Computation Offloading in Multi-Access Edge Computing Networks: A Multi-Task Learning Approach

Conference Paper · May 2019

DOI: 10.1109/ICC.2019.8761212

6 authors, including:

Bo Yang
Prairie View A&M University
8 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Xiangfang Li
Institute of Electrical and Electronics Engineers
213 PUBLICATIONS **2,525** CITATIONS
SEE PROFILE

Lijun Qian
Prairie View A&M University
162 PUBLICATIONS **1,912** CITATIONS
SEE PROFILE

Computation Offloading in Multi-Access Edge Computing Networks: A Multi-Task Learning Approach

Bo Yang*, Xuelin Cao†, Joshua Bassey*, Xiangfang Li*, Timothy Kroecker‡, Lijun Qian*
*Department of Electrical and Computer Engineering and CREDIT Center, Prairie View A&M University,
Texas A&M University System, Prairie View, TX 77446, USA
†Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004, USA
‡US Air Force Research Laboratory (AFRL), Rome, NY 13441, USA

***Abstract*—Multi-access edge computing (MEC) has already shown the potential in enabling mobile devices to bear the computation-intensive applications by offloading some tasks to a nearby access point (AP) integrated with a MEC server (MES). However, due to the varying network conditions and limited computation resources of the MES, the offloading decisions taken by a mobile device and the computational resources allocated by the MES may not be efficiently achieved with the lowest cost. In this paper, we propose a dynamic offloading framework for the MEC network, in which the uplink non-orthogonal multiple access (NOMA) is used to enable multiple devices to upload their tasks via the same frequency band. We formulate the offloading decision problem as a multiclass classification problem and formulate the MES computational resource allocation problem as a regression problem. Then a multi-task learning based feedforward neural network (MTFNN) model is designed to jointly optimize the offloading decision and computational resource allocation. Numerical results illustrate that the proposed MTFNN outperforms the conventional optimization method in terms of inference accuracy and computation complexity.**

***Index Terms*—Multi-access edge computing, computation offloading, non-orthogonal multiple access, multi-task learning.**

I. Introduction

To cope with the exponentially increasing data traffic with stringent requirements on computation resources, multi-access edge computing (MEC) network plays a key role in bringing cloud functionalities to the edge that in close proximity to mobile devices which support multiple access [1]. With computation offloading technique, the resource-constrained mobile devices can save energy and enrich users' experience by fully or partially offloading computation-intensive tasks to the nearby MEC server (MES). The MES could be colocated with access points (APs), wireless relays or small base stations (BSs), as shown in Fig. 1. Due to a large amount of computation input data may be uploaded from the mobile devices to the MES, abundant wireless spectrum is required, which has become more and more scarce and precious. To alleviate this problem, non-orthogonal multiple access (NOMA) has been introduced into the (MEC) networks of the 5G era enabling multiple devices to transmit their data simultaneously on the same frequency band [2].

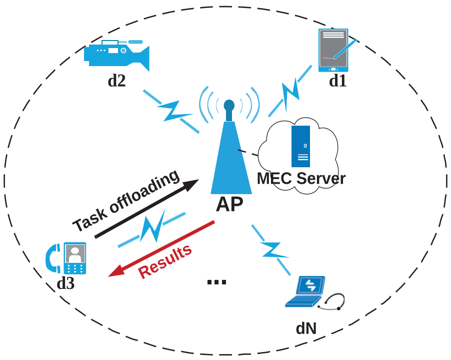


Fig. 1: Task offloading scenario in MEC system

In order to minimize the tasks completion time and energy consumption of the mobile devices, one of the challenges in offloading computation-intensive tasks to MES is to *determine whether to offload and the portion of computational resources allocated to the device*. This could be formulated as a mixed integer nonlinear programming (MINLP) problem that minimizes the total system cost (e.g., the weighted sum of delay and energy consumption) under the constraints of the task's tolerable delay and MES's available resources, which is NP-hard in general [3]. Furthermore, *the input parameters to the optimization problem may vary frequently, which leads to the requirement of the offload decision making in near-real-time*. However, the conventional optimization algorithms usually take a lot of time to solve this NP-hard optimization problem and only sub-optimal solutions are obtained in some cases. As a result, the optimal or sub-optimal offloading decision may not be reached within the specified delay constraints (e.g., in the video conference, real-time image processing, etc.).

In order to address this challenge, a novel computation offloading framework is proposed in this work that can adapt to the varying network conditions and the requirements of devices' applications. Specifically, a multi-task learning based feedforward neural network (MTFNN) model is designed to solve the mixed integer nonlinear programming (MINLP) problem in near-real-time, where the offloading decision making is formulated as a multiclass classification problem and the computational resource allocation is formulated as a regression problem. The proposed MTFNN model is firstly trained offline with the dataset collected by traversing all the possible combinations of features including the parameters representing wireless channel conditions. Then the trained MTFFN model can be used to predict the optimal offloading decision and computational resource allocation in near-real-time with high accuracy. Simulation results show that the proposed MTFFN model based offloading scheme achieves better performance compared with existing benchmark offloading approaches.

The remainder of this paper is organized as follows. Section II presents the system models. In Section III, we present the formulation and analysis of the cost minimization problem. In Section IV, we describe our proposed offloading scheme in detail. Numerical results are given in Section V, followed by a review of related works in Section VI. Finally, we conclude this paper in Section VII.

II. System Models

A. Network Model

We consider a scenario of multi-devices single-MES in the MEC network, as shown in Fig. 1. There exist N devices, i.e., $\mathcal{N} = \{d_1, d_2, ..., d_N\}$, which are associated with the MES located in the AP (denoted as $\mathcal{A}$). In this architecture, the widely deployed WLAN can be considered as a potential technology for wireless communications, which works on the unlicensed frequency band. We assume that each device $d_i, \forall i \in [1, N]$, has only one computation-intensive task (denoted as $\mathcal{J}_i$) to be processed during a computation offloading period, which is atomic and cannot be further divided. Each device can choose to offload $\mathcal{J}_i$ to the MES through wireless links or execute it locally In general, the total computation ability and storage capacity of the MES is limited and thus maybe not always sufficient for all associated devices to offload their tasks simultaneously.

B. Task Model

For $\mathcal{J}_i, \forall i \in [1, N]$, d_i can only execute it locally or by offloading computing in the MES. Denote $D_i \in \{0, 1\}$ as the computation offloading decision, which is a N-dimensional binary vector (denoted as $\mathbf{D}$), i.e., $\mathbf{D} = [D_1, D_2, ..., D_N]$. Specifically, we have $D_i = 0$ when d_i executes $\mathcal{J}_i$ locally. Otherwise, $D_i = 1$ can hold for the computation offloading. Moreover, we let $\mathbf{F} = [f_1, f_2, ..., f_N]$ be the allocated computational resource (i.e., central processing unit (CPU) cycles per second) vector by the MES. Hence, the offloading strategy can be defined as $\boldsymbol{S} = \{\mathbf{D}, \mathbf{F}\}$. In order to make $\mathcal{J}_i$ more visible and intuitive, we characterize $\mathcal{J}_i$ by a three-tuple of parameters, i.e., $\mathcal{J}_i(s_i, c_i, \vartheta_i)$. In particular, s_i denotes the size of computation input data needed for processing $\mathcal{J}_i$. c_i denotes the total number of CPU cycles required to process $\mathcal{J}_i$, and ϑ_i denotes the maximum tolerable delay of $\mathcal{J}_i$.

C. Communication Model

In the uplink NOMA system, the received signal from d_i in $\mathcal{A}$ is given as

$$y_i = \underbrace{\sqrt{P_t^i} h_i x_s}_{\text{Desired signal}} + \underbrace{\sum_{j \neq i, j \in \mathcal{N}} \sqrt{P_t^j} h_j x_j}_{\text{Interferences}} + \underbrace{z_i}_{\text{Noise}}, \quad (1)$$

where h_i denotes the channel power gain[1] for d_i connecting with $\mathcal{A}$. P_t^i is the transmit power of d_i, and the noise power z_i can be generally considered as the white Gaussian noise in additive white Gaussian noise (AWGN) channel with zero mean and variance δ^2.

To separate and decode the overlapped signals from $d_i, \forall i \in [1, N]$, the successive interference cancellation (SIC) modular can be implemented in $\mathcal{A}$. By sorting the overlapped signals descendly according to the channel gains, i.e.,

$$h_1 \geqslant h_2 \geqslant h_3 \geqslant ... \geqslant h_N, \ \forall i \in [1, N], \quad (2)$$

the received signal-to-interference-plus-noise ratio (SINR) of d_i served by $\mathcal{A}$ can be calculated as

$$\mathrm{SINR}_i = \frac{P_t^i |h_i|^2}{\delta^2 + \sum_{j=i+1}^{N} P_t^j |h_j|^2}. \quad (3)$$

D. Computation Model

For the offloading strategy $\boldsymbol{S} = \{\mathbf{D}, \mathbf{F}\}$, the offloading decision of $d_i, \forall i \in [1, N]$ can be "locally" or "offloading", i.e., $D_i \in \{0, 1\}$, so the two computation models are presented as follows.

1) Locally: Let τ_l^i be the local execution delay of $\mathcal{J}_i$, denote f_l^i as the the CPU cycle frequency (i.e., CPU cycles per second) of d_i[2]. The local execution delay of $\mathcal{J}_i$ is

$$\tau_l^i = \frac{c_i}{f_l^i}. \quad (4)$$

Denote κ as the energy efficiency parameter that is mainly depends on the chip architecture [4]. In order to process $\mathcal{J}_i$ with the CPU clock speed f_l^i, the energy consumption is

$$\varepsilon_l^i = \kappa \left(f_l^i\right)^2 c_i. \quad (5)$$

Based on (4), (5), the total cost for computing $\mathcal{J}_i$ locally is

$$\mathcal{O}_l^i = \alpha \tau_l^i + \beta \varepsilon_l^i, \quad (6)$$

where α and β are the weights of execution delay and energy consumption. In general, $0 \leq \alpha, \beta \leq 1$ and $\alpha + \beta = 1$ hold.

[1] It is assumed that the channel remains static within each time frame, in which the optimal offloading strategy $\boldsymbol{S} = \{\mathbf{D}^*, \mathbf{F}^*\}$ can be obtained.

[2] Without loss generality, we assume that the computational capabilities of each device may be different.

2) Offloading: To process $\mathcal{J}_i$ with the offloading approach, d_i firstly needs to upload the data to $\mathcal{A}$ which is co-located with the MES through the wireless access network. Then, the MES allocates the computational resources accordingly and execute $\mathcal{J}_i$ instead. Finally, $\mathcal{A}$ returns the executing results to d_i. In the following, we describe the three stages in detail.

- *Uploading*. The delay to upload data to $\mathcal{A}$ is
$$T_u^i = \frac{s_i}{r_u^i}, \tag{7}$$
where r_u^i stands for the achieved uplink data rate of the wireless link from d_i to $\mathcal{A}$. Denote W as the frequency bandwidth, we have $r_u^i = Wlog_2(1 + \mathrm{SINR}_i)$.
The energy consumption during the data uploading is
$$e_u^i = P_t^i T_u^i = \frac{P_t^i s_i}{Wlog_2(1 + \mathrm{SINR}_i)}. \tag{8}$$
- *Processing*. The time to process $\mathcal{J}_i$ by the MES is
$$T_p^i = \frac{c_i}{f_i}, \tag{9}$$
where f_i denotes the allocated computational resource to d_i by the MES. Let F be the entire resources of the MES, we have $\sum_{i=1}^N D_i f_i \leq F$.
We suppose that d_i stays idle while waiting for the results from MES, the power consumption is defined as P_I^i. The energy consumption is
$$e_I^i = P_I^i T_p^i = \frac{P_I^i c_i}{f_i}. \tag{10}$$
- *Downloading*. The time to download the executive results from $\mathcal{A}$ is
$$T_d^i = \frac{w_i}{r_d^i}, \tag{11}$$
where w_i is the size of the results, r_d^i denotes the data rate of the wireless down link between $\mathcal{A}$ and d_i.
For d_i, $\forall i \in [1, N]$, denote the power required to download the execusive results as P_d^i.
Accordingly, the energy consumption of d_i during downloading the results is
$$e_d^i = P_d^i T_d^i = \frac{P_d^i w_i}{r_d^i}. \tag{12}$$

Generally, due to $w_i \ll s_i$ (e.g., face recognition) and r_d^i is relatively high, the total execution delay and energy consumption of d_i can be approximately given as
$$\tau_o^i \approx \frac{s_i}{Wlog_2(1 + \mathrm{SINR}_i)} + \frac{c_i}{f_i}, \tag{13}$$
$$\varepsilon_o^i \approx \frac{P_t^i s_i}{Wlog_2(1 + \mathrm{SINR}_i)} + \frac{P_I^i c_i}{f_i}, \tag{14}$$
where T_d^i and e_d^i can be neglected [5].

Therefore, the total cost for computing $\mathcal{J}_i$ is
$$\mathcal{O}_o^i = \alpha \tau_o^i + \beta \varepsilon_o^i. \tag{15}$$

To this end, the sum cost of all devices can be expressed as
$$\mathcal{O}_{total} = \sum_{i=1}^N (1 - D_i)\, \mathcal{O}_l^i + D_i \mathcal{O}_o^i. \tag{16}$$

III. Cost Minimization Problem

A. Problem Formulation

In this subsection, we formulate the offloading and resource allocation by the MES as a cost minimization problem (**P1**).

$$\begin{aligned} \mathbf{P1}: \ & \underset{\mathbf{D},\mathbf{F}}{\text{minimize}} \ \mathcal{O}_{total} \\ \text{s.t.} \ & \mathbf{C1}: \ D_i \in \{0,1\}\,,\ \forall i \in N, && \text{(17a)} \\ & \mathbf{C2}: \ (1 - D_i)\, \tau_l^i + D_i \tau_o^i \leq \vartheta_i, && \text{(17b)} \\ & \mathbf{C3}: \ 0 \leq f_i \leq F,\ \forall i \in N, && \text{(17c)} \\ & \mathbf{C4}: \ \sum_{i=1}^N D_i f_i \leq F,\ \forall i \in N. && \text{(17d)} \end{aligned}$$

In the optimization problem, $\mathbf{D} = [D_1, D_2, ..., D_N]$ is the offloading decision, and $\mathbf{F} = [f_1, f_2, ..., f_N]$ denotes the computational resource allocation. $\mathbf{C1}$ shows that d_i can only choose to execute $\mathcal{J}_i$ locally or offloading to the MES. $\mathbf{C2}$ makes sure that the time cost to process $\mathcal{J}_i$ should not exceed the maximum tolerable delay ϑ_i. $\mathbf{C3}$ and $\mathbf{C4}$ guarantee that the computational resource allocated to d_i and the sum of the computational resources allocated to all the offloading devices should not exceed the total resources of the MES.

B. Problem Analysis

Intuitively, the optimization problem $\mathbf{P1}$ can be solved by going through all the combinations of the offloading decision vector $\mathbf{D}$ and the computational resource allocation $\mathbf{F}$. Denote the optimal offloading decision and computational resource allocation result as $\mathbf{D}^*$ and $\mathbf{F}^*$, i.e.,
$$\{\mathbf{D}^*, \mathbf{F}^*\} = \underset{\mathbf{D},\mathbf{F}}{\operatorname{argmin}} \ \mathcal{O}_{total}. \tag{18}$$

However, due to the fact that $\mathbf{D}$ is the binary vector, and the objective function of $\mathbf{P1}$ is not convex, so the resolving of $\mathbf{P1}$ is difficult to tackle [3]. Generally, the spatial branch and bound (sBB) method is used to solve this problem [6], where a hierarchy of nodes represented by a binary tree is created (a.k.a. the sBB tree) and then a pure continuous NLP sub-problem can be formed by dropping the integrality requirements of the discrete variables [7]. As a result, the initial optimization problem $\mathbf{P1}$ becomes the root of the sBB tree. Although the sBB can resolve the MINLP problem faster than the exhaustive searching, large overhead will be still introduced into the MEC networks due to the varying of channel condition and input parameters. Moreover, the obtained results using the sBB method are sometimes sub-optimal, which degrades the performance of the MEC system. In this paper, instead of the conventional optimization methods, we build a machine learning model to predict $\mathbf{D}^*$ and $\mathbf{F}^*$ more efficiently while ensuring the prediction accuracy.

IV. Computation Offloading with Multi-task Learning

A. Problem Mapping

The two output vectors (i.e., $\mathbf{D}^*$ and $\mathbf{F}^*$) of $\mathbf{P1}$ are related to each other. If we consider the prediction of $\mathbf{D}^*$ and $\mathbf{F}^*$

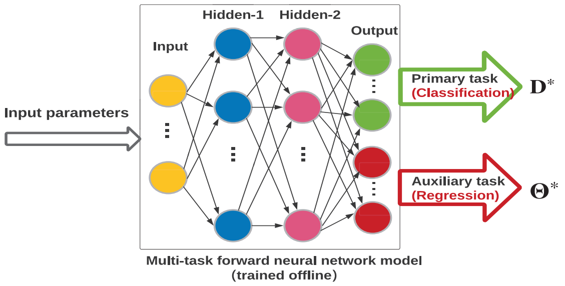


Fig. 2: The proposed MTFNN model with an input layer, two hidden layers, and an output layer. Taking $N = 3$ as an example, the input contains 18 neurons, the two hidden layers contain 15 and 10 neurons, respectively. In the output layer, the classification output contains 8 neurons and the regression output contains 3 neurons

as two machine learning tasks, it is known that learning the two related tasks jointly can get better generalization effect than the learning them individually [8]. Therefore, $\mathbf{P1}$ can be formulated as a multi-task learning (MTL) problem, as shown in Fig. 2, where the MTFNN model is proposed to predict $\mathbf{D}^*$ and $\mathbf{F}^*$ with multi-task learning. Suppose that there exist l learning tasks $\{\mathcal{T}_i\}_{i=1}^{l}$ that are related to each other, where $l = 2$ in our proposed MTFNN model. Each learning task $\mathcal{T}_i$ is usually accompanied by a training dataset $\mathcal{S}_i$ which consists of m_i training samples, i.e., $\mathcal{S}_i = \left\{\mathbf{X}_j^{(i)}, \mathbf{Y}_j^{(i)}\right\}_{j=1}^{m_i}$, where $\mathbf{X}_j^{(i)}$ is the j-th training instance in $\mathcal{T}_i$, $\mathbf{Y}_j^{(i)}$ represents its label. For the output, denote $y_j^{(i)}$ as the j-th corresponding output from $\mathbf{Y}_j^{(i)}$. When $y_j^{(i)}$ is in a discrete space, e.g., $y_j^{(i)} \in \{0,1\}$ for $\mathbf{D}^*$, and thus the corresponding task can be considered as a multiclass classification problem, where the task is to predict a discrete offloading decision class for a given set of input parameters. If $y_j^{(i)}$ is continuous, e.g., $y_j^{(i)} \in \mathbb{R}$ for $\mathbf{F}^*$, the corresponding task turns to be a regression problem, where the task is to predict a numeric value. It should be noted that in our regression model, we define all the labels as $\mathbf{\Theta}_j^{(i)} = \mathbf{Y}_j^{(i)}/F$ for simplicity. Therefore, instead of $\mathbf{F}$, the prediction of the regression model becomes a computational resource allocation ratio, i.e., $\mathbf{\Theta}_j^{(i)} \in [0.0, 1.0]$. Owing that the offloading decision plays a more important role in the MEC networks, so the prediction of $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ can be considered as a primary task (a.k.a. classification problem) and a auxiliary task (a.k.a. regression problem) in our MTFNN model, respectively.

B. Data Collection

We independently generate 4×10^4, 5×10^4, 8×10^4 and 10^5 data samples for $N \in [2, 5]$ in the dataset by traversing all the possible combinations of $\mathbf{D}$ and $\mathbf{\Theta}$ with the exhaustive searching algorithm[3], so $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ can be obtained for a given set of parameters. During each execution, the network parameters are randomly chosen from their ranges given in Table I, and the statical parameters are given as follows. The channel bandwidth (W) is 1 MHz, and the white noise power is (δ^2) is 7.9×10^{-13}. The energy efficiency parameter (κ) is set as 1×10^{-28}. The CPU computation capacity of the MES (F) is 2.5 GHz. The transmission power (P_t) and idle power (P_I) of each device are set to be 0.3 W and 0.1 W, respectively [9]. The uplink data rate (r_u^i) can be calculated according to $r_u^i = Wlog_2(1 + \text{SINR}_i)$. In order to enable the collected data to be applied to our MTFNN model, we preprocess the dataset as a specific groundtruth matrix $\mathbf{H}$. Specifically, for each device $d_i, \forall i \in [1, N]$, the input parameters of the MTFNN model include s_i, c_i, f_l^i, h_i, α_i and β_i. The output from the MTFNN model includes $\mathbf{D}^*$ and $\mathbf{\Theta}^*$. The collected dataset is split into 80% for training phase and the rest 20% for testing phase.

TABLE I:
CRITICAL PARAMETERS AND DEFINITIONS

Parameters	Value range
The number of devices (N)	$2, 3, 4, 5$
Data payload size (s)	$[1 - 500]$ kbits
CPU cycle required to process the data (c)	$[3 - 1500]$ Megacycles
CPU frequency of the device (f_l)	[1Hz – 1GHz]
Weights of delay and energy cost (α, β)	$[0.0 - 1.0]$

C. Offline Training

During the training phase, we train the MTFFN model which contains 2 hidden layers, as shown in Fig 2, using the collected data. In our MTFFN model, for the classification problem, the probability of each class is predicted using the Softmax function, i.e., the predicted probability for the j-th class given a sample vector $\mathbf{x}$ and a weighting vector $\mathbf{w}$ is $P(y = j|\mathbf{x}) = \frac{\exp(\mathbf{x}^{\mathrm{T}}\mathbf{w}_j)}{\sum_{k=1}^{K}\exp(\mathbf{x}^{\mathrm{T}}\mathbf{w}_k)}$, where K is the number of classes. We conventionally set the loss function of the multi-class classification (denoted as $\boldsymbol{l}_c$) as cross-entropy [10]. For the regression problem, the loss function (denoted as $\boldsymbol{l}_r$) is calculated using mean square error (MSE) [11]. In our proposed MTFFN model, the loss function is defined as $\boldsymbol{l} = \chi_1 \boldsymbol{l}_c + \chi_2 \boldsymbol{l}_r$, where χ_1 and χ_2 are the weights. Here, we have $\chi_1 = \chi_2 = 1$ and the Adam optimizer [12] is used to optimize the MTFFN model. Therefore, the prediction of $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ can be obtained when $\boldsymbol{l}$ is minimized. It should be noted that several methods have been proposed to scale up deep neural network (DNN) training across graphics processing unit (GPU) clusters [13], which helps to reduce the runtime of the offline training.

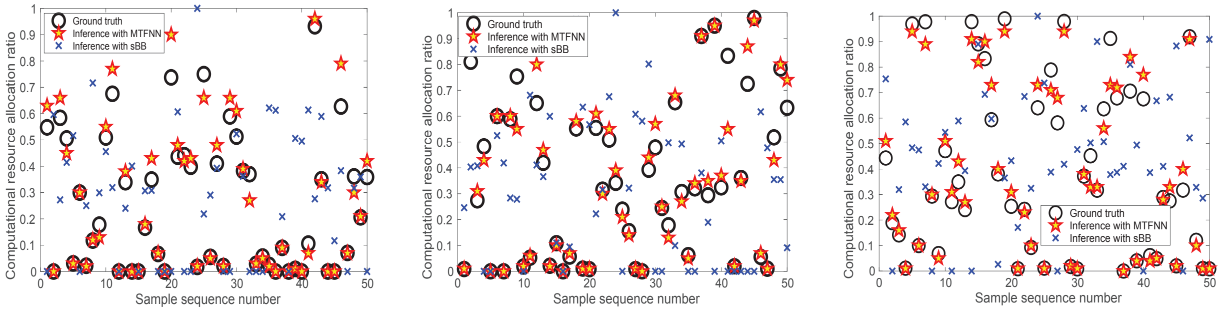
V. RESULTS AND DISCUSSIONS

A. Testing Results

During the testing phase, the performance of the MTFFN model is evaluated based on the outputs[4] and the corresponding labels. To demonstrate the superiority on resolving the MINLP problem using the proposed MTFFN model, we compare with a benchmark scheme "sBB" which is

[3] Due to the $\mathbf{\Theta}$ is a decimal vector which ranges from $[0.0,\ 1.0]$. In this paper, the interval between traversal values is set as 0.1, which is also denoted as the granularity of resource allocation (ω) in the follow-up contents.

[4] The outputs obtained from the MTFFN model are performed 50 epochs and normalized to make sure the condition $\mathbf{C4}$ is met.



(a)

(b)

(c)

Fig. 3: The predicted computational resource ratio of the MES (i.e., $\mathbf{\Theta} = [\Theta_1, \Theta_2, \Theta_3]$), where the number of devices is 3. Θ_1, Θ_2 and Θ_3 are respectively shown in (a), (b) and (c)

TABLE II:
Computation Accuracy and Complexity

η, ε, t S N	sBB	MTFFN
2	$70\%,\ 0.055,\ 14.1\ ms$	$96\%,\ 0.016,\ 2.5\ \mu s$
3	$62\%,\ 0.047,\ 14.2\ ms$	$89\%,\ 0.027,\ 2.5\ \mu s$
4	$58\%,\ 0.053,\ 14.5\ ms$	$83\%,\ 0.029,\ 2.2\ \mu s$

implemented using the MATLAB toolbox of the APMonitor Optimization Suite [14].

1) Inference accuracy: The inference accuracy of getting $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ are defined as follows. We define $\eta = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$ to indicate the accuracy of the offloading decision making (a.k.a. multiclass classification). We use the MSE to indicate the accuracy of resource allocation strategy (a.k.a. regression), i.e., $\varepsilon = \frac{1}{mN}\sum_{i=1}^{m}\sum_{j=1}^{N}(y_j^i - x_j^i)^2$, where m denotes the total number of samples, N is the total number of devices. y_j^i is the predicted value of d_j from the i-th sample and x_j^i is its label.

2) Computation complexity: In this paper, the computation complexity denotes the execution time per sample, which is defined as $t = \frac{Total\ execution\ time}{Number\ of\ samples}$. As the number of devices (denoted as N) grows, the conventional exhaustive search strategy suffers from the exponential time complexity $O((2g)^N)$, where $g = \frac{1}{\omega}+1, \omega \in (0,1)$ denotes the granularity of computational resource allocation. Meanwhile, to solve the MINLP problem, the sBB always has exponential worst-case complexity, i.e., $O(2^N)$ [15]. In our proposed MTFFN model, the quadratic time complexity can be achieved as $O(M^2L)$, where L is the number of layers, M is the number of neurons in a hidden layer which indicates the scale of the neuron network model. Moreover, we just need to train our learning model once, which can be performed offline via the machines with strong computing and storage capabilities, e.g., the GPU clusters. Therefore, the MTFFN model has a relatively low complexity compared to the "sBB" and exhaustive search schemes.

The inference accuracy (η and ε) and computation complexity (t) are reported in Table II[5]. It can be observed that compared to the "sBB", our proposed "MTFNN" model obtains the much lower complexity on the premise of a relatively high accuracy, i.e., the time cost of "MTFNN" is less than one-tenth of one percent of the "sBB", and outperforms the "sBB" by average 40% in the classification inference accuracy. Moreover, the performance comparison of regression inference for the case of three devices is presented in Fig. 3. It can be observed that the proposed "MTFNN" model predicts the computational resource ratio more accurately, which match the ground truth well.

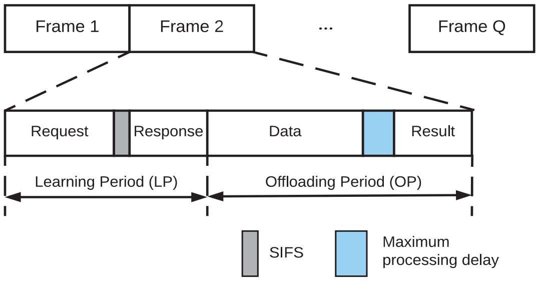


Fig. 4: MAC framework

B. Implementation

A MAC framework is presented to achieve channel coordination and intelligent offloading, as shown in Fig. 4, where the MTFNN module can be deployed onto the MES co-located with the AP [16]. Different from using TDMA and OFDMA protocols for the uplink offloading in [5] and [17], in our MAC framework, the time is divided into Q frames, each of which is subdivided into a learning period (LP) and an offloading period (OP). During the LP, all devices with tasks to be processed send a "Request" message to AP using NOMA[6], which contains the input parameters to the MTFFN model. On receiving the "Request", AP decodes the "Request" message based on the SIC technology and then predicts $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ based on the MTFNN model[7]. Then,

[5] Note that the "Schemes" is abbreviated as "S" to save space in Table II.

[6] Although an uplink NOMA scenario is assumed in this paper, our proposed MTFFN model can be also extended into other orthogonal multiple access schemes with a minor modification on the communication model, and then re-train the model offline.

[7] We assume that the channel remains static within each frame.

AP notifies the devices of the predicted $\mathbf{D}^*$ and $\mathbf{\Theta}^*$ by replying a "Response" message. During the OP, the devices with $\mathbf{D}^* \neq 0$ send the data needed to be processed to AP simultaneously in a NOMA way. On receiving the data, AP processes the corresponding tasks with the computational resources allocated to these tasks based on $\mathbf{\Theta}^*$, i.e, $\mathbf{F}^* = \mathbf{\Theta}^* \cdot F$. After all the tasks have been processed, AP returns the executed results back to the devices.

VI. Related Works

Recently, joint optimization of computation offloading strategy and computing resources allocation to achieve different objectives has received ever-increasing attention [18]–[22]. Specifically, a large body of existing works solve the optimization by seeking an optimal or sub-optimal solution using mathematical algorithms. For example, the authors in [18] proposed an optimal offloading strategy using convex optimization. In [19], an efficient game-theoretic computation offloading scheme was proposed for MEC in 5G HetNets. However, due to the time-varing characteristic of the wireless channel, the previous works need to resolve the optimization problem very frequently to obtain the optimal/sub-optimal offloading decision, which introduces a large overhead to the MEC system. To tackle this problem, combining machine learning with the computation offloading has become an effective and attractive solution. In [20], an optimal offloading scheme was proposed for intermittently connected fog system using Markov decision process (MDP). In [21], a machine learning-based runtime adaptive scheduler was proposed for a mobile offloading framework based on past behavior and current conditions. In [22], an offloading decision problem was formulated as a multi-label classification problem for a single-user single-cell scenario, and a deep supervised learning method is developed to minimize the system overhead. Even though the offloading decision-making problem can be solved in [20]–[22], the computational resource allocation problem for the resource-limited MES is not considered.

VII. Conclusions

In this paper, we presented a multi-task learning (MTL) enabled offloading framework for MEC networks with uplink NOMA. We first formulated the joint optimization problem of offloading decision and MES computation resource allocation as a mixed integer nonlinear programming problem. Then, we developed an MTL based feedforward neural network model to solve the optimization problem more efficiently with high accuracy. Simulation results demonstrate that our proposed offloading approach achieves better performance than other benchmarks in terms of system cost saving. This paper is one of our first attempts to integrate MEC system design with machine learning. Future work is in progress to take more complicated scenario into consideration.

Acknowledgment

This research work is supported in part by the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119.

References

[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19(3), pp. 1657-1681, May 2017.

[2] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Hybrid computation offloading in fog and cloud networks with non-orthogonal multiple access," in Proc. *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Honolulu, HI, April 2018.

[3] S. Burer, and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17(2), pp. 97-106, July 2012.

[4] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 13(2-3), pp. 203-221, August 1996.

[5] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing System," in Proc. *IEEE International Conference on Communications (ICC)*, Paris, France, May 2017.

[6] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wchter, "Branching and bounds tighteningtechniques for non-convex MINLP," *Optimization Methods & Software*, vol. 24(4-5), pp. 597-634, August 2009.

[7] E.M.B. Smith and C.C. Pantelides, "A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs," *Computers & Chem. Eng.*, vol. 23, pp. 457-478, May 1999.

[8] R. Caruana, "Multitask Learning," Springer US, July 1997.

[9] Y. Cao, T. Jiang, and C. Wang, "Optimal radio resource allocation for mobile task offloading in cellular networks," *IEEE Network*, vol. 28(5), pp. 68-73, September 2014.

[10] J. Nam, J. Kim, E. L. Menca, I. Gurevych, and J. Frnkranz, "Large-scale multi-label text classificationrevisiting neural networks," in Proc. *In Joint european conference on machine learning and knowledge discovery in databases*, Springer, Berlin, pp. 437-452, September, 2014.

[11] D. M. Allen, "Mean square error of prediction as a criterion for selecting variables," *Technometrics*, vol. 13(3), pp. 469-475, 1971.

[12] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," in Proc. *International Conference for Learning Representations (ICLR)*, San Diego, May 2015.

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," in Proc. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, June 2016.

[14] J. Hedengren, "MATLAB toolbox for the APMonitor Modeling Language," [online]: http://APMonitor.com/wiki/index.php/Main, 2014.

[15] G. Pataki, M. Tural, and E. B. Wong, "Basis reduction and the complexity of branch-and-bound," In Proc. *ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, January, 2010.

[16] B. Yang, X. Cao, and L. Qian, "A scalable MAC framework for Internet of Things assisted by machine learning," in *Proc. IEEE VTC2018-Fall*, Chicago, IL, August, 2018.

[17] J. Feng, L. Zhao, J. Du, X. Chu, and F. R. Yu,"Energy-Efficient Resource Allocation in Fog Computing Supported IoT with Min-Max Fairness Guarantees," in Proc. *IEEE International Conference on Communications (ICC)*, Kansas City, MO, May 2018.

[18] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31(6), pp. 45-55, November 2014.

[19] H. Guo, J. Liu, and J. Zhang, "Efficient Computation Offloading for Multi-Access Edge Computing in 5G HetNets," in Proc. *IEEE International Conference on Communications (ICC)*, Kansas City, MO, May 2018.

[20] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14(12), pp. 2516-2529, February 2015.

[21] H. Eom, P. S. Juste, R. Figueiredo, O. Tickoo, R. Illikkal, and R. Iyer, "Machine learning-based runtime scheduler for mobile offloading framework," in Proc. *IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, Washington, DC, December 2013.

[22] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: a deep learning approach," in Proc. *IEEE Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Montreal, QC, October 2017.