algorithms based on the aggregative game theory to solve the optimization problem. Finally, numerical results demonstrate that the proposed scheme is scalable to different types of DNNs and shows the superiority over the baseline schemes in terms of processing delay and energy consumption.

Index Terms—Deep neural networks, mobile edge computing, task partitioning and offloading, aggregative game

1 Introduction

Deep neural networks (DNNs) are widely used in various mobile applications (e.g., Apple Siri and Google Assistant) due to outstanding performance of artificial intelligence (e.g., computer vision, natural language processing, and big data analysis) [1–3]. However, DNN tasks are computationally expensive on the side of mobile devices (MDs) because of limited energy and computing capability of MDs. Recently, mobile edge computing (MEC) is a promising technology to meet the high-computation requirements of DNN tasks and relieve the computation burden of MDs [4–11]. In MEC, a number of edge servers with abundant computation resources are placed in the proximity of MDs [12], and each MD decides to offload a portion of its task to the edge servers under constraints of computation and communication resources. In this context, task partitioning and offloading are enabling technologies for practical implementation of MEC.

First, the general goal of task partitioning is to partition the tasks into into two parts with one executed at the MD and the other offloaded for server execution [13]. For conventional MEC without DNNs, most existing partitioning strategy relies on an assumption that the task is fully divisible and can be arbitrary partitioned according to different network settings. For example, [14] jointly optimized the energy transmit beamforming at the access point, the central processing unit frequencies, the numbers of offloaded bits at the users, and the time allocation among users. Recently, [15] investigated a resource allocation policy to maximize the processing capacity for the MEC-aided IoT networks with constrained power and unpredictable tasks. Later, the work in [16] jointly optimized unmanned aerial vehicle position, time slot allocation, and computation task partition to minimize the energy consumption of multiuser aerial mobile edge computing systems. However, in view of characteristics of DNNs, it should be stressed that the fully divisible assumption cannot be applied to DNN-task enabled MEC for the following reasons. First, the DNN has multilayer structures, and computation within the same layer cannot be readily partitioned due to strong interdependence among neurons. Second, arbitrary partitioning is technically unavaliable, because the partitioning granularity of tasks cannot be arbitrarily small in programming. Therefore, how to design an efficient partitioning scheme for the DNN tasks remains challenging.

To address this challenge, prior works in the field of DNN-task partitioning have made few effort [17, 18]. Ref. [17] proposed a lightweight scheduler, i.e., Neurosurgeon, to automatically partition DNN computation between the MD and the data center at the granularity of neural network layers. Nevertheless, Neurosurgeon is designed for a simple edge computing network with one user and one server, which is not feasible to the multiuser MEC networks. Ref. [18] assumed that the DNN has

M. Gao (gaomingjin@ict.ac.cn), and R. Shen (shenrujing@ict.ac.cn) are with the Institute of Computing Technology, Chinese Academy of Science as well as the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing, China. R. Shen is also with the University of Chinese Academy of Sciences, Beijing, China. L. Shi (slong1007@gmail.com) and J. Li (jun.li@njust.edu.cn.) are with the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, 210094, China. W. Qi is with China Telecom Research Institute. Y. Li is with the university of Sydney, Australia. (Corresponding author: Jun Li.)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

been partitioned into a fixed set of partitions. Instead of optimizing the partitioning points, [18] optimized the deployment of fixed DNN partitions at the MEC servers in order to minimize the end-to-end delay. However, the research on a general DNN-task partitioning scheme in the multi-user MEC networks remains open, which primarily motivates the DNN-task partitioning design in our work.

Second, upon the partitioning strategy, task offloading can reduce the computation burden of MDs by migrating part of tasks from resource-limited MDs to servers [19–21]. Many existing offloading schemes focused on the power minimization. For example, [22] developed an online joint radio and computational resource management algorithm for multi-user MEC systems, with the objective of minimizing the long-term average weighted sum power consumption of the MDs and the MEC server. Ref. [23] jointly optimized the power and time allocation to reduce the energy consumption of computation offloading in a non-orthogonal multiple access assisted MEC. It is noted that the aforementioned works in [22, 23] assumed that the server owned sufficient computing resources such that the subtasks offloaded from all MDs could be concurrently executed by the server. However, the ideal assumption is not true in reality due to limited computing capability of the server.

To cope with this problem, [24, 25] investigated the pricing-driven offloading schemes to optimize the usage of limited computing resources in the servers, where the server charges the subtasks offloaded from each MD according to the resource provisioning of servers. For example, [24] proposed a distributed price-adjustment algorithm to motivate uniform provisioning of resources on different servers. Ref. [25] used a two-tier matching game to optimize computing resource allocation and computing service pricing. Specifically, the first tier targets at achieving maximum social welfare, and the second tier focuses on efficiently utilize limited computing resources on servers. However, without an efficient scheduling design for the task offloading, the server with limited task buffer in [24, 25] can be easily congested by the overloaded tasks. Moreover, to our best knowledge, most existing offloading schemes are not scalable to the DNN-task enabled MEC. Motivated by these issues, the DNN-task enabled MEC networks demands a compatible offloading strategy that enables the task scheduling at the server.

In this paper, we consider a DNN-task enabled MEC network with multiple MDs and a single MEC server, where the server and each MD employ the well trained DNNs for task computation. The primary goal of this paper is to develop a joint design of task partitioning and offloading for the MEC network. To be specific, the contributions of this paper are summarized as follows:

- We propose a general DNN-task partitioning and offloading scheme in the multi-user MEC network, which is based on layer-level computation partitioning method for DNN tasks. The proposed scheme is applicable to different MDs with different DNNs.
- We develop a delay prediction model for DNN to characterize the computation delay of each subtask at the MD and server (see Section 2.2). The model is scalable to different types of DNNs, and can be used to analyze the energy consumption of each subtask.
- We design a slot model and a dynamic pricing strategy for the server to motivate MDs towards an efficient task scheduling under budget and delay constraints (see Section 3). The proposed slot model is designed according to parallel and sequential scheduling rules of offloaded subtasks, and the pricing strategy is developed to dynamically adjust the unit computation price of each slot.
- We jointly optimize the design of task partitioning and offloading to minimize each MD's cost that includes the computation delay, energy consumption, and price paid to the server. In particular, we propose two distributed algorithms based on the aggregative game theory to solve the optimization problem (see Section 4).

Simulations results demonstrate that the proposed scheme is scalable to different types of DNNs, and can reduce processing delay and energy consumption compared with the baseline schemes.

The rest of this paper is organized as follows. Section 2 details the system model. Section 3 describes the slot model and the dynamic pricing strategy. Section 4 formulates the optimal task partitioning and offloading problem. Section 5 shows the numerical results. Finally, we conclude this paper in Section 6.

2 System Model

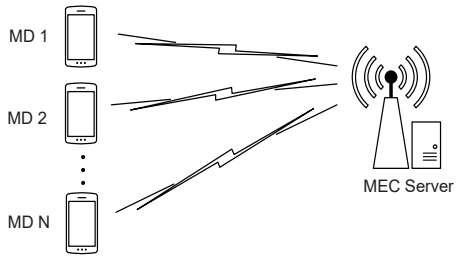
In this section, we first propose a DNN-task partitioning and offloading model, and then formulate the processing delay and energy consumption of each MD.

2.1 DNN-Task Partitioning and Offloading Model

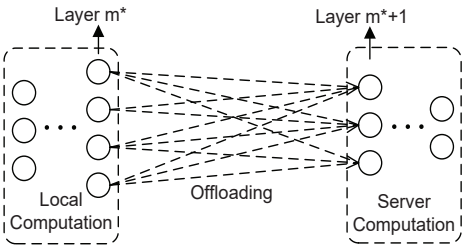
Consider an MEC network with a single server that serves N MDs. Let $\mathcal{N} = \{1, \ldots, N\}$ denote a set that collects the indices of MDs. Each MD employs a well-trained DNN to compute its task. For simplicity, we call a task computed by the DNN as a DNN task. Suppose that the server is capable of training and storing weights of each MD, and has a library of well-trained DNNs that cover all DNN types employed by the MDs. Thus, the server can compute DNN tasks offloaded from any MD.

For this MEC network, we propose a DNN-task partitioning and offloading model to alleviate the computing burden of MDs and reduce the cost of each MD. The proposed task partition features the layer-level computation partitioning strategy based on different characteristics of each layer (e.g., processing delay, energy consumption, and input data sizes) in the DNN. This strategy is primarily motivated by a Neurosurgeon system in [17] that can automatically partition a DNN between the MD and cloud.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing



(a) An MEC network with N MDs and one MEC server.



(b) If an MD partitions its DNN task between the layer m^* and the layer (m^*+1), the former m^* layers will be computed locally, and the output data from the layer m^* will be offloaded to the server.

Fig. 1. The MEC network with the proposed task partitioning and offloading.

Before delving into the task partition, let us first study the structure of DNN. The DNN consists of multiple layers such as convolutional layer, fully connected layer, and normalization layer [26]. We suppose that the DNN employed by MD n has M_n layers. Let $\mathcal{M}_n = \{1, \ldots, M_n\}$ denote a set that collects the indices of layers. For a forward pass through the DNN, the output of a layer is the input to the next layer. As such, the computation is sequentially executed by each layer from left to right in the DNN. In this context, the computation in the DNN can be decomposed into M_n subtasks, where the subtask m corresponds to the computation at the layer m. Fig. 1 illustrates the proposed task partitioning and offloading model in the MEC network. The figure considers that an MD partitions its DNN task between the layer m^* and the layer (m^*+1). It implies that, the task is locally computed by the DNN from the layer 1 to the layer m^*, and the output data from the layer m^* is offloaded to the server for further computing. We define the layer index m^* as the partition point of an MD's DNN task. Assuming that the server is aware of the partition point through a reliable link, the server can compute the output data offloaded from the MD using the same DNN with the layer (m^*+1) to the layer M_n. In particular, the whole task will be either locally computed at the MD if $m^* = M_n$ or offloaded to the server if $m^* = 0$.

2.2 Processing Delay and Energy Consumption Analysis

In this subsection, we analyze delay and energy consumption caused by the task partitioning and offloading.

2.2.1 Processing Delay

The processing delay consists of local computation delay (i.e., computation delay at the MD), uploading delay (i.e., delay induced by data uploading from the MD to the server), server computation delay (i.e., computation delay at the server), and downloading delay (i.e., delay induced by data downloading from the server to the MD).

Computation delay: The computation delay includes local computation delay and server computation delay. Different from traditional tasks, it is difficult to characterize the computation delay at each layer, since the computation of each layer depends on many dynamic factors such as the output data from the former layer, layer type, and layer structure. To cope with this problem, we develop a delay prediction model to predict both local computation delay and server computation delay at each layer. To this end, we employ a well-trained DNN to compute image recognition tasks. Using the source code of Caffe framework [27,28], we first analyze factors that influence the computation delay at each layer, such as input size and kernel size. Then, we obtain the influential pattern of each factor by testing how each factor is related to the computation delay at this layer. By exploring these relationships, we can not only select the influential factors, but also ignore the useless factors for simplicity. Based on the influential factors and their influential patterns, we develop the delay prediction model by polynomial fitting [29].

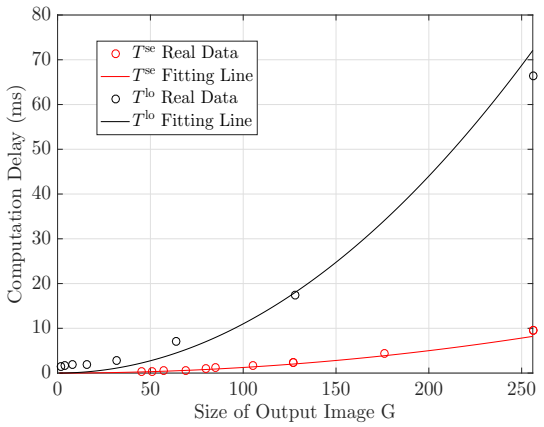


Fig. 2. Influential patterns of size of output image G on T^{se} and T^{lo} under AlexNet with $K = 5$, $O = 3$, and $I = 3$.

To illustrate this, Fig. 2 considers that the type of DNN is AlexNet, the convolution kernel size is 5, and the numbers of color channels in the input and output image are both 3. First, we analyze the factors that affect the computation delay of convolutional layer under AlexNet. With reference to the source code of Caffe, we obtain four influential factors, i.e., the size of output image G, the convolution kernel size K, the number of color channels in the input image O, and the number of color channels in the output image I. Second, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

fix the values of K, O, and I, and employ AlexNet to compute a DNN task locally under different values of G. Specifically, we use Orange Pi Win Plus as the MD. We repeat the experiment over 1000 times, and record the values of G and local computation delay T^{lo} under convolutional layer. Third, similar to the experiments on the MD, we use the same DNN task on the server, and record a sufficient number of values of G and T^{se} under convolutional layer. Specifically, we use a single computer with CPU i5, 4G RAM, 3.2GHz CPU clock speed as the server.

Based on the experimental data, we plot (G, T^{lo}) and (G, T^{se}) with black circles and red circles in Fig. 2 respectively. From the scatter plot, we observe that the relationships between either G and T^{lo} or G and T^{se} are approximately quadratic, denoted by

$$T^{\mathrm{lo}} \propto G^2, \tag{1}$$

$$T^{\mathrm{se}} \propto G^2, \tag{2}$$

where T^{lo} denotes the local computation delay, T^{se} denotes the server computation delay, and $\propto$ means "in proportion to". Based on this observation, we conduct polynomial regression to the circles of (G, T^{lo}) with (1) and (G, T^{se}) with (2), respectively. The black and red fitting lines demonstrate the quadratic influential pattern of G on T^{lo} and T^{se}, respectively.

Similarly, we obtain the influential patterns of other factors as

$$T^{\mathrm{lo}} \propto K^2,\ T^{\mathrm{lo}} \propto O,\ T^{\mathrm{lo}} \propto I, \tag{3}$$

$$T^{\mathrm{se}} \propto K^2,\ T^{\mathrm{se}} \propto O,\ T^{\mathrm{se}} \propto I. \tag{4}$$

Based on (1) and (3), we obtain influential patterns of G, K, O and I for local computation delay T^{lo}. Meanwhile, we obtain test data of G, K, O, I and corresponding T^{lo}. According to multiple regression theory [30], the polynomial fitting model of T^{lo} is given by

$$\begin{aligned} T^{\mathrm{lo}} =& \eta_1 G^2K^2OI + \eta_2 G^2K^2I + \eta_3 G^2K^2O + \eta_4 K^2OI \\ &+ \eta_5 G^2OI + \eta_6 G^2K^2 + \eta_7 G^2O + \eta_8 G^2I \\ &+ \eta_9 K^2O + \eta_{10} K^2I + \eta_{11} OI + \eta_{12} G^2 + \eta_{13} K^2 \\ &+ \eta_{14} O + \eta_{15} I + \eta_{16}, \end{aligned} \tag{5}$$

where $\eta_i,\ i = 1, \ldots, 16$, are parameters to be determined. To determine these parameters, we apply the method of least squares and maximum likelihood estimation to test data. As a result, we obtain the polynomial fitting model of T^{lo} as

$$T^{\mathrm{lo}} = (0.3G^2K^2OI + 2.6G^2K^2I + 4.8G^2O) \times 10^{-5}. \tag{6}$$

Similarly, the polynomial fitting model of T^{se} is given by

$$T^{\mathrm{se}} = (3G^2K^2OI + 7.4G^2K^2I + 25.8G^2O) \times 10^{-7}. \tag{7}$$

Then, we use the data in the test set to evaluate the fitting degree and the prediction accuracy of the prediction model in (6) and (7). The experimental results show that the R-square in the test set is 99.58%, and the average absolute error is 2.78 milliseconds[1].

Uploading delay: According to [31], the uploading delay of output data from the DNN layer m of the MD n is given by

$$T^{\mathrm{up}}_{n,m} = \frac{O_{n,m}}{R^{\mathrm{up}}_n}, \tag{8}$$

where $O_{n,m}$ denotes the size of output data from the DNN layer m of the MD n, and R^{up}_n denotes the uploading rate from the MD n to the MEC server. With reference to [32], the uploading rate from the MD n to the MEC server is given by

$$R^{\mathrm{up}}_n = W_n \log_2 \left(1 + \frac{P^{\mathrm{up}}_n G_n}{W_n N_0}\right),\ n \in \mathcal{N}, \tag{9}$$

where W_n and G_n are the bandwidth and the channel power gain between the MD n and the server, respectively; P^{up}_n is the uploading power of the MD n; N_0 denotes the power spectral density of noise.

Downloading delay: The downloading delay of output data from the server to the MD n is given by

$$T^{\mathrm{do}}_n = \frac{O_{n,M_n}}{R^{\mathrm{do}}_n}, \tag{10}$$

where O_{n,M_n} denotes the size of output data from last DNN layer M_n of the MD n. Moreover, the transmission rate from the server to MD n is given by

$$R^{\mathrm{do}}_n = W_n \log_2 \left(1 + \frac{P_s G_n}{W_n N_0}\right),\ n \in \mathcal{N}, \tag{11}$$

where P_s is the transmit power of the server.

2.2.2 Energy Consumption

At each MD, energy consumption is caused by local computation and data uploading. Consider the DNN layer m at the MD n. First, the energy consumed by local computation of the layer m at the MD n is

$$E^{\mathrm{lo}}_{n,m} = P^{\mathrm{lo}}_n T^{\mathrm{lo}}_{n,m}, \tag{12}$$

where P^{lo}_n is the computing power of the MD n, and the local computation delay of the layer m at the MD n, i.e., $T^{\mathrm{lo}}_{n,m}$, is obtained by the proposed prediction model in Section 2.2.1.

Second, the energy consumed by uploading the output data of the layer m at the MD n to the server is

$$E^{\mathrm{up}}_{n,m} = P^{\mathrm{up}}_n T^{\mathrm{up}}_{n,m}, \tag{13}$$

where the uploading delay of the layer m at the MD n, i.e., $T^{\mathrm{up}}_{n,m}$, is given in (8).

1. We define the size of test data set as D. (1) R-square $= 1 - \sum_{i=1}^{D}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{D}(y_i - \bar{y}_i)^2$, where y_i is the real data (e.g., real data in Fig. 2), $\hat{y}_i$ is the fitting data (e.g., the fitting line in Fig. 2), and $\bar{y}_i$ is the mean value of the real data. Large R-square means that the prediction model has a high fitting degree of the real data. (2) Average absolute error $= \frac{1}{D}\sum_{i}^{D} |y_i - \hat{y}_i|$. Small average absolute error corresponds to a high prediction accuracy of the model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

3 Slot Model and Dynamic Pricing Strategy

In this section, we design a slot model and a dynamic pricing strategy to schedule the offloaded subtasks in the MEC server.

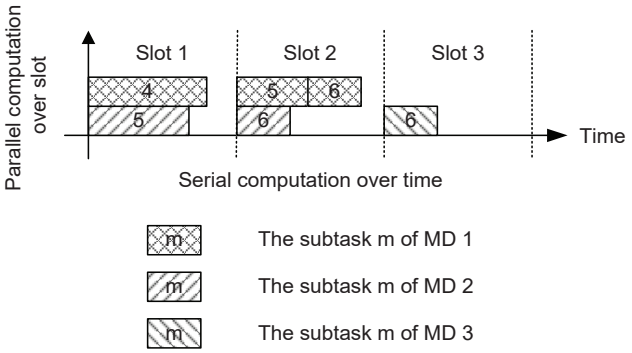


Fig. 3. An example of subtask scheduling in a 3-slot model.

3.1 Slot Model at the MEC Server

The MEC server computes the subtasks offloaded from the MDs within a certain duration T_d. Notably, T_d is much smaller than the local computation delay at each MD due to higher computing capability of the server. This work considers that the server has parallel computing capability.

To schedule the computing order of offloaded subtasks properly, we propose a slot model at the MEC server to equally divide T_d into H time slots each with a fixed duration T_d/H. Let $\mathcal{H} = \{1, \cdots, H\}$ denote a set that collects the indices of slots. Note that each offloaded subtask is scheduled to a slot according to the following rules:

- Rule 1: Subtasks from different MDs can be computed either in parallel in the same slot or sequentially over different time slots;
- Rule 2: Subtasks from the same MD must be computed sequentially even within the same slot.

Rule 1 holds since the server has parallel computing capability to compute independent subtasks from different MDs. Rule 2 is true, since, for the DNN at the same MD, the output of a subtask in a layer is the input of the subtask in the next layer. Furthermore, suppose that the duration of each slot is sufficiently long to sequentially compute several subtasks in the consecutive layers. For illustration, Fig. 3 shows a 3-slot model, where each shadowed block represents a subtask with a different size. By Rule 1, subtasks of the MDs 1 and 2 are computed in parallel in slot 1, or sequentially over slots 1 and 2. By Rule 2, subtasks of the MD 2 are computed sequentially in slots 1 and 2, and subtasks of the MD 1 are computed sequentially in slot 2.

Based on the two rules, we characterize the partitioning and offloading strategies of all MDs. First, in view of various types of DNN tasks at different MDs, we define $M_{max} = \max\{M_1, \ldots, M_N\}$ as the maximal number of DNN layers among different DNNs. Then, we use an $N \times M_{max} \times (H+1)$ matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$ to characterize the partitioning and offloading strategies of all MDs, where the n-th submatrix

$$\mathbf{y}_n = \begin{bmatrix} y_{n,1,0} & y_{n,1,1} & \cdots & y_{n,1,H} \\ y_{n,2,0} & y_{n,2,1} & \cdots & y_{n,2,H} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,M_{max},0} & y_{n,M_{max},1} & \cdots & y_{n,M_{max},H} \end{bmatrix}, \ n \in \mathcal{N}, \tag{14}$$

corresponds to the partitioning and offloading strategy of the MD n with $y_{n,m,h} \in \{0,1\}$ being the computing state of the subtask m of the MD n, $m \in \{1, \ldots, M_{max}\}$ and $h \in \{0\} \cup \mathcal{H}$. Concretely, we have two cases as follows:

- For $m \in \mathcal{M}_n$: if the subtask m of the MD n is computed locally, $y_{n,m,0} = 1$ and $y_{n,m,1} = \cdots = y_{n,m,H} = 0$; if the subtask m of the MD n is computed in the slot h at the server, $y_{n,m,h} = 1$ and $y_{n,m,h'} = 0, \ \forall h' \in \{0\} \cup \mathcal{H}, \ h' \neq h$.
- For $m \in \{M_n + 1, \ldots, M_{max}\}$: we set $y_{n,m,h} = 0, \ \forall h \in \{0\} \cup \mathcal{H}$.

Thus, we have the following constraint for $\mathbf{y}_n$:

$$\sum_{h=0}^{H} y_{n,m,h} = 1, \ \forall m \in \mathcal{M}_n, \ \forall n \in \mathcal{N}, \tag{15}$$

which means that the subtask m of the MD n can be either computed locally at the MD n or computed in a slot at the server.

From Rule 2, $\mathbf{y}_n$ is also constrained by

$$h \leq h', \quad \text{if } y_{n,m,h} = y_{n,m',h'} = 1 \text{ and } m \leq m', \text{ where } m, m' \in \mathcal{M}_n, \tag{16}$$

which means that subtasks from the same MD must be computed sequentially no matter if they are in the same slot.

$$\sum_{n=1}^{N} \sum_{m=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\text{se}} \leq \frac{T_d}{H}, \ \forall h \in \mathcal{H}, \tag{17}$$

which means that the computation delay of subtasks in the same slot should be no less than the duration of the slot.

Notably, $y_{n,m,0} = 1$ means the subtask m of the MD n is computed locally. Thus, the partition point of the DNN task from the MD n is given by

$$m^* = \sum_{m=1}^{M_{max}} y_{n,m,0}. \tag{18}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

3.2 Pricing Strategy

In practice, the MDs demand the fast computation, resulting in the crowded subtasks in the first few slots. To make efficient use of computing resources in the server, we develop a dynamic pricing strategy to not only dynamically adjust the computation price of each slot, but also encourage the MDs to offload their subtasks to less crowded slots.

In this paper, the unit computation is defined as the computation by a single CPU cycle. Let p_h denote the unit price per cycle at slot $h \in \mathcal{H}$. Consider that p_h depends on the computation of subtasks scheduled to slot h and the slot index h. Generally, p_h increases as total size of subtasks in slot h goes large [33], and decreases as the index h rises. The motivation is that, higher p_h will push the MD with limited budget towards slots with lower unit computation price, if more subtasks are crowded in some slots. However, if an MD offloads its subtasks to slots with larger h, the subtasks will be processed with larger delay. Thus, we set a lower p_h to compensate for a larger processing delay. In addition, the MDs with the delay-sensitive subtasks choose slots with smaller h even though p_h is higher. According to the above analysis, the unit computation price of the slot h is given by [33]

$$p_h = a_h + \frac{b_h}{h} S_\Sigma^h,\ h \in \mathcal{H}, \tag{19}$$

where a_h denotes the fixed unit computation price that reflects the real value of slot h; b_h denotes the growth rate of the unit price caused by the increase of $\frac{1}{h} S_\Sigma^h$; $S_\Sigma^h = \sum_{n=1}^{N} \sum_{m=1}^{M_{max}} y_{n,m,h} S_{n,m}$ denotes the number of CPU cycles to compute all subtasks in the slot h with

$$S_{n,m} = \delta Q_{m,n}, \tag{20}$$

where δ is the number of CPU cycles consumed by computing every single bit [25], and the size of the subtask m of the MD n is [34]

$$Q_{n,m} = f_s T_{n,m}^{\mathrm{se}}, \tag{21}$$

where f_s denotes the computing frequency that indicates the computing ability of the server, and $T_{n,m}^s$ is obtained by the delay prediction model in Section 2.2.1. From (20) and (21), we have

$$S_\Sigma^h = \delta f_s T_\Sigma^h, \tag{22}$$

where $T_\Sigma^h = \sum_{n=1}^{N} \sum_{m=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\mathrm{se}}$.

4 Aggregative Game Based Task Partitioning and Offloading Scheme

In this section, we formulate the optimization problem to minimize each MD's cost, and then solve this optimization by two distributed algorithms based on an aggregative game.

4.1 Optimization Problem

Under the proposed task partitioning and offloading strategy in (14), the total cost of each MD is given by

$$C_n(\mathbf{y}_n, \mathbf{Y}) = \alpha_n C_n^{\mathrm{T}} + \beta_n C_n^{\mathrm{E}} + \gamma_n C_n^{\mathrm{P}},\ n \in \mathcal{N}, \tag{23}$$

where $C_n^{\mathrm{T}} = c_n^{\mathrm{T}} T_n$ denotes the cost induced by the processing delay of the MD n's task, $C_n^{\mathrm{E}} = c_n^{\mathrm{E}} E_n$ denotes the cost induced by the energy consumption at the MD n, and $C_n^{\mathrm{P}} = \sum_{h=1}^{H} p_h \left(\sum_{m=1}^{M_{max}} S_{n,m} y_{n,m,h} \right)$ denotes the price paid by the MD n to the server. We convert processing delay and energy consumption into processing delay cost and energy consumption cost by c_n^{T} and c_n^{E} respectively. Moreover, in the context of different task requirements, we employ different weights α_n, β_n, γ_n to adjust the impact of delay, energy, and price on the total cost of each MD respectively with $\alpha_n + \beta_n + \gamma_n = 1$. For example, α_n and β_n are relatively large, if the MD demands delay-sensitive and energy-restricted task offloading.

In this paper, each MD optimizes its partitioning and offloading strategy in (14) to minimize its cost in (23). Recall that the processing delay for each MD is composed of local computation delay, uploading delay, server computation delay, and downloading delay, given by

$$T_n = \sum_{m=1}^{m^*} T_{n,m}^{\mathrm{lo}} y_{n,m,0} + T_{n,m^*}^{\mathrm{up}} + \frac{h_{n,max}}{H} T_d + T_n^{\mathrm{do}}, \tag{24}$$

where $h_{n,max}$ denotes the largest slot index such that $y_{n,M_{max},h_{n,max}} = 1$. Recall that m^* is the partition point of the DNN task from the MD n defined in (18). Thus, T_{n,m^*}^{up} is the uploading delay of the output data from the partition point at the MD n, which can be obtained from (8).

From Section 2.2.2, the energy consumption of each MD is composed of local computation energy consumption and uploading energy consumption, given by

$$E_n = \sum_{m=1}^{m^*} E_{n,m}^{\mathrm{lo}} y_{n,m,0} + E_{n,m^*}^{\mathrm{up}}, \tag{25}$$

where E_{n,m^*}^{up} denotes the energy consumption consumed by uploading the output data from the partition point at the MD n (see (13)).

Now let us further characterize the constraints for the delay and parallel computation respectively. Give (24), the delay constraint is given by

$$T_n \le \tau_n,\ \forall n \in \mathcal{N}, \tag{26}$$

where τ_n is defined as the tolerable processing delay of MD n's task. The constraint means that each task should be accomplished within its tolerable processing delay.

Considering that subtasks from different MDs can be computed in parallel in the same slot, we assume that at most B subtasks can be computed in parallel within each slot. As such, the parallel computing constraint is

given by

$$\sum_{n=1}^{N} \mathbf{1}\left(\sum_{m=1}^{M_{max}} y_{n,m,h}\right) \leq B,\ \forall h \in \mathcal{H}, \tag{27}$$

where $\mathbf{1}(\cdot)$ is an indicative function, i.e., $\mathbf{1}(x) = 1$, if $x \neq 0$, and $\mathbf{1}(x) = 0$, if $x = 0$.

Given the objective in (23) and constraints in (15), (16), (17), (26), and (27), the optimization problem is formulated as

$$\min_{\mathbf{y}_n}\ C_n(\mathbf{y}_n, \mathbf{Y}) \tag{28a}$$

$$\text{s.t. } (15), (16), (17), (26), (27). \tag{28b}$$

It is observed that, except for $\mathbf{y}_n$, only T_Σ^h in (22) is unknown at the MD n, as T_Σ^h is coupled with the aggregative strategies of all MDs. Thus, the optimization problem in (28) can be treated as an aggregative game.

In the MEC networks, each MD can obtain $T_\Sigma^h,\ h \in \mathcal{H}$ by either feedback from the MEC server or estimation. The former way is a distributed method with certain aggregative information, while the later is a distributed method with uncertain aggregative information.

4.2 Distributed Task Partitioning and Offloading Strategy with Certain Aggregative Information

This subsection considers that each MD has perfect knowledge of T_Σ^h through the feedback from the MEC server.

In practice, for energy saving purpose, each MD is active intermittently to upload and download data. We assume that each MD participating in the aggregative game has a local clock to count the number of individual iterations. At each tick of the clock, the MD wakes up according to a given probability. When the MD is active, it receives the aggregative information from the MEC server, and updates its optimal strategy.

As shown in Algorithm 1, we propose a distributed task partitioning and offloading strategy with certain aggregative information (TPOS-CAI) to minimize the total cost of each MD.

4.3 Distributed Task Partitioning and Offloading Strategy with Uncertain Aggregative Information

In practice, it is challenging for each MD to obtain certain aggregative information from the server. To address this issue, we propose to estimate the aggregative information T_Σ^h for each MD based on the strategies of its neighbors. We define a set of the MD n's neighbors as $\mathcal{N}_n$. Only the active MDs can exchange information with their neighbors. Specifically, each active MD estimates T_Σ^h as

$$\hat{T}_\Sigma^h = \min\left\{\frac{BT_d}{H}, N \times \left(w_{n,n}\left(\sum_{j=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\text{se}}\right) + \sum_{q\in\mathcal{N}_n} w_{n,q}\left(\sum_{m=1}^{M_{max}} y_{q,m,h} T_{q,m}^{\text{se}}\right)\right)\right\}, \tag{29}$$

Algorithm 1 TPOS-CAI

Input: ε.
Output: $\mathbf{Y}$.
Steps:

1: Initialization:
2: Set a local clock tick to denote the iteration index of MD n: $k_n = 0,\ n \in \mathcal{N}$.
3: MD n chooses a random partitioning and offloading strategy $\mathbf{y}_n$, that satisfies constraints (28b).
4: Set the total cost of MD n as $C_n^{(k_n)} = +\infty$.
5: $k_n = k_n + 1$.
6: MD n wakes up according to a given probability.
7: **If** MD n is not active **Then**
8: Back to Step 5.
9: **Else**
10: MD n obtains T_Σ^h through the feedback from the server, and updates its optimal strategy by solving (28).
11: Compute the average cost $\overline{C_n^{(k_n)}} = \frac{1}{N}\sum_{n=1}^{N} C_n^{(k_n)}$.
12: **If** $|\overline{C_n^{(k_n)}} - \overline{C_n^{(k_n-1)}}| > \varepsilon$ **Then**
13: Back to Step 5.
14: **Else**
15: Output $\mathbf{Y}$ as the optimal strategies.
16: **End If**
17: **End If**

where $\sum_{j=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\text{se}}$ is the computation delay of subtasks from MD n in the slot h, and $\sum_{m=1}^{M_{max}} y_{q,m,h} T_{q,m}^{\text{se}}$ is the computation delay of subtasks from MD q in the slot h. Note that MD q belongs to the collection of MD n's neighbors $\mathcal{N}_n$. Moreover, $w_{n,n}$ is the nonnegative weight that MD n assigns to its own delay, $w_{n,q}$ is the nonnegative weight that MD n assigns to MD q's delay, and we set $w_{n,n} + \sum_{q\in\mathcal{N}_n} w_{n,q} = 1$. In this context, $w_{n,n}\left(\sum_{j=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\text{se}}\right) + \sum_{q\in\mathcal{N}_n} w_{n,q}\left(\sum_{m=1}^{M_{max}} y_{q,m,h} T_{q,m}^{\text{se}}\right)$ denotes the estimated average computation delay of subtasks in slot h, and $N \times \left(w_{n,n} \times \left(\sum_{j=1}^{M_{max}} y_{n,m,h} T_{n,m}^{\text{se}}\right) + \sum_{q\in\mathcal{N}_n} w_{n,q} \times \left(\sum_{m=1}^{M_{max}} y_{q,m,h} T_{q,m}^{\text{se}}\right)\right)$ denotes the estimated total computation delay of subtasks in slot h. To avoid the case that the estimated total computation delay exceeds the maximum computation delay in slot h (i.e., $\frac{BT_d}{H}$), the estimated total computation delay is given by (29).

5 Numerical Results

In this section, we evaluate the performance of the proposed algorithms. The evaluation system consists of $N = 5$ MDs and one MEC server. The Orange Pi Win Plus acts as the MD, and one computer with CPU i5, 4G RAM, 3.4GHz CPU is used as the MEC server. The MDs and the server communicate over WiFi. In the evaluation, we adopt 4 types of DNNs with different computational complexities, i.e., VGG16, VGG13, ALEXNET, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

Algorithm 2 TPOS-UAI

Input: ε.
Output: $\mathbf{Y}$.
Steps:

1: Initialization:
2: Set a local clock tick to denote the iteration index of MD n: $k_n = 0,\ n \in \mathcal{N}$.
3: MD n chooses a random partitioning and offloading strategy $\mathbf{y}_n$, that satisfies constraints (28b).
4: Set the total cost of MD n as $C_n^{(k_n)} = +\infty$.
5: $k_n = k_n + 1$.
6: MD n wakes up according to a given probability.
7: **If** MD n is not active **Then**
8: Back to Step 5.
9: **Else**
10: MD n communicates with its active neighbors, generates its estimated $\hat{T}_\Sigma^h$ according to the equation (29), and updates its optimal strategy by solving the problem (28).
11: Compute the average cost $\overline{C_n^{(k_n)}} = \frac{1}{N}\sum_{n=1}^{N} C_n^{(k_n)}$.
12: **If** $|\overline{C_n^{(k_n)}} - \overline{C_n^{(k_n-1)}}| > \varepsilon$ **Then**
13: Back to Step 5.
14: **Else**
15: Output $\mathbf{Y}$ as the optimal strategies.
16: **End If**
17: **End If**

Parameters	Symbols	Values
Real value of slot *h*	a_h	1
Growth rate of the unit computing price	b_h	2
Parallel limit of the server	B	4
Maximum tolerable processing delay of MD *n*'s task	τ_n	5s
Conversion factor from processing delay into processing delay cost	c_n^T	3400
Conversion factor from energy consumption into energy consumption cost	c_n^E	3400
Number of CPU cycles consumed by computing subtasks per bit	δ	[2000,2500]
Computing frequency of the server	f_s	3400cycles/s
Uploading power of the MD *n*	P_n^{up}	4W
Computing power of the MD *n*	P_n^{lo}	4.05W
Bandwidth between the MD *n* and the server	W_n	2KHz
Power spectral density of noise	N_0	−30dbm/KHz
Channel power gain between the MD *n* and the server	G_n	1.2×10^6
Number of time slots	H	5
Total duration of slots	T_d	0.75s

Fig. 4. Parameters used in the simulations.

LENET, where the computational complexities decrease from VGG16 to LENET. The DNN with more layers and more complex structure has higher computational complexities, leading to larger processing delay and energy consumption. Fig. 4 lists the key parameters used in this section.

Fig. 5 depicts the convergence of Algorithms 1 and 2 with different numbers of MDs. Consider that each MD employs the ALEXNET with 21 layers. Let us define the average cost as $\overline{C}_n = \frac{1}{N}\sum_{n=1}^{N} C_n(\mathbf{y}_n, \mathbf{Y})$. First, it is observed that the convergence speed of each algorithm drops sharply as N increases. Second, Algorithms 1 and 2 converge to the same $\overline{C}_n$ with similar convergence speed. This result implies that Algorithm 2 with uncertain aggregative information can achieve the same performance as Algorithm 1 with certain aggregative information. Based on the second observation, the following simulations only consider Algorithm 2.

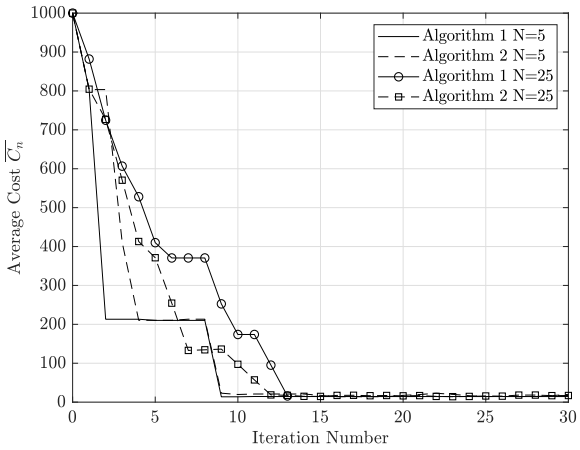


Fig. 5. Convergence of Algorithms 1 and 2 with $\alpha_n = 0.42,\ \beta_n = 0.17$, and $\gamma_n = 0.41$, where the DNN type is ALEXNET with 21 layers.

We use a global optimal task partitioning and offloading scheme (GTPOS) as a baseline for our proposed scheme in Algorithm 2. In this scheme, the server acts as a central controller that has global knowledge of each MD's strategy. To minimize average processing delay and efficiently use computing resources, the server globally optimizes the task partitioning and offloading strategies of all MDs. In this context, there is no need for the global optimal scheme to adopt the slot model and pricing strategy in Section 3. For the global scheme, each offloaded subtask is computed at the server according to the following rules:

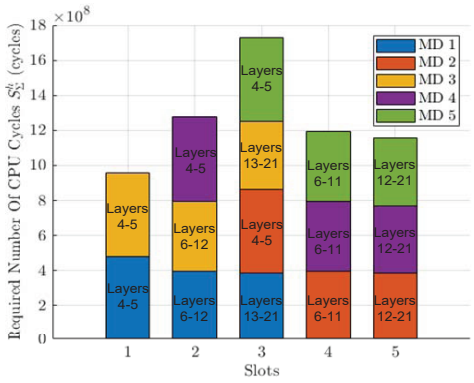
- Rule 3: Subtasks from different MDs are computed either in parallel or sequentially, if the number of subtasks being computed in parallel exceeds the parallel limit of the server;
- Rule 4: Subtasks from the same MD must be computed sequentially.

For simplicity, we refer to TPOS-UAI in Algorithm 2 as **Case 1**, GTPOS as **Case 2**, and

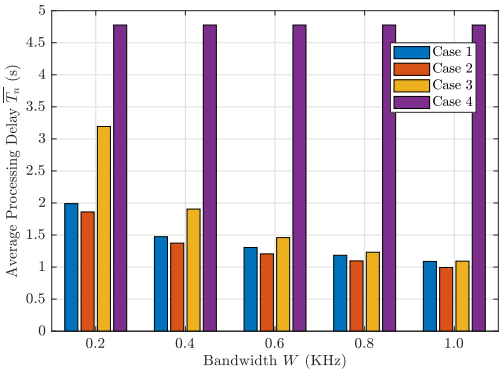
- **Case 3:** Each MD randomly offloads its whole task to the server.
- **Case 4:** Each MD computes its whole task locally.

Fig. 6 shows the proposed scheduling model of Case 2 and compares the average processing delay of Cases 1-4 under different communication bandwidths. Fig. 6

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing



(a) The optimal partitioning and offloading strategies of all MDs under Algorithm 2 with $\alpha_n = 0.95$, $\beta_n = 0.049$, and $\gamma_n = 0.001$, where the DNN type is ALEXNET with 21 layers and the bandwidth is 0.2KHz.

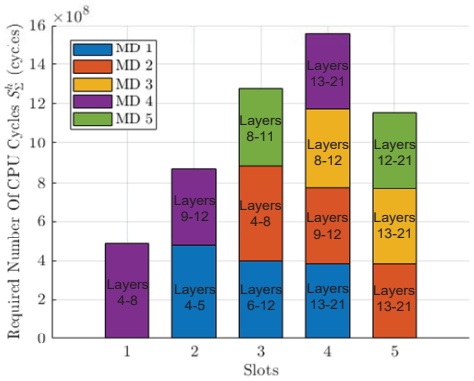


(b) The average processing delay $\overline{T_n}$ under different bandwidths.

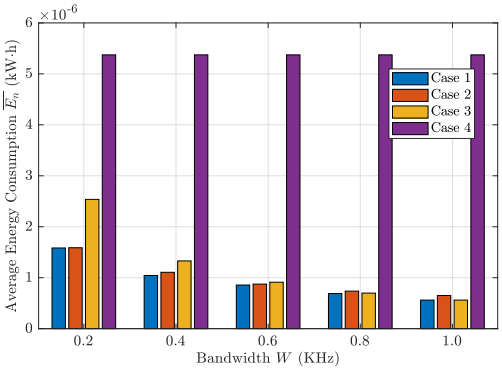
Fig. 6. The average processing delay comparison among Cases 1-4.

(a) presents subtask scheduling model under the optimal task partitioning and offloading strategy in Algorithm 2. Consider that the subtasks from 5 MDs are scheduled to the server with 5 slots, and each MD employs ALEXNET with 21 layers. The scheduling results in Fig. 6 (a) are consistent with Rules 1 and 2. Let us define the average processing delay as $\overline{T_n} = \frac{1}{N}T_n$. Consider that the bandwidth between each MD and the MEC server is identical, i.e., $W_1 = \ldots = W_N = W$. In Fig. 6 (b), we first observe that Case 1 can achieve the smaller $\overline{T_n}$ than both Cases 3 and 4. It is stressed that, in spite of the minimum $\overline{T_n}$, Case 2 is not practical since it is unlikely to have a central controller that has global knowledge of each MD's strategy. Second, we observe that, $\overline{T_n}$ under Cases 1, 2, and 3 decrease as W increases, while $\overline{T_n}$ under Case 4 remains unchanged as W increases. This is due to the fact that, the uploading delays under Cases 1, 2, and 3 decrease with the growth of W, while the uploding delay under Case 4 is always equal to zero regardless of the value of W. Third, the reduction of $\overline{T_n}$ under Case 3 is more significant than that under Cases 1 and 2. The reason is that, in Case 3, each MD offloads all its DNN task to the cloud, leading to a relative large uploading delay. However, in Cases 1 and 2, each MD only offloads a part of its task to the MEC server, leading to relatively small uploading delay.

Fig. 7 shows the proposed scheduling model of Case 2 and compares the average processing energy consumption of Cases 1-4 under different communication bandwidths. Fig. 7 (a) presents subtask scheduling model under the optimal task partitioning and offloading strategy in Algorithm 2. The scheduling results in Fig. 7 (a) are consistent with Rules 1 and 2. Let us define the average energy consumption as $\overline{E_n} = \frac{1}{N}E_n$. First, it is observed that Case 1 can achieve the minimum $\overline{E_n}$ compared with other cases. This is because that, Case 2 does not consider energy consumption, Case 3 has high uploading energy consumption due to large uploading delay, and Case 4 has high local computation energy consumption due to large computation delay. Second, we observe that $\overline{E_n}$ under Cases 1, 2, and 3 decrease as W increases, while $\overline{E_n}$ under Case 4 remains unchanged as W increases.



(a) The optimal partitioning and offloading strategies of all MDs under Algorithm 2 with $\alpha_n = 0.1$, $\beta_n = 0.75$, and $\gamma_n = 0.15$, where the DNN type is ALEXNET with 21 layers and the bandwidth is 0.4KHz.



(b) The average energy consumption $\overline{E_n}$ under different bandwidths.

Fig. 7. The average energy consumption comparison among Cases 1-4.

We use Figs. 8 and 9 to verify the the scalability of the proposed TPOS-UAI to different types of DNNs.

Fig. 8 illustrates the relationship between average processing delay $\overline{T_n}$ and different types of DNNs under Cases 1-4. First, we observe that Case 1 can approach

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3114193, IEEE Transactions on Mobile Computing

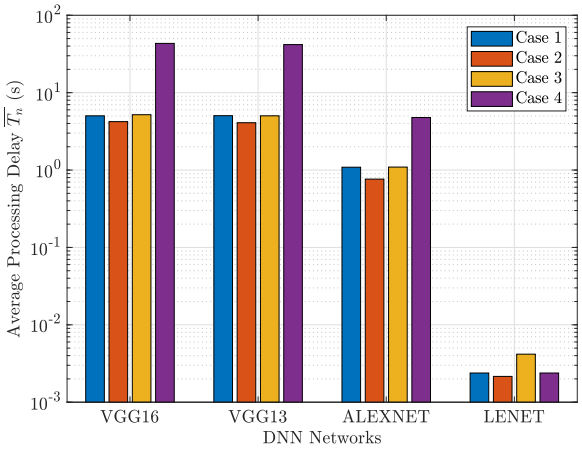


Fig. 8. The relationships between average processing delay $\overline{T_n}$ and different types of DNNs with $W_n = 1$KHz, $\alpha_n = 0.95, \beta_n = 0.049$, and $\gamma_n = 0.001$, $n \in \mathcal{N}$.

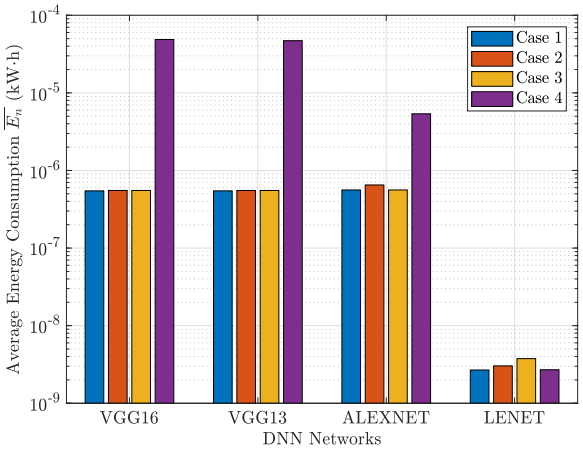
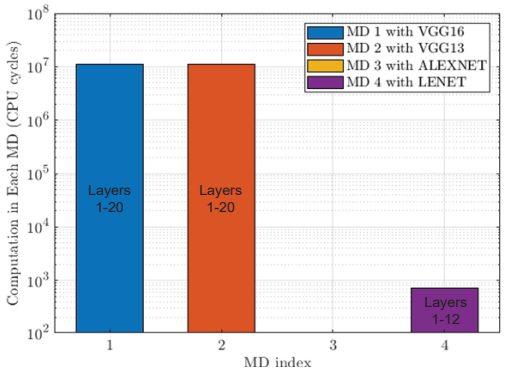


Fig. 9. The relationships between average energy consumption $\overline{E_n}$ and different types of DNNs with $W_n = 1$KHz, $\alpha_n = 0.049, \beta_n = 0.95$, and $\gamma_n = 0.001$, $n \in \mathcal{N}$.

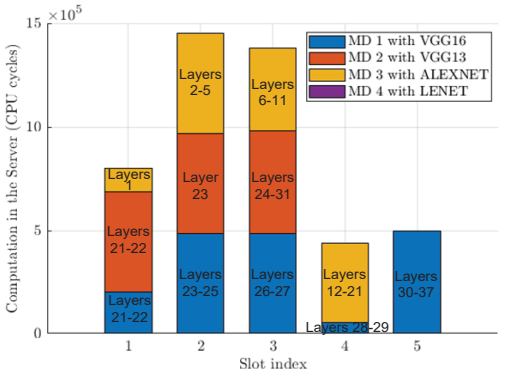
the optimal $\overline{T}_n$ of Case 2 regardless of the DNN types. Second, it is observed that, Case 2 achieves the smallest $\overline{T}_n$ among the four cases. This is due to the fact that, for DNN tasks with relative high computational complexity (i.e., VGG16, VGG13, and ALEXNET), Case 4 is rather time-consuming due to limited computing capability of MDs. However, for DNN tasks with relative low computational complexity (i.e. LENET), Case 3 becomes slow due to large uploading delay. Third, $\overline{T}_n$ under Cases 1-4 decreases as the computational complexity of DNN decreases.

Fig. 9 illustrates the relationship between average energy consumption $\overline{E_n}$ and different types of DNNs under Cases 1-4. First, it is observed that Case 1 can achieve the smallest $\overline{E}_n$ compared with other cases regardless of the DNN types. This is due to the fact that, for DNN tasks with relative high computational complexity (i.e., VGG16, VGG13, and ALEXNET), Case 4 has high local computation energy consumption due to large computation delay. However, for DNN tasks with relative low computational complexity (i.e. LENET), Case 3 has high uploading energy consumption due to large uploading delay. Case 2 is energy-consuming since it does not take energy consumption into account. Second, $\overline{E}_n$ under Cases 1-4 decreases as the computational complexity of DNN decreases.

Fig. 10 shows the optimal partitioning and offloading strategies of MDs with different DNN tasks under Algorithm 2. Fig. 10 (a) presents the local processing decision of each MD. For example, the MD 2 with VGG13 decides the local computation of Layers 1-20 and the task offloading of Layers 21-31. Fig. 10 (b) depicts the scheduling results of subtasks offloaded from each MD to the server. For example, the MD 2 with VGG13 offloads the computation of Layers 21-22 to Slot 1, Layer 23 to Slot 2, and Layers 24-31 to Slot 3, respectively. It is observed that the MDs with complex DNN tasks (i.e., VGG16, VGG13, and ALEXNET) prefer to offload their



(a) The optimal strategies in each MD.



(b) The optimal strategies in the server.

Fig. 10. The optimal strategies of MDs with different DNN tasks under Algorithm 2, where $\alpha_n = 0.95$, $\beta = 0.049$, and $\gamma = 0.001$.

subtasks for powerful computing capability, while the MDs with simple DNN tasks (i.e., LENET) prefer to local execution to save their cost.

6 Conclusion

In this paper, we proposed a general DNN-task partitioning and offloading scheme in the multi-user MEC

network. First, we used the layer-level computation partitioning method to partition each MD's DNN task into subtasks that can be either computed locally or offloaded to the server. Second, we obtained a delay prediction model for the computation delay of each subtask, and further analyzed the energy consumption of each subtask. Third, towards an efficient task execution, we proposed a joint design of slot model and dynamic pricing strategy for the server. Fourth, we designed the optimal DNN-task partitioning and offloading strategy by minimizing each MD's cost that includes the computation delay, energy consumption, and price paid to the server. To obtain the optimal strategy, we designed two distributed algorithms based on the aggregative game theory. Numerical results demonstrated that, the proposed scheme was scalable to different types of DNNs, and could reduce processing delay and energy consumption compared with the baseline schemes.

Several interesting directions follow this work. First, how to offload tasks from multiple MDs to multiple servers remains challenging. In this context, a matching algorithm between the MDs and servers is required to associate each MD to a best server. Second, this work adopts a layer-level computation partitioning strategy, where subtasks from the same DNN task are computed sequentially at either the MD or the server. To improve computation efficiency, it is of interest to design a different partitioning strategy, where subtasks from the same DNN task can be computed in parallel by multiple servers.

Acknowledgments

This work is supported in part by 2020 Industrial Internet Innovation and Development Project of Ministry of Industry and Information Technology of the People's Republic of China under Grant TC200A00Y, in part by National Natural Science Foundation of China under Grant 61872184.

References

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

[2] Z. Zhang, Y. Li, C. Huang, Q. Guo, C. Yuen, and Y. Guan, "DNN-aided block sparse bayesian learning for user activity detection and channel estimation in grant-free non-orthogonal random access," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12 000–12 012, Dec. 2019.

[3] H. Yu, Z. Tan, Z. Ma, R. Martin, and J. Guo, "Spoofing detection in automatic speaker verification systems using DNN classifiers and dynamic acoustic features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4633–4644, Oct. 2018.

[4] 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.-Feb. 2018.

[5] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 098–11 112, Nov. 2018.

[6] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[7] M. Gao, J. Li, D. N. K. Jayakody, H. Chen, Y. Li, and J. Shi, "A super base station architecture for future ultra-dense cellular networks: Toward low latency and high energy efficiency," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 35–41, Jun. 2018.

[8] M. Gao, R. Shen, S. Yan, J. Li, H. Guan, Y. Li, J. Shi, and Z. Han, "Heterogeneous computational resource allocation for c-ran: A contract-theoretic approach," *IEEE Trans. Services Comput. (Early Access)*, Apr. 2019.

[9] Y. Zhou, L. Liu, L. Wang, N. Hui, X. Cui, J. Wu, Y. Peng, and Y. Qi, "Service aware 6G: an intelligent and open network based on convergence of communication, computing and caching," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 253–260, Aug. 2020.

[10] Y. Zhou, L. Tian, L. Liu, and Y. Qi, "Fog computing enabled future mobile communication networks: a convergence of communication and computing," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 20–27, May 2019.

[11] Y. Qi, Y. Zhou, Y.-F. Liu, L. Liu, and Z. Pan, "Traffic-aware task offloading based on convergence of communication and sensing in vehicular edge computing," *IEEE Internet Thing J. (Early Access)*, May 2021.

[12] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[13] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Survey Tuts.*, vol. 19, no. 4, pp. 2322 – 2358, Fourthquarter 2017.

[14] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[15] 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 Internet Thing J.*, vol. 6, no. 3, pp. 4330–4343, Jun. 2019.

[16] J. Hu, M. Jiang, Q. Zhang, Q. Li, and J. Qin, "Joint optimization of uav position, time slot allocation, and computation task partition in multiuser aerial mobile-edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 7231–7235, Jul. 2019.

[17] Y. Kang, J. Hauswald, C. Gao, and A. Rovinski, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, Apr. 2017.

[18] W. He, S. Guo, S. Guo, X. Qiu, and F. Qi, "Joint dnn partition deployment and resource allocation for delay-sensitive deep learning inference in iot," *IEEE Internet Thing J.*, vol. 7, no. 10, pp. 9241–9254, Oct. 2020.

[19] Z. Kuang, Y. Shi, S. Guo, J. Dan, and B. Xiao, "Multi-user offloading game strategy in OFDMA mobile cloud computing system," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12 190–12 201, Dec. 2019.

[20] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

[21] M. Gao, R. Shen, J. Li, S. Yan, Y. Li, J. Shi, Z. Han, and L. Zhuo, "Computation offloading with instantaneous load billing for mobile edge computing," *IEEE Trans. Services Comput. (Early Access)*, May 2020.

[22] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sept. 2017.

[23] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for NOMA MEC offloading," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6207–6211, Jun. 2019.

[24] K. Xie, X. Wang, G. Xie, D. Xie, J. Cao, Y. Ji, and J. Wen, "Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 6, pp. 925–940, Nov.-Dec. 2019.

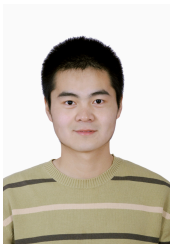
[25] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, "Two-tier matching game in small cell networks for mobile edge computing," *IEEE Trans. Services Comput. (Early Access)*, Aug. 2019.

[26] M. Gao, W. Cui, D. Gao, R. Shen, J. Li, and Y. Zhou, "Deep neural network task partitioning and offloading for mobile edge computing," in *Proc. IEEE GLOBECOM*, Waikoloa, HI, USA, Dec. 2019.

[27] M. Komar, P. Yakobchuk, V. Golovko, V. Dorosh, and A. Sachenko, "Deep neural network for image recognition based on the caffe framework," in *Proc. IEEE DSMP*, Lviv, Ukraine, Aug. 2018.
[28] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 11, Nov. 2019.
[29] H. Fu, J. Zhu, C. Wang, and R. Wang, H. Zhao, "A wavelet decomposition and polynomial fitting-based method for the estimation of time-varying residual motion error in airborne interferometric sar," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 49–59, Jan. 2018.
[30] S. Kim, C. Kim, S. Jung, and Y. Kim, "Shape optimization of a hybrid magnetic torque converter using the multiple linear regression analysis," *IEEE Trans. Magn.*, vol. 52, no. 3, Mar. 2016.
[31] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 2795–2808, Oct. 2016.
[32] R. Zhang, "Throughput maximization in wireless powered communication networks with energy saving," *IEEE Trans. Wireless Commun.*, vol. 2015-April, pp. 516–520, Apr. 2015.
[33] H. Chen, Y. Li, R. H. Y. Louie, and B. Vucetic, "Autonomous demand side management based on energy consumption scheduling and instantaneous load billing: An aggregative game approach," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1744–1754, Jul. 2014.
[34] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, pp. 2795–2808, Aug. 2017.

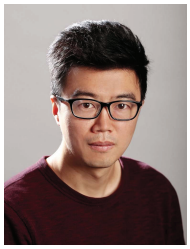
Mingjin Gao received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China in 2017. From July 2017 to now, he is an Assistant Professor in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include wireless communications, mobile edge computing, signal processing and machine learning.

Rujing Shen received her B.S. degree in Statistics from University of International Business and Economics in 2018. She is currently a Ph.D. candidate at the University of the Chinese Academy of Sciences. Her research interests include wireless communications and intelligent edge computing.



Long Shi (Member, IEEE) received the Ph.D. degree in Electrical Engineering from University of New South Wales, Sydney, Australia, in 2012. From 2013 to 2016, he was a Postdoctoral Fellow at the Institute of Network Coding, Chinese University of Hong Kong, China. From 2014 to 2017, he was a Lecturer at Nanjing University of Aeronautics and Astronautics, Nanjing, China. From 2017 to 2020, he was a Research Fellow at Singapore University of Technology and Design. Currently, he is a Professor at School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include wireless network coding, mobile edge computing, coded caching, and blockchain.

Wen Qi received her M.E. degree in Electronic and communication engineering from China University of Petroleum in 2021. Currently, she works at China Telecom Research Institute. Her research interests include mobile communications, with a particular focus on core network.



Jun Li (M'09-SM'16) received Ph. D degree in Electronic Engineering from Shanghai Jiao Tong University, Shanghai, P. R. China in 2009. From January 2009 to June 2009, he worked in the Department of Research and Innovation, Alcatel Lucent Shanghai Bell as a Research Scientist. From June 2009 to April 2012, he was a Postdoctoral Fellow at the School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia. From April 2012 to June 2015, he was a Research Fellow at the School of Electrical Engineering, the University of Sydney, Australia. From June 2015 to now, he is a Professor at the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China. He was a visiting professor at Princeton University from 2018 to 2019. His research interests include network information theory, game theory, distributed intelligence, multiple agent reinforcement learning, and their applications in ultra-dense wireless networks, mobile edge computing, network privacy and security, and industrial Internet of things. He has co-authored more than 200 papers in IEEE journals and conferences, and holds 1 US patents and more than 10 Chinese patents in these areas. He was serving as an editor of IEEE Communication Letters and TPC member for several flagship IEEE conferences. He received Exemplary Reviewer of IEEE Transactions on Communications in 2018, and best paper award from IEEE International Conference on 5G for Future Wireless Networks in 2017.

Yonghui Li (M'04-SM'09-F19) received his PhD degree in November 2002 from Beijing University of Aeronautics and Astronautics. Since 2003, he has been with the Centre of Excellence in Telecommunications, the University of Sydney, Australia. He is now a Professor and Director of Wireless Engineering Laboratory in School of Electrical and Information Engineering, University of Sydney. He is the recipient of the Australian Queen Elizabeth II Fellowship in 2008 and the Australian Future Fellowship in 2012. He is a Fellow of IEEE.

His current research interests are in the area of wireless communications, with a particular focus on MIMO, millimeter wave communications, machine to machine communications, coding techniques and cooperative communications. He holds a number of patents granted and pending in these fields. He is now an editor for IEEE transactions on communications, IEEE transactions on vehicular technology. He also served as the guest editor for several IEEE journals, such as IEEE JSAC, IEEE Communications Magazine, IEEE IoT journal, IEEE Access. He received the best paper awards from IEEE International Conference on Communications (ICC) 2014, IEEE PIRMC 2017 and IEEE Wireless Days Conferences (WD) 2014.