# Joint DNN Partitioning and Resource Allocation for Multiple Machine Learning-based Mobile Applications at the Network Edge

Cheng-Yu Cheng*, Robert Gazda†, Hang Liu*
*Dept. of Electrical Engineering & Computer Science, The Catholic University of America, Washington, DC, USA
†InterDigital Communications, Inc., Conshohocken, PA, USA
Email: {chengc, liuh}@cua.edu, robert.gazda@interdigital.com

***Abstract*—As a core technique of machine learning, deep neural networks (DNNs) have been extensively used in today's mobile applications. However, users' mobile devices (MDs) have limited capabilities to execute computation-intensive DNN inference operations and meet the latency constraints. Offloading part of DNN computations to edge servers (ESs) in a mobile edge computing (MEC) network can mitigate these challenges. However, prior studies on offloading either overlook the varying computation demands and output data sizes for different layers of neural networks or only consider split DNN offloading based on a single user or a single ES. Driven by the question of how to partition multiple parallel DNN inferences and offload multiple partitioned DNN processes in a multi-user multi-ES network, in this paper, we design a distributed scheme that jointly optimizes user-ES association, DNN layer-level partitioning, computing and wireless communication resource allocation, and offloading. Specifically, MDs and ESs make decisions to maximize their own utilities based on a multi-leader multi-follower Stackelberg game by leveraging DNN layer characteristics and taking account of multi-server heterogeneous network environments, computation load, and available resources. The evaluation results show that the proposed joint optimization scheme can significantly improve the performance of DNN inferences, compared to the commonly used benchmarks.**

***Index Terms*—Mobile Edge Computing, Deep Neural Network, Partitioning, Association, Resource Allocation, Offloading.**

## I. Introduction

Artificial intelligence (AI) and machine learning (ML) are increasingly popular, and deep neural networks (DNNs) are the core techniques for AI and ML. DNNs have been widely used in mobile applications, e.g., image/video recognition, augmented reality (AR)/virtual reality (VR), robot perception, video games, sensor data mining, and anomaly detection. Many DNN-based applications require stringent quality of service (QoS), such as low latency and high accuracy. On the other hand, DNN model execution is computation-intensive and power-consuming, and users' mobile devices (MDs) usually have certain limitations in computing capability and energy consumption. For example, AlexNet [1], VGG-16 [2] and GoogleNet [3] require 724M, 15.5G, and 1.43G multiply-add computations (MACs), respectively, for a typical image classification task. Therefore, many AI/ML applications intend to offload the DNN inference processing tasks from MDs to mobile edge computing (MEC) servers co-located at the base stations and/or remote cloud data centers. For example, interactive AR/VR processing can be conducted at a MEC edge server (ES), and then the rendered images and related data are transmitted and shown to the user on his mobile device. However, the MEC- and cloud-based DNN task offloading approach requires significant amounts of data to be sent to edge/cloud servers over wireless networks and puts heavy computational pressure on the servers under situations with many users and heavy demands. Thus, it needs to take into account the available network bandwidth, mobile device and server computation capacity, and communication and computing latency requirements.

DNN consists of multiple connected layers of neurons. There are varying computation complexities and output data sizes for different layers of neural networks. A DNN inference operation can be appropriately partitioned based on its layer structure between the MD and edge/cloud server. The MD executes the inference up to a specific DNN layer and sends the intermediate data to a server. The server runs through the remaining DNN layers, which is able to offload the computation-intensive parts to the server [4], [5]. The computation and data rate requirements may thus be reduced at the MD and the latency performance can be improved, compared to the entire execution on either device or server side.

However, the performance of split DNN operations depends on its layer-level characteristics (computation complexities and intermediate output data sizes), the DNN task demands, the computation capabilities of mobile devices and servers, as well as network environments. For example, in a video recognition application using a convolutional neural network (CNN) model, the intermediate data size transferred from one CNN layer to the next may change a lot, thus the required network bandwidth is related to the model split point as well as the video frame rate and resolution. The required computation cycles for different DNN layers and the available resources at the MDs and servers also vary and affect the processing time of split DNN operations. The challenges are to determine the optimal DNN partitioning point and offload the appropriate part of DNN computation tasks from MDs to ESs under dynamic DNN task demands and network environments, especially when there are multiple mobile users and multiple servers to perform multiple DNN operations in a MEC network.

Substantial research has been conducted on task offloading and resource allocation for MEC [6]–[8]. However, those works mainly considered general computational task offloading problems, where the whole tasks or a portion of data bits belonging to a task were offloaded from mobile devices to edge servers for processing, but the DNN layer characteristics and layer-level partitioning were not considered. There are a few recent studies that specifically address DNN layer partitioning. The mechanisms to profile the computation time and output data size of DNN layers in real-time have been proposed [9], [10]. There were a number of works to design the schemes to decide the best DNN partitioning point and computation offloading strategy for different applications, such as image/video recognition, mobile AR/VR, and robot control [11], [12]. Nevertheless, these previous works only considered DNN computation partitioning based on a single mobile device, a single DNN inference process, or a single edge/cloud server. How to partition multiple parallel DNNs inferences and allocate computing and communication resources to offload multiple partitioned DNN processes in a multi-user multi-ES network is still an open problem, and it requires a different strategy to optimize overall system performance.

In this paper, we propose a joint optimization framework of user-ES association, DNN layer-level partitioning, offloading, and computing and wireless communication resource allocation for multiple parallel DNN inference operations with multiple MDs and multiple ESs. Given the distributed nature of MEC networks, we develop a distributed solution. Specifically, we design a multi-leader multi-follower Stackelberg game to model the interplay between MDs and ESs, leveraging DNN layer characteristics and considering multi-server heterogeneous network environments, computation load, and processing power. The MDs and ESs maximize their own utilities by making distributed optimal decisions through the game to determine the DNN partition points, establish association relationships, allocate radio and computing resources, and orchestrate DNN computation tasks only based on local information. The evaluation results show that the proposed joint optimization scheme can significantly improve the performance of DNN computation tasks, compared to the benchmarks.

The rest of this paper is organized as follows. We describe the system model and analyze the computation and output data size characteristics of DNN layers in Section II. We formulate the multi-leader multi-follower Stackelberg game and design the joint optimization algorithms in Section III. The performance of our work is evaluated in Section VI, and the conclusions are summarized in Section V.

## II. System Model and DNN Layer Characteristics

As shown in Fig.1, we consider a densely deployed MEC network, which consists of a set of small-cell base stations (BS). An ES is co-located with a BS to perform edge computing with low delay. Let $\mathcal{S} = \{e_1, e_2, ..., e_s, ..., e_S\}$ denote the BSs with the co-located ESs. The MEC network serves


Fig. 1: Mobile edge computing system model for multiple partitioned DNN operations.

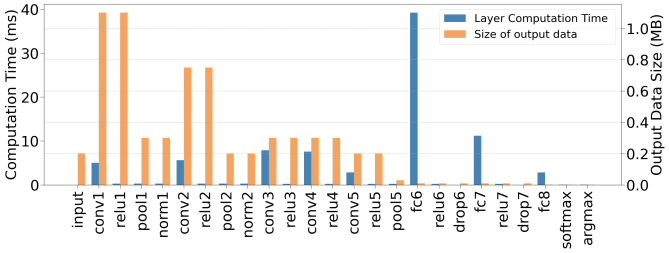



Fig. 2: The per-layer profiling for AlexNet.

a set of mobile devices, $\mathcal{U} = \{u_1, u_2, ..., u_m, ..., u_M\}$. MDs run machine learning applications with DNNs. For example, a group of visitors runs AR applications on their mobile devices to obtain related information in a city, and a popular algorithm, AlexNet is employed for object recognition. To understand the characteristics of the DNN model, we have profiled the computation time and output data size of each AlexNet layer on a Linux machine equipped with an Intel Core i7-8700 CPU at a clock speed of 3.2 GHz and 16 GB DDR3 memory. As shown in Fig.2, the AlexNet layers have significantly different computation times and output data sizes. A MD can take advantage of device and edge server synergy to properly partition the DNN inference operations into two portions, executing the first portion and offloading the remaining more computation-intensive portion to the ES for execution to improve performance.

Furthermore, we define a computing resource block (CRB) and a radio resource block (RRB) to be the unit amount of computing resources and communication resources that can be allocated to a MD by an ES, respectively. Each CRB can provide $C_0$ CPU cycles per unit time for computation, and each RRB is $B_0$ bandwidth in frequency and one unit in time for MDs to transmit data. Let ES $e_s$ charge a MD to use its CRBs and RRBs at a price of $g_s^{(c)}$ per CRB and $g_s^{(r)}$ per RRB. For ease of explanation, we let a CBR be worth the same as a RRB, that is, $g_s^{(c)} = g_s^{(r)}$. According to its DNN computation task demand, the DNN layer characteristics, the channel conditions to the different ESs, its own computing resource, and the resource prices set by the different ESs, a MD is able to choose an ES to associate with, determine the numbers of CRBs and RRBs to purchase, and decide how to partition the DNN layer-based operation between the ES and itself to maximize the overall DNN computation performance.

We first assume that a MD $u_m$ is associated with and

served by an ES $e_s$, which can be represented as

$$x_{ms} = \begin{cases} 1, & \text{MD } u_m \text{ associated to ES } e_s \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

where $x_{ms}$ is a Boolean variable that indicates whether MD $u_m$ is associated to ES $e_s$. The ES association method will be described in the next section. Generally, a DNN model $D$ contains $N$ layers, labeled as $L_1, L_2, ..., L_N$. For a layer $L_j$, $\forall j \in \{1, 2, 3, ..., N\}$, its output data size, which is also the input to the next layer, is $V_j$, and the required CPU cycles to compute layer $j$ is $F_j$. Consider that MD $u_m$ partitions the DNN model at layer $\alpha_{ms}$. When $u_m$ associates to $e_s$, $u_m$ will compute the first portion of the DNN model from layer $L_1$ to layer $L_{\alpha_{ms}}$. The intermediate output from layer $L_{\alpha_{ms}}$ will be sent to ES $e_s$, which will execute the remaining DNN operation from layer $L_{\alpha_{ms}+1}$ to layer $L_N$. The service rate of $u_m$ is defined as the number of DNN tasks per unit time when $u_m$ computes the first portion of the DNN model, which can be represented as

$$\mu_m^{(c)} = \frac{C_m}{\sum_{j=1}^{\alpha_{ms}} F_j} \quad (2)$$

where $C_m$ is the CPU cycles per unit time that can be used by $u_m$ to perform DNN computation and $\sum_{j=1}^{\alpha_{ms}} F_j$ is the total CPU cycles that are needed to compute the DNN layers from $L_1$ to $L_{\alpha_{ms}}$. In addition, the service rate for each CRB is defined as the number of DNN tasks per unit time when a CRB is used to process the remaining layers of the DNN model, which can be expressed as

$$\mu_{ms}^{(c)} = \frac{C_0}{\sum_{j=\alpha_{ms}+1}^{N} F_j} \quad (3)$$

where $C_0$ represents the CPU cycle per unit time for each CRB and $\sum_{j=\alpha_{ms}+1}^{N} F_j$ is the required CPU cycles to compute the remaining DNN layers from $L_{\alpha_{ms}+1}$ to layer $L_N$. Moreover, when $u_m$ associates to $e_s$, i.e., $x_{ms} = 1$, we can obtain the achievable data transmission rate for each RRB as [13], [14]

$$r_{ms} = B_0 \log_2(1 + SNR_{ms}) \quad (4)$$

where $SNR_{ms} = \frac{|h_{ms}|^2 P_m}{\sigma_{ms}^2}$ is the channel signal-to-noise ratio which is a function of the transmit power $P_m$, the noise power $\sigma_{ms}^2$, and the channel coefficient $h_{ms}$. The channel coefficient depends on the distance between $u_m$ and $e_s$ as well as the channel fading conditions. Given the output data size $V_{\alpha_{ms}}$ of DNN layer $L_{\alpha_{ms}}$, the number of tasks that can be sent from $u_m$ to $e_s$ per unit time, i.e., the outgoing task rate is

$$\mu_{ms}^{(r)} = \frac{r_{ms}}{V_{\alpha_{ms}}} \quad (5)$$

Real-time DNN computation tasks are delay-sensitive. Various types of delays should be taken into consideration in the design of DNN partitioning and resource allocation schemes, including roundtrip transmission and computation queuing delays. Let $\lambda_m$ denote the arrival rate of DNN tasks at MD $u_m$. Suppose that MD $u_m$ is designated to process the DNN layers from $L_1$ to $L_{\alpha_{ms}}$, and offloads the layers $L_{\alpha_{ms}+1}$ - $L_N$ to ES $e_s$ for processing. Following the commonly adopted queuing delay model in [15], [16], which can be easily extended to other models, the average sojourn time that a DNN task spends in MD $u_m$ to be processed is then expressed as

$$t_m^{(c)} = \frac{1}{\mu_m^{(c)} - \lambda_m} \quad (6)$$

In addition, if MD $u_m$ purchases $z_{ms}$ RRBs to transmit the output data of layer $L_{\alpha_{ms}}$ to ES $e_s$, the average time that a DNN task spends in waiting and transmission is

$$t_{ms}^{(r)} = \frac{1}{z_{ms}\mu_{ms}^{(r)} - \lambda_m} \quad (7)$$

Moreover, if MD $u_m$ buys $y_{ms}$ CRBs from ES $e_s$ to process the DNN layers $L_{\alpha_{ms}+1}$ - $L_N$, then the average sojourn time that a DNN task spends at ES $e_s$ to be served can be represented as

$$t_{ms}^{(c)} = \frac{1}{y_{ms}\mu_{ms}^{(c)} - \lambda_m} \quad (8)$$

Therefore, the average total service delay for a DNN task, including the time for MD $u_m$ to compute the first portion of the DNN model and send the intermediate result to ES $e_s$, and for ES $e_s$ to compute the second part of the DNN model and send the result back to $u_m$ can be expressed as

$$T_{ms} = t_m^{(c)} + \sum_{s=1}^{S} x_{ms} t_{ms}^{(r)} + \sum_{s=1}^{S} x_{ms} t_{ms}^{(c)} + \sum_{s=1}^{S} x_{ms} t_{ms}^{(b)} \quad (9)$$

where $t_{ms}^{(b)}$ is the time for $e_s$ to send the final DNN execution result back to $u_m$. We assume that $t_{ms}^{(b)}$ is a fixed value because the size of the DNN execution result, e.g., the class or name of a recognized object, is generally much smaller than the output data size of intermediate DNN layers.

## III. Joint Optimization Algorithm Design

This section describes the algorithm design for joint optimization of MD-ES association, DNN layer-level partitioning, offloading, and computing and radio resource allocation. Considering the distributed nature of MEC networks, we design distributed algorithms to allow MDs and ESs to make decisions cooperatively. Specifically, we model the interplay between MDs and ESs as a sequential decision-making process and develop a joint optimization algorithm based on the multi-leader, multi-follower Stackelberg game, in which ESs are the leaders and set the prices for CRBs and RRBs according to the MD demands and the prices set by other ESs; MDs are the followers that determine which ES to associate, how many RRBs and CRBs to purchase, and how to partition the DNN operation based on the prices set by the different ESs, the channel conditions to the different ESs, its own capability to compute DNN tasks, and the arrival rate of its DNN tasks. Each player, i.e., each of the MDs and ESs maximizes its own utility through the game.

As MDs in the network pay ESs for services, the utility of MD $u_m$ can be denoted as the reward received from the

DNN computation tasks it processes minus the incurred cost of service delay and payment to the ES $e_s$, which can be expressed as follows.

$$U_m(x_{ms}, \alpha_{ms}, y_{ms}, z_{ms}) = \gamma_m \lambda_m - \delta_m T_{ms} - \omega_m [\sum_{s=1}^{S} x_{ms}(y_{ms} g_s^{(c)} + z_{ms} g_s^{(r)})] \quad (10)$$

where $\gamma_m$ is the reward that $u_m$ can obtain for the unit task rate it handles. $\delta_m$ and $\omega_m$ are the weight factors indicating the importance of task service delay and payment, respectively, in the utility function of MD $u_m$. Moreover, as an ES in the network provides resources in terms of CRBs and RRBs, the utility of ES $e_s$ can be expressed as the income received from selling resource blocks minus the increment of various costs such as energy consumption, operation and equipment costs, etc., which can be denoted as

$$U_s(g_s^{(c)}, g_s^{(r)}) = \sum_{m=1}^{M} x_{ms}(y_{ms} g_s^{(c)} + z_{ms} g_s^{(r)}) - \sum_{m=1}^{M} x_{ms}(y_{ms} f_s^{(c)} + z_{ms} f_s^{(r)}) \quad (11)$$

where $f_s^{(c)}$ is the increment of cost per CRB, and $f_s^{(r)}$ is the increment of the cost per RRB. Each of MDs and ESs aims to achieve its maximal utility, that is,

$$\max_{\{x_{ms}, \alpha_{ms}, y_{ms}, z_{ms}\}} U_m, m = 1, 2, ..., M. \quad \max_{\{g_s^{(c)}, g_s^{(r)}\}} U_s, s = 1, 2, ..., S. \quad (12)$$

We exploit a multi-leader and multi-follower game [17], [18] and develop an algorithm for the MDs and ESs to achieve the above objective in a distributed way. As shown in Algorithm 1, all ESs, $e_s$, $\forall s \in \{1, 2, 3, ..., S\}$ firstly set the initial price for CRBs and RRBs to $max\ g_s^{(c)}$ and $max\ g_s^{(r)}$ respectively, which is a very high price so that no MDs $u_m$, $\forall m \in \{1, 2, 3, ..., M\}$ would choose to offload their DNN tasks to any of the ESs $e_s$. The ESs then start sequentially adjusting down their prices step-by-step to attract the MDs. Given the prices of the CRBs and RRBs set by the ESs, each MD would determine the ES to associate, the DNN layer partition point, and the amount of CRBs and RRBs to purchase to achieve its maximal utility. Specifically, as there are no close-form expressions for the layer-level computing time and output data size of DNNs, we build a DNN partition table that includes the required computation cycles and output data size for different DNN partition points based on the DNN profile [9]. A MD $u_m$ chooses an ES $e_s$ and calculates the DNN layer partition point $\alpha_{ms}$ and the amounts of CRBs and RRBs to purchase from the corresponding ES that yield the highest utility value according to (10) and the DNN partition table. It records the calculated results for every ES, including the DNN layer-level partition point, the amounts of CRBs and RRBs to purchase, and the obtained utility value. The MD then determines a combination of the ES to associate, the DNN partition point, and the amounts of CRBs and RRBs that achieve the overall maximal utility, which represents its best response to the prices of CRBs and RRBs set by the ESs. It sends the request to the ES it intends to associate.

**Algorithm 1** Algorithm for multi-leader multi-follower Stackelberg game

1: Initially, each ES $e_s$, $\forall s \in \{1, 2, 3, ..., S\}$ sets the price for CRB = $max\ g_s^{(c)}$ and the price for RRB = $max\ g_s^{(r)}$, which is a very high price so that no MD $u_m$ chooses to offload the DNN tasks to the ES
2: Each MD $u_m$ builds a DNN partition table based on the DNN profile
3: **while** at least one ES $e_s$ adjusts its price **do**
4:  **for** MD $u_m$ **do**
5:  **while** choose an ES $e_s, \forall s \in \{1, 2, 3, ..., S\}$ **do**
6:  MD $u_m$ calculates the DNN layer partition point $\alpha_{ms}$ and the amounts of CRBs and RRBs to purchase from ES $e_s$ that yield the highest utility value according to (10) and the DNN partition table
7:  records the calculation results for each ES
8:  **end while**
9:  MD $u_m$ determines a combination of the ES to associate, the DNN layer partition point, and the amounts of the CRBs and RRBs to purchase that achieves the overall maximal utility
10:  sends the request to the ES to associate
11:  **end for**
12:  **for** ES $e_s$ **do**
13:  Calculate its own utility based on eq (11) and the total number of CRBs and RRBs purchased by the associated MDs
14:  **if** $U_s(g_s^{(c)} + \Delta_i,\ g_s^{(r)} + \Delta_i) > U_s$ **then**
15:  $new\ g_s^{(c)} = \min\{\max\ g_s^{(c)},\ g_s^{(c)} + \Delta_i\}$
16:  $new\ g_s^{(r)} = \min\{\max\ g_s^{(r)},\ g_s^{(r)} + \Delta_i\}$
17:  **else**
18:  **if** $U_s(g_s^{(c)} - \Delta_i,\ g_s^{(r)} - \Delta_i) > U_s$ **then**
19:  $new\ g_s^{(c)} = \max\{0,\ g_s^{(c)} - \Delta_i\}$
20:  $new\ g_s^{(r)} = \max\{0,\ g_s^{(r)} - \Delta_i\}$
21:  **else**
22:  $new\ g_s^{(c)} = g_s^{(c)}$
23:  $new\ g_s^{(r)} = g_s^{(r)}$
24:  **end if**
25:  **end if**
26:  **end for**
27:  **for** ES $e_s$ **do**
28:  Each ES $e_s$ update the price for $g_s^{(c)} = new\ g_s^{(c)}$ and $g_s^{(r)} = new\ g_s^{(r)}$
29:  **end for**
30: **end while**

The ESs set their prices of CRBs and RRBs based on an iterative algorithm [16]. They update their prices in sequence for each round of iterations. An ES tries to decrease or increase its price by a fixed amount each time and then

calculates its utility according to (11), which depends on the behaviors of the MDs and other ESs. If the action (i.e., increasing or decreasing the price) increases the utility of the ES, it would perform the action correspondingly to update the price in the next round. Otherwise, it would keep its price. The ESs will announce the updated prices to the MDs. On the other hand, the MDs will recalculate the ES association, the partition point, and the amount of CRBs and RRBs to purchase once receiving the price updates. The above process continues until all ESs stop changing their prices, i.e., the increase for the utility due to the price change is less than a preset threshold.

## IV. Evaluation Results

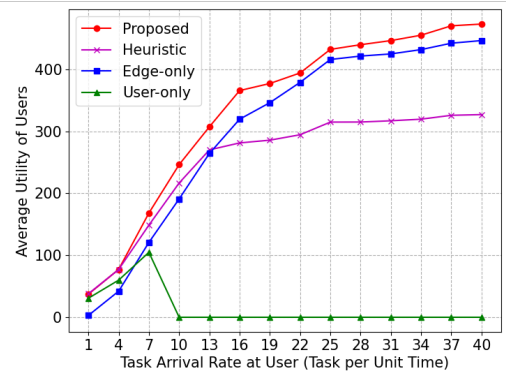
In this section, we present the evaluation results to show the performance of the proposed joint association, DNN layer-level partitioning, offloading, and resource allocation optimization scheme. We also compare the performance of our proposed scheme with three different benchmarks:

*a) User-only scheme:* A MD itself computes all layers of the DNN processing tasks without purchasing any CRB and RRB from any ES, in other words, no edge server association and DNN layer partitioning are performed for this benchmark scheme.

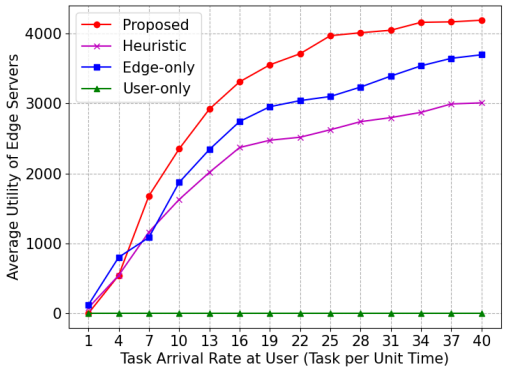
*b) Edge-only scheme:* All layers of the DNN processing tasks are offloaded to an associated ES for computation. That is, only the edge server association is performed and no DNN layer partitioning is executed.

*c) Heuristic scheme:* Each MD is associated to the ES with the association method based on the SNR of the co-located base station as in [12], and then only the DNN layer partitioning and offloading is applied.

For the first experiment, we introduced 20 MDs and 2 ESs randomly distributed in a circular area with a radius of 50 m. In addition, we consider that the MDs can run different machine learning applications with different DNN models. In the experiment, a MD may choose to run one of the three DNN models randomly, the AlexNet DNN model described in Section II, the speech recognition model, and the facial recognition model described in [9]. The ESs will support all the models. We assume that a mechanism as in [10] can be used to profile the DNN layer-level computation time and output data size. This paper focuses on determining MD association, resource allocation, and DNN partitioning point of multiple DNN operations. We first evaluate the average utility of the MDs and ESs when the average task arrival rates at the MDs change. As shown in Fig. 3a, the proposed scheme improves the average utility of the MDs, compared to all the benchmark schemes. This is because our scheme adjusts association, resource allocation, and DNN partition point to achieve joint optimization. The average MD utility increases as the average user task arrival rate increases and the MDs process more workload. The increasing rate is high at the beginning and slows down as the average task arrival rate becomes large. The reason is that when more workload is presented at the MDs, they need to purchase more CRBs and RRBs from the ESs, which results in a lower MD utility




(a)



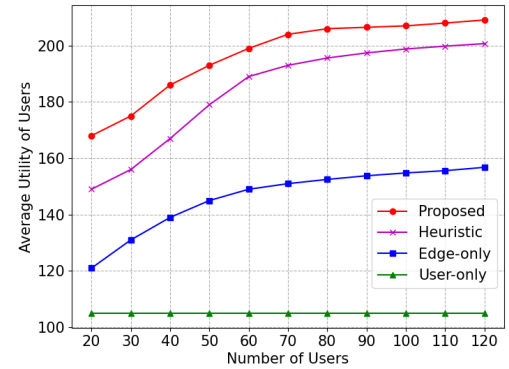

(b)

Fig. 3: The average utility of the MDs (a) and the average utility of the ESs (b) versus the average user task arrival rate.

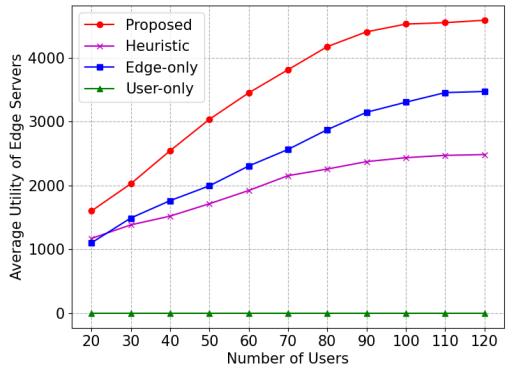
due to the payment to the ESs. Additionally, for the user-only scheme, the MDs can handle the workload at a low task arrival rate, however, the MD won't be able to handle the DNN tasks all by itself and the utility becomes zero due to overloading when the task arrival rate is greater than a threshold, 10 tasks per unit time with this setting.

Furthermore, we can observe in Fig. 3b that our proposed scheme also performs better than the benchmarks in terms of the average utility of the ESs due to joint optimization of association, resource allocation, and DNN partitioning point and offloading. In addition, the average utility of the ESs increases and then flats out as the average task arrival rate becomes large. The reason is that an ES has a limited number of CRBs and RRBs. After all the available CRBs and RRBs of ESs are allocated to their associated MDs, the ESs can increase their utility only by increasing the price of the CRBs and RRBs, but the impact of price increase to the ES utility would decrease since the MDs purchase fewer resources due to a high price. For the user-only scheme, the utility of the ESs is zero because the users will not offload their DNN processing tasks to the ESs.

For the second experiment, we fixed the average task arrival rate to 5 tasks per unit time for each MD and evaluated the average utility of the MDs and ESs as the number of MDs changed. As shown in Fig.4a, we can see that the average utility of the MDs for our proposed joint optimization scheme is higher than those of the benchmarks. In addition, as the number of MDs increases, the average utility of the




(a)




(b)

Fig. 4: The average utility of the MDs (a) and the average utility of the ESs (b) versus the number of MDs.

MDs increases at the beginning and becomes flat. This is because more MDs compete for the resources, causing higher resource prices, especially after the ESs allocate all their available CRBs and RRBs to the MDs. The MDs will get less amount of CRBs and RRBs at the same amount of payment, reducing the utility value. In Fig.4b, we evaluate the average utility of the ESs as the number of the MDs changes. Once again, the proposed scheme outperforms the benchmarks in terms of the average ES utility by jointly optimizing association, resource allocation, and DNN partitioning point. The average ES utility first increases at a high rate, and then the increasing rate slows down as the number of MDs becomes large. As discussed before, this is because the ESs can lease more resources to MDs at the beginning to increase their utility, but after they have allocated all their resources to the MDs, they could only increase the price to obtain a higher utility, which is less effective.

## V. Conclusions

This work proposes a joint optimization framework that partitions multiple DNN-based machine learning inference processes, determines the association of mobile devices to edge servers, and allocates computing and wireless communication resources for offloading part of DNN computations from multiple MDs to multiple ESs in a mobile edge computing network. A multi-leader multi-follower Stackelberg game is designed, which allows MDs and ESs to make distributed decisions to maximize their own utilities by leveraging the computation complexity and output data size characteristics of different DNN layers and taking into consideration computation load, available resources, and multi-server heterogeneous network environments. The evaluation results show that the proposed scheme can improve the utility of both MDs and ESs, compared to the benchmarks, especially when the DNN task arrival rates are high or the number of MDs increases.

## References


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv Preprint arXiv:1409.1556*, 2020.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[4] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[5] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2021.

[6] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for noma–mec offloading," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6207–6211, 2019.

[7] S. Xia, Z. Yao, Y. Li, and S. Mao, "Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6743–6757, 2021.

[8] W.-B. Sun, J. Xie, X. Yang, L. Wang, and W.-X. Meng, "Efficient computation offloading and resource allocation scheme for opportunistic access fog-cloud computing networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 2, pp. 521–533, 2023.

[9] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[10] B. Zhang, T. Xiang, H. Zhang, T. Li, S. Zhu, and J. Gu, "Dynamic dnn decomposition for lossless synergistic inference," in *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 13–20, 2021.

[11] X. Chen, M. Li, H. Zhong, Y. Ma, and C.-H. Hsu, "Dnnoff: offloading dnn-based intelligent iot applications in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2820–2829, 2022.

[12] P. Ren, X. Qiao, Y. Huang, L. Liu, C. Pu, and S. Dustdar, "Fine-grained elastic partitioning for distributed dnn towards mobile web ar services in the 5g era," *IEEE Transactions on Services Computing*, vol. 15, no. 6, pp. 3260–3274, 2022.

[13] L. Ahlin, J. Zander, and S. Ben Slimane, *Principles of wireless communications*. Studentlitteratur, 2006.

[14] A. Khalili, A. Ashikhmin, and H. Yang, "Cell-free massive mimo with low-complexity hybrid beamforming," in *ICC 2022 - IEEE International Conference on Communications*, 2022.

[15] C. Newell, *Applications of queueing theory*, vol. 4. Springer Science & Business Media, 2013.

[16] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[17] J. Chen, Q. Wu, Y. Xu, N. Qi, T. Fang, L. Jia, and C. Dong, "A multi-leader multi-follower stackelberg game for coalition-based uav mec networks," *IEEE Wireless Communications Letters*, vol. 10, no. 11, pp. 2350–2354, 2021.

[18] J. Patel and K. M, "Game theoretic approach for electric vehicles charging solution: A study of the interplay between time, price and location," in *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, 2023.