IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2019, accepted October 3, 2019, date of publication October 11, 2019, date of current version October 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2947053

A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing

ZAIWAR ALI[1], LEI JIAO[2], (Senior Member, IEEE), THAR BAKER[3], (Member, IEEE), GHULAM ABBAS[4], (Senior Member, IEEE), ZIAUL HAQ ABBAS[5], AND SADIA KHAF[5]

[1]Telecommunications and Networking (TeleCoN) Research Laboratory, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[2]Department of Information and Communication Technology, University of Agder (UiA), 4898 Grimstad, Norway
[3]Department of Computer Science, Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool L3 3AF, U.K.
[4]Faculty of Computer Sciences and Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan
[5]Faculty of Electrical Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan

Corresponding author: Thar Baker (t.baker@ljmu.ac.uk)

ABSTRACT Mobile edge computing (MEC) has shown tremendous potential as a means for computationally intensive mobile applications by partially or entirely offloading computations to a nearby server to minimize the energy consumption of user equipment (UE). However, the task of selecting an optimal set of components to offload considering the amount of data transfer as well as the latency in communication is a complex problem. In this paper, we propose a novel energy-efficient deep learning based offloading scheme (EEDOS) to train a deep learning based smart decision-making algorithm that selects an optimal set of application components based on remaining energy of UEs, energy consumption by application components, network conditions, computational load, amount of data transfer, and delays in communication. We formulate the cost function involving all aforementioned factors, obtain the cost for all possible combinations of component offloading policies, select the optimal policies over an exhaustive dataset, and train a deep learning network as an alternative for the extensive computations involved. Simulation results show that our proposed model is promising in terms of accuracy and energy consumption of UEs.

INDEX TERMS Computational offloading, deep learning, energy efficient offloading, mobile edge computing, user equipment.

I. INTRODUCTION

Mobile and wearable devices, herein after referred to as user equipment (UE), have experienced a tremendous increase in computational power over the years but, the applications running on these devices are becoming increasingly complex at the same time [1], [2]. The task of executing computationally intensive applications on devices is not fully prepared to handle the computational workload, which demands an alternative solution. Cloud computing gained much popularity as a promising alternative [3]. However, the delays involved in communication between the UEs and the cloud servers pose serious challenges on the viability of such solutions [4]. Consequently, Mobile Cloud Computing (MCC) is not an effective solution to manage the computational needs of mobile devices [5], [6]. Recently, placing small edge servers close to end-users to reduce the latency was proposed by the European Telecommunications Standards Institute (ETSI), and has been studied in [7] and [8] as Mobile Edge Computing (MEC). MEC is particularly important for applications that are delay sensitive, such as medical applications [9]. MEC involves the placement of small but powerful servers close to mobile users in the form of a distributed network and offers an effective solution for computation offloading in a ''smart'' way.

The traditional offloading schemes suffer from numerous problems including, but not limited to, the assumptions of unlimited computational power of servers, constant uplink and downlink network conditions for all users, and equal priority to every mobile user regardless of the energy requirements and network conditions [10], [11]. While these assumptions do not seriously affect the traditional MCC, they become the key factors in offloading decisions for MEC since edge devices also have limited computational capabilities.

The associate editor coordinating the review of this manuscript and approving it for publication was Yaser Jararweh.

A ''smart'' decision mechanism for achieving the maximum benefits of offloading to the edge devices is required for the optimal performance of the network.

A. NOVELTY AND CONTRIBUTIONS

In this paper, we propose a novel partial offloading scheme based on the fine-grained partial offloading framework. To the best of our knowledge, this is the first attempt to consider the energy consumption of UEs in the deep learning based modeling for partial offloading schemes in MEC. The remaining energy and energy consumption by each application component are very important parameters to be considered in offloading decisions, because the decision of the next component for offloading is directly dependent on these parameters. Until now most of the deep learning based approaches only consider the delay constraints and ignore the energy consumption of the UEs. However, we are generating data to train the Deep Neural Network (DNN), by considering the remaining energy of UEs, the energy consumed by previous component, local and cloud resources, varying network conditions, amount of data to transfer, and delays in communication. All these parameters are taken into account by our cost function in the execution of each application component either on the UE or on Mobile Edge Server (MES). The novelty of our proposed work in mathematical modeling improves the accuracy of offloading decisions and minimizes the cost and energy consumption of UE.

Our proposed scheme intelligently selects the optimal combinations of application components to offload, reducing the overall cost of execution per application. While making such decisions, this approach prioritizes users with urgent need, such as a dying battery, over others. Our contributions are summarized as follows:

- We provide an optimal decision-making mechanism for computational offloading in MEC. Our approach minimizes the overall cost of execution per application, considering the remaining energy of the UE, cost of local and server execution, previous offloading decisions and varying network conditions. It means our cost function considers all the important parameters in decision-making process for computational offloading. Most of the previous work does not consider the varying local execution cost [12] and finite cloud computation power of edge devices for deep learning based models.
- We provide a mathematical model for local execution cost under varying energy conditions and cost of offloading to the Cloudlet [10] under varying network conditions. We formulate an exhaustive decision-making process to find the optimal decision in a partial offloading scheme. This process calculates the cost for all possible policies for c components of a task (2^c possible decisions). The proposed work then selects the decision with minimum cost. We consider this decision as the optimal offloading decision.
- We propose a deep learning based algorithm, as an alternative to the exhaustive decision-making process. This algorithm can be trained over an exhaustive dataset (generated by our mathematical model) and then used for the said decision-making. Our algorithm takes varying local and network conditions as an input and learns the optimal decision policy from the exhaustive scheme. Once trained, this algorithm can be used as a decision-maker for offloading the specific components of the application to MES. The problem is formulated as a multi-component binary classification problem where each component should either be executed locally or offloaded to the Cloudlet. This alternative saves us from calculating cost for all 2^c offloading policies and choosing the best one for every application under varying conditions.

The remainder of this paper is organized as follows: Section II surveys the state-of-the-art. Section III describes the problem formulation, mathematical model for local execution and remote execution, and the decision-making process. Section III also provides the cost function, the optimization problem, and the algorithm design. Section IV discusses the simulation results, and finally, the paper concludes in Section V.

II. RELATED WORK

A lot of research work is done on computational offloading to improve the performance of UEs. However, different proposed offloading techniques have different goals. Computational offloading problems are generally divided into two categories: total offloading where all the computations are handed over to the MES [10], and partial offloading where only a subset of application components is offloaded to an edge server [13]. Orsini *et al.* [14] explain that partial offloading requires the calculations of computational cost for each application component and, thus, puts an additional strain on the computation resources as well as the energy reserves. However, such calculations can be used to intelligently decide the optimal set of components to be offloaded to minimize the amount of data transfer as well as reduce the latency and overall energy consumption. In our proposed work we consider partial offloading scheme. Because partial offloading reduces latency, energy consumption, and unnecessary transmission overhead as compared to total offloading scheme [12]. Al-Khafajiy *et al.* [15] present a collaborative edge offloading method, which permits fog nodes cooperation for big data processing, relying on pre-defined fog parameters. This approach deems effective in processing data at the edge level on-time due to the obvious reason that all necessary information about the fog nodes capabilities (i.e., processors) are known in advance. Yet, this approach overlooks the energy consumption of fog nodes, thereof, it is not an energy efficient one.

Mazouzi *et al.* [16] propose a computation offloading policy for multi-user and multi-cloudlet MEC environment.

Their main objective is to minimize the offloading cost function, which depends on execution time and energy consumption only. The computation resources of MES and radio resources are ignored in the cost function. Due to the heuristic approach, it will be very difficult to adapt to complex and dynamic applications. Using dynamic programming techniques, many researchers use Markov Decision Process (MDP) to obtain an optimal policy for computation offloading in MEC. However, it requires a fixed state transition probability matrix. Using the concept of MDP Liu *et al.* [17] propose a delay-optimal single-user task scheduling policies. However, the actual transition probability matrix is very difficult to obtain.

Li *et al.* [18] propose a deep reinforcement learning approach for the total offloading scheme. However, in reinforcement learning approaches the global minima may not be guaranteed due to its unsupervised learning nature. Therefore, in recent years supervised deep learning approaches are gaining much popularity in computational offloading in MEC. The main advantages of deep learning are high accuracy in decision making and high calculation speed for trained models. The trained deep learning algorithm can avoid the exhaustive calculations for finding the optimal solution. Cloud computing and MEC methods have been studied extensively in [17] and [18]. Machine learning for optimization problems in MEC has also been employed in [20] and [21]. However, the previous work based on machine learning mainly focuses on the total offloading scheme and considers only delay constraints in cost function.

Eom *et al.* [10] propose a computational offloading technique based on machine learning. They show the best scheduling decision is selected by their instance-based online offloading scheduler. Similarly Eom *et al.* [11] propose a machine learning-based mobile offloading scheduler (MALMOS) with online training. However, both of these techniques consider total offloading schemes and are based on the Local Area Network (LAN) and Wide Area Network (WAN). Also, these schedulers are not able to handle partial offloading because they do not consider the varying local and network conditions for selecting the optimal set of components to offload. Yu *et al.* [12] suggest a partial offloading scheme using a deep learning approach. However, the cost function does not consider the energy consumption of the UEs. The energy consumption of UEs is a very important parameter for the cost function. Therefore, a deep learning approach is required to consider the energy consumption of UEs and to obtain an energy-efficient and faster computational offloading scheme for MEC.

The previous work in this area has either assumed infinite edge server resources or infinite energy reserves for both mobile user and edge server [12]. Such decisions may not always be optimal since they do not take into full consideration the varying energy and network conditions. Machine learning has replaced many conventional methods in the areas of computer vision, speech recognition, natural language procession, etc. Deep learning has surpassed the performance of conventional machine learning (shallow learning) methods [22], [23] and is now being employed in almost every field of life [24]. Deep learning for communication networks has achieved wonderful results for runtime scheduling [25], [10], [11], and saves a lot of energy and time in decision-making applications with a pre-trained network. Deep learning algorithms can learn complex decision boundaries [26] and complicated patterns in the data and that motivates us to design a deep learning algorithm for smart offloading decision-making processes.

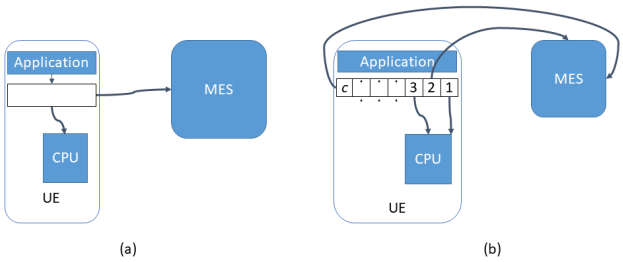


FIGURE 1. (a) Total offloading of an application to MES; (b) Partial offloading of an application to MES.

III. PROBLEM FORMULATION AND MODELING

This section first presents mathematical models for local execution and remote execution (offloading to MES). We also present the decision policies, cost function and its optimization in this section. In our proposed work, called (EEDOS), we consider the remaining energy parameter. To the best of our knowledge, this is the first work to consider the remaining energy of UE in such a mathematical model for DNN approach. In the deep learning based modeling of the partial offloading schemes, the remaining energy has not been considered previously.

A UE can execute many tasks either locally (using the CPU of UE) or remotely (offloading to MES). The coarse-grained approach (total offloading scheme) is that the whole task can either be executed locally or offloaded to MES, as shown in Figure 1(a). However, a more efficient approach is partial offloading, in which the UE can split a task into multiple components. Each component can be executed locally or offloaded to MES, as shown in Figure 1(b). All the notations used in this paper are given in Table 1.

We use the concept of a partial offloading scheme in this paper. Call graph [27] can be used to model the relation between multiple components as a linear directed Graph, $A = (C, D)$, where C denotes the set of components and D is the data required from one component to the next component.

A. LOCAL EXECUTION MODEL

Let the edge of the graph d_{c_0,c_1}, $(d_{c_0,c_1} \in D)$ represents the data of computation results between two components c_0 and c_1. Similarly d_{c_1,c_2} is the data of computation results between two components c_1 and c_2. d_{c_0,c_1} is the input data of c_1 and the output data of c_0. Similarly d_{c_1,c_2} represents the output

TABLE 1. Notations.

Notations	Meaning
A	Directed graph
B	Bandwidth
C	Set of components
$c \in C$	Components
D	Set of edge weights (input/output data)
$d \in D$	Edge weight
d	Distance between UE and MES
E_c	Energy consumed by c
E_t	Total Energy
E_r	Remaining energy of UE
F	Cost function
F_l	Local execution cost
F_r	Remote execution cost
f_s	CPU rate of MES
f_u	CPU rate of UE
g_{dl}	Desired bit error rate for downlink
g_{ul}	Desired bit error rate for uplink
h_{dl}	Downlink channel fading coefficient
h_{ul}	Uplink channel fading coefficient
M	Set of CPU cores in MES
$m \in M$	CPU cores allocated to UE
N	Set of subcarriers
N_0	Noise power
$n \in N$	Subcarriers allocated to UE
P	Set of decision policies
P^*	Optimal decision policy
$p_c \in P$	Decision policy
p_s	Transmission power of MES
p_u	Transmission power of UE
r_{dl}	Maximum achievable downlink data rate
r_{ul}	Maximum achievable uplink data rate
S	State vector
T_e	Execution delay
$T_l(c)$	Local execution delay
T_r	Reception delay
T_t	Transmission delay
t_d	Decoding delay
V	CPU clock cycles per byte
W_c	Workload for c
β	Path loss exponent
$\gamma_1, \gamma_2, \gamma_3, \gamma_4$	Unit balancing coefficients
$\xi_1, \xi_2, \ldots, \xi_6$	Weighting coefficients

data of c_1 and the input data of c_2. Let W_c is the workload which represents the weight of component c, measured in CPU cycles, represented as:

$$W_c = Vd_{c-1,c}, \tag{1}$$

where V is measured in cycles per byte (cpb) and it represents the number of clock cycles a microprocessor will perform per byte. The study about this value is presented in [28]. Now, if the component c is executed locally on the UE, the time required to complete the execution of workload W_c is given by:

$$T_l(c) = \frac{W_c}{f_u}, \tag{2}$$

where f_u represents the CPU rate of the UE. It is measured in Million Instructions Per Second (MIPS). Let the energy consumption due to this workload is E_c, and E_t represents the total energy of the UE, then the remaining energy for the next component $(c+1)$ is given by:

$$E_r = E_t - E_c, \tag{3}$$

where E_r is the remaining energy of the UE. It means that the remaining energy for the next component $(c+1)$ will be decreased by an amount of E_c, while the energy consumption E_c for a component c is directly proportional to the amount of data to be executed.

B. REMOTE EXECUTION MODEL

The UE can offload a component to MES for execution. If we consider the network deployment utilizes the orthogonal frequency division multiple access (OFDMA), then we can safely assume that the bandwidth B for transmission is divided into N subcarriers. Let the available subcarriers that will be allocated in each component execution period be $n \in 1, 2, 3, \ldots, N$. Similarly, at the MES, if the total number of available CPU cores are M, then $m \in 0, 1, 2, 3, \ldots, M$ denotes the number of CPU cores to be used in execution of a component by MES, where $m = 0$ means that the MES is busy and the offloading component is rejected for execution in MEC.

Similarly, for transmission, the maximum achievable uplink and downlink data rate for an additive white Gaussian noise (AWGN) channel can easily be derived as in [12],

$$r_{ul} = n\frac{B}{N}\log_2\left(1 + \frac{p_u|h_{ul}|^2}{\Gamma(g_{ul})d^{\beta}N_o}\right), \tag{4}$$

$$r_{dl} = n\frac{B}{N}\log_2\left(1 + \frac{p_s|h_{dl}|^2}{\Gamma(g_{dl})d^{\beta}N_o}\right). \tag{5}$$

We assume the same noise behavior in transmission for uplink and downlink. Here, B is the bandwidth, d denotes the distance between UE and MES, N_0 is the noise power, p_u and p_s refer to the transmit power of UE and MES, respectively, h_{ul} and h_{dl} are the channel fading coefficient for uplink and downlink, respectively, β is the path loss exponent, while g_{ul} and g_{dl} are the required bit error rate for uplink and downlink, respectively. The $\Gamma(g_{ul}) = \frac{-2\log(5g_{ul})}{3}$) represents the SNR margin to meet the required bit error rate with QAM constellation. We have considered Rayleigh-fading in our scenario. In the offloading process, the energy consumption of UE, due to transmission and reception, depends upon the amount of data to be transferred.

C. DECISION POLICIES

We assume a time interval of t, called decision period, in which a component can be executed completely either locally or by MES. The UE sends information of its input data $d_{c-1,c}$ for component c and channel quality to MES at the beginning of each decision period. In LTE standards, the Buffer Size Report (BSR) and Channel Quality Information (CQI) messages are used for this process [29]. After the reception of these messages, the MES allocates n number of sub-carriers (communication resources) and m number of CPU cores (computation resources) to the UE according to the sent information by UE and the currently available resources in MES. The parameters, namely, n, m, d, and the energy consumption of UE for a single component E_c can be

used to introduce a composite state $S = (c, d, n, m, E)$. The UE calculates a cost for local execution and remote execution according to the given composite state. For execution of component c, the UE takes a decision p_c to execute locally ($p_c = 0$) on the UE or to offload to MES ($p_c = 1$) on the basis of cost value. For example, a three-component task may have the decision scheme as $P = [1, 0, 1]$. It means that the first and the last components are offloaded to MES and the second component is executed on the mobile UE. Thus, we can conclude that the state space of UE and possible action space is given by:

State Space:

$$\{S = (c, d, n, m, E) \in \mathcal{S} | c \in \mathcal{C}, d \in \mathcal{D}, n \in \mathcal{N}, m \in \mathcal{M}\}.$$

Action Space:

$$\mathcal{P} = \{p_c \in \{0, 1\}, c \in \mathcal{C}\}.$$

The decision vector (offloading scheme) for a task depends on the cost value of each component in the task. These cost values are calculated from a cost function for each component. In this paper, we propose a new cost function depending on the time delay, energy consumption and other resources, such as n sub-carriers and m CPU cores in MES. Time delay means the time required for processing, transmission, and reception of data for a single component. The next subsection presents the proposed cost function, while the usefulness of our new cost function will be demonstrated in Section IV.

D. COST FUNCTION

The cost function $F(S, p_c)$ for a current decision period is expressed as:

$$F(S, p_c) = \begin{cases} F_l(S), & p_c = 0 \\ F_r(S), & p_c = 1, \end{cases} \tag{6}$$

where $F_l(S)$ is the cost for local execution and $F_r(S)$ is the cost for remote execution. $F_l(S)$ depends on time delay and energy consumption for the execution of a component. Mathematically, it can be written as:

$$F_l(S) = \gamma_1 T_l(c) + \gamma_2 E_c, \tag{7}$$

$$\gamma_1 = \frac{\xi_1}{T_{max}}, \tag{8}$$

$$\gamma_2 = \frac{\xi_2}{E_{max}}, \tag{9}$$

where ξ_1 and ξ_2 are the weighting coefficients by which we can change the contribution of time delay and energy consumption in the cost function respectively. T_{max} is the maximum time for the execution of a single component by UE. Similarly E_{max} is the expected maximum energy consumption of UE for a single component execution. Therefore we can say that γ_1 and γ_2 are the unit balancing and weighting coefficients for the cost function. $T_l(c)$ is the execution time (given by (2)) of component c executed by UE. E_c is the energy consumption of UE due to the execution of component c.

Similarly $F_r(S)$ can be modeled as:

$$\begin{aligned} F_r(S) = \gamma_3[(1 - p_{c-1})T_t(d_{c-1,c}) + T_e(d_{c-1,c}) \\ + T_r(d_{c,c+1}) + t_d] + \gamma_4[E(d_{c,c+1}) \\ + E(d_{c-1,c})(1 - p_{c-1})] + \xi_5 K(m) \\ + \xi_6 K(n), \end{aligned} \tag{10}$$

$$\gamma_3 = \frac{\xi_3}{T_D}, \tag{11}$$

$$\gamma_4 = \frac{\xi_4}{E_{max}}, \tag{12}$$

where ξ_3, ξ_4, ξ_5, and ξ_6 are the weighting coefficients by which we can change the contribution of each parameter in the cost function. T_D is the maximum time for a component to be executed on MES, which is also called deadline time. γ_3 and γ_4 are called the unit balancing and weighting coefficients. t_d is the decoding delay for UE to decode the computation results sent by MES, and p_{c-1} represents the previous decision. The first term in (10) represents the time delay for transmission data $d_{c-1,c}$ (input data of component c), and is given by:

$$T_t(d_{c-1,c}) = \frac{d_{c-1,c}}{r_{ul}}. \tag{13}$$

If the decision for previous component $p_{c-1} = 1$, it implies the previous component has executed by MES and MES has the data $d_{c-1,c}$ as output of component $c-1$. Therefore, we do not need to transmit this data to MES. This is why we multiply $T_t(d_{c-1,c})$ by $(1 - p_{c-1})$. Thus, we have zero delay for the transmission process if the previous decision is offloaded to MES.

The second term in (10) represents the execution time delay for the workload W_c by m CPU cores in MES, and it can be written as:

$$T_e(d_{c-1,c}) = \frac{W_c}{mf_s}, \tag{14}$$

where m is the number of CPU cores allocated for component c to execute in MES, and f_s is the CPU rate of MES.

The third term in (10) represents the time delay for the reception of the output data of component c, $(d_{c,c+1})$. It is given by:

$$T_r(d_{c,c+1}) = \frac{d_{c,c+1}}{r_{dl}}. \tag{15}$$

In (10) $E(d_{c-1,c})$ denotes the energy consumption of UE due to transmission of $d_{c-1,c}$ data to MES. Therefore, we have multiplied this term by $(1 - p_{c-1})$ for the same reason as explained for transmission delay term. When the previous decision $p_{c-1} = 1$, we do not need the transmission process. The term $E(d_{c,c+1})$ represents the energy consumption of UE due to reception of the output data $d_{c,c+1}$ of component c, where $K(m)$ is the cost for computation resources and is given by:

$$K(m) = \frac{m}{M}. \tag{16}$$

Here m is the number of CPU cores used in execution of component c, M is the total number of CPU cores in MES, and $K(n)$ is the cost due to radio resources given by:

$$K(n) = \frac{n}{N}, \tag{17}$$

where n is the number of sub-carriers allocated for transmission and N is the total number of sub-carriers.

E. OPTIMIZATION PROBLEM

From the above mathematical modeling, we can generate a data set with optimal offloading decision vectors as labels. Therefore, we can use the supervised deep learning approach to train a DNN [13]. We aim to find the optimal offloading policy for an application to execute. We have a decision matrix $P = p_c,\ c = 1, 2, 3 \ldots |C|$, where $|C|$ is the total number of components of a task to be executed. Therefore, the order of decision matrix is $2^{|C|} \times |C|$. For each task, there are $2^{|C|}$ possible offloading schemes. It means each row of matrix P is a possible offloading scheme or decision vector. We need to find the optimal offloading scheme $P^*, P^* \in P$. Note that the optimal offloading scheme is one of the row vector of matrix P with minimum cost. Therefore, we can write the objective function as:

$$P^* = \arg_P \min \left(\sum_{c \in \mathcal{C}} F(S, p_c) \right). \tag{18}$$

The optimization problem is obviously non-convex and the complicated data patterns make rule-based algorithms not realistic. The main idea of our solution, instead of applying traditional optimization approaches, is to compute all possible offloading schemes according to our mathematical model and then select the optimal offloading scheme (minimum cost) as a training data set. Using the data set, we design a deep learning algorithm for a smart offloading decision-making process. More specifically, we train a DNN with the data generated by our mathematical model. After training, we obtain a trained DNN for computing the optimal and energy efficient offloading scheme. The improved accuracy and minimized energy consumption and cost are the main advantages of our EEDOS, which will be demonstrated in Section IV.

F. ALGORITHM DESIGN

We use a deep supervised learning algorithm for optimal offloading decision making, which is a multi-label classification framework[18]. The input of our model is the composite state of all components and the output will be an energy-efficient and optimal offloading scheme, which is a row vector of matrix P.

The first phase of this approach focuses on generating optimal decision policies for various local and network conditions through the mathematical modeling presented in Section IV, Subsection C. Since these decision policies are calculated from an exhaustive $2^{|C|}$ policy set, the approach is computationally intensive but, 100% accurate. We generate 10 datasets of size 1000 samples to 10,000 samples by generating data for the state variables from their respective distributions, calculating the cost for all possible decision policies, and then selecting the decision policy with the minimum cost. The state vectors as well as their corresponding optimal offloading decision policy are stored in a matrix with columns representing the features, and the last column containing the label (100% accurate decision vector), for training the DNN.

DNNs are becoming more popular due to the supremacy of their accuracy when trained with big data [30]. We can generate huge datasets using our mathematical model with the optimal decision vector. By increasing the number of components, the input data parameters and the length of the decision vector increases due to which the calculation of output from input parameters becomes hard and complex. Therefore, we use a DNN to present our proposed EEDOS.

Algorithm 1 Train Deep Network Using EEDOS Data

BEGIN

Require: $d \in [100, 500]$, $V \in [4000, 12000]$, $E \in [1, 100]$, $m \in [0, 16]$, $n \in [1, 256]$

Ensure: $P^* = \arg_P \min \left(\sum_{c \in \mathcal{C}} F(S, p_c) \right)$

  while $i \leq datasize$ **do**

    while $j \leq c$ **do**

      $W_c \leftarrow Vd_{c-1,c}$

      $S_i \leftarrow \{c_i, d_i, m_i, n_i, E_i\}$

      if $p_c = 0$ **then**

        $F_l(S) \leftarrow \gamma_1\, T_l(c) + \gamma_2\, E_c$

      else

$$F_r(S) \leftarrow \gamma_3[(1 - p_{c-1})T_t(d_{c-1,c}) + T_e(d_{c-1,c}) + T_r(d_{c,c+1}) + t_d] + \gamma_4[E(d_{c,c+1}) + E(d_{c-1,c})(1 - p_{c-1})] + \xi_5\, K(m) + \xi_6\, K(n)$$

      end if

$$F(S, p_c) = \begin{cases} F_l(S), & p_c = 0 \\ F_r(S), & p_c = 1 \end{cases}$$

    end while

    $F_i \leftarrow F(S, p_c)$

    $P^* \leftarrow \arg_P \min \left(\sum_{c \in \mathcal{C}} F(S, p_c) \right)$

  end while

  divide data into Training, Validation and test sets

  $network \leftarrow pattennet([64, 64])$

  while $i \leq datasize$ **do**

    $net_{trained} = train(network, S, label)$

  end while

  Test performance on test set

  efficiency $\leftarrow$ number of correct policies /total policies

  repeat for different dataset sizes

END

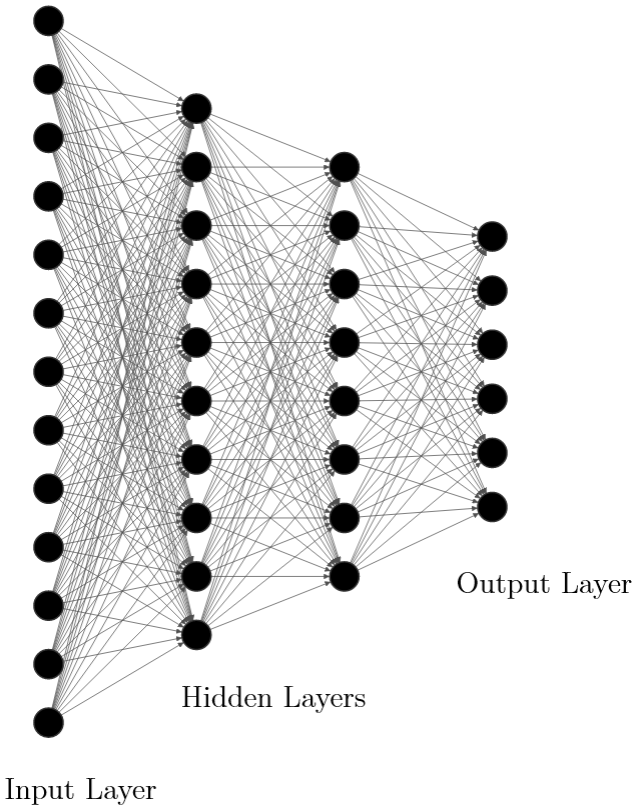


FIGURE 2. A deep neural network with 2 hidden layers.

It is important to note that this exhaustive scheme requires extensive calculations and the objective of training a neural network on this exhaustive dataset is to avoid these calculations for all future decisions with high accuracy.

After obtaining state vectors and their corresponding decision policies, a deep network can be trained on this dataset. The network takes the state vector as the input and the optimal decision policy as the label. The data is divided randomly into training, validation, and test sets of ratios 75%, 15%, and 15%, respectively. The dimension of the input is the number of components times the length of the state vector, and the label is a vector of length equal to number of components. We use *pattennet* deep learning network [31] with 2 hidden layers and 64 neurons in each hidden layer (through cross validation) as shown in Figure 2.

By increasing the number of neurons and hidden layers, the complexity of the network increases. However, 2 hidden layers with 64 neurons give the maximum accuracy in our dataset. Through numerical analysis, we have established that a higher number of neurons and hidden layers yields the same accuracy. Thus, we have chosen 2 hidden layers with 64 neurons in each layer. By increasing the input data parameters and the number of components, one can increase the number of neurons and hidden layers [32]. Sigmoid activation functions [1], [33] are used in the hidden layers whereas softmax is used in the output layer for classification [34]. We use the standard cross-entropy as the loss function [35]. The network architecture is shown in Figure 3.

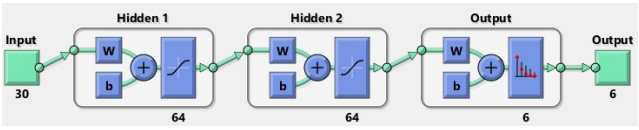


FIGURE 3. Network architecture.

The trained network can then be used for testing the performance on the test set, and can be used for all future decisions. The performance of the network is tested on the unseen 15% data we kept separate for the testing phase. Having a pre-trained network takes away the computation load of calculating cost for 2^c possible policies and selecting the optimum policy. Generating exhaustive datasets of various sizes trains the network over an extensive range of local and network conditions, so the future decisions can be made by giving the state vector to the trained network and obtaining the decision policy for that state as output. The performance of this network is compared against various other offloading schemes, as shall be demonstrated in the next section.

IV. RESULTS AND DISCUSSIONS

The mathematical model is used as a benchmark for the performance evaluation of other offloading policies. The decision policies calculated through extensive mathematical model (EMM) are assumed to be 100% accurate. The performance of all other schemes is calculated as the ratio of the number of decision policies that match the mathematical model and the total number of decision policies. We used MATLAB (R2019a) on Intel Core i7 CPU @3.4GHz for simulations. In simulations, we consider 6 components of an application. It means the proposed work divides each task of UE into six components which will be executed sequentially either on UE or on MES. The input and output data of the components follow the uniform distribution with $d \in [100, 500]$. Similarly the CPU cycles per byte (V), the available energy of UE (E_r), sub-carriers (n), and CPU cores (m) are also considered as uniformly distributed as: $V \in [4000, 12000]$, $E_r \in [1100]$, $n \in [1, 256]$, and $m \in [0, 16]$. All these random variables are independent for different components.

The proposed EEDOS redefines the local execution cost and the remote execution cost, considering the remaining battery of the UE and the amount of energy consumed in each component execution. From literature, we take three offloading policies and compare their performance against EMM. These three policies are explained as follow:

- Total offloading scheme (TOS) [12] is a coarse-grained approach, which offloads all the computation load to MES. Since all the computations are being offloaded, this strategy does not require any decision making about offloading policies.
- Random offloading scheme (ROS) [12] randomly selects application components regardless of the amount of data transfer required, network conditions, and local and remote resources.

TABLE 2. Network parameters.

Parameter	Value	Parameter	Value
B	3	ξ_5	0.1
β	2	ξ_6	0.1
d	10	g_{ul}	10^-3
f_s	10^10	p_u	0.01
f_u	10^5	M	16
γ_1	0.5	N	256
γ_2	0.5	N_0	5×10^-5
γ_3	0.5	p_s	0.1
γ_4	0.3	t_d	0.1

TABLE 3. Offloading accuracy.

Scheme	Accuracy	Scheme	Accuracy
EMM	100%	EEDOS	72%
TOS	22%	ROS	1%
DOS	61%		

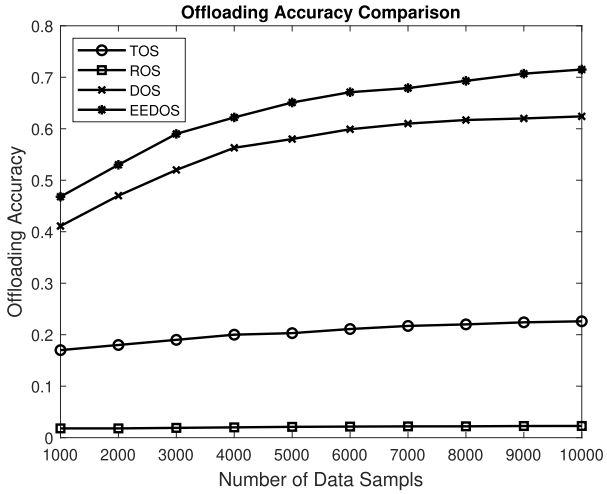


FIGURE 4. Comparison of offloading accuracies of TOS, ROS, DOS, and EEDOS.

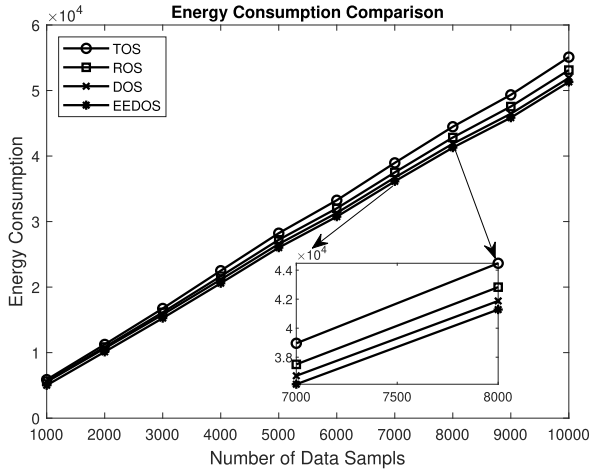


FIGURE 5. Comparison of energy consumption of TOS, ROS, DOS, and EEDOS.

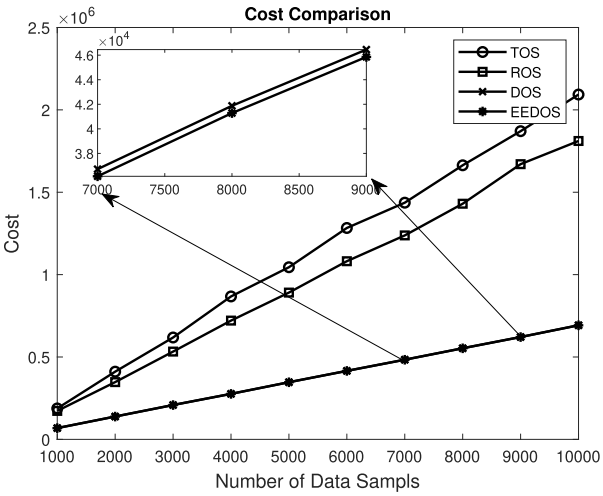


FIGURE 6. Comparison of overall cost of TOS, ROS, DOS, and EEDOS.

- Deep learning based offloading scheme (DOS) [12] takes into account the network conditions and amount of data transfer required and uses a deep learning network with 2 hidden layers and 128 neurons in each layer. This approach does not consider the remaining battery of the UE, and the amount of energy consumed in each application component execution.

Table 2 provides various network parameters used in this paper. Most of the parameters used were the same as for [12], however, the CPU rates (f_u and f_s) were redefined based on the assumption that MES has higher CPU rates than the UE. Table 3 provides the offloading accuracy for different offloading schemes for a dataset of 10,000 samples. ROS has the worst accuracy whereas EEDOS has the best. EEDOS has considerably higher accuracy than DOS.

Figure 4 shows the comparison of offloading accuracy for different decision policies against the size of the dataset used. As shown in the figure, the accuracy of all the schemes improves by using a larger dataset, but EEDOS outperforms all other schemes. This is because we consider the energy consumption parameter in our mathematical model due to which the input information becomes stronger and, consequently, the result of training based on the DNN is improved.

An important parameter to evaluate the decision policies is the amount of energy consumed by overall application execution. Compared with EEDOS, the approach in [12] consumes more energy. For the sake of comparison, the amount of energy consumed is calculated and shown in Figure 5 for all the approaches. EEDOS reduces the amount of energy consumption by 3%, 6%, and 10% when compared to DOS, ROS, and TOS, respectively. This is because our mathematical model has parameters that also make the cost function dependent upon energy consumption.

Figure 6 shows the comparison of the total cost of application execution for a different number of data samples used. EEDOS executes the applications at minimum overall cost when compared with other methods. DOS closely follows EEDOS but never outperforms it, because DOS ignores the energy consumption parameter in the cost function.

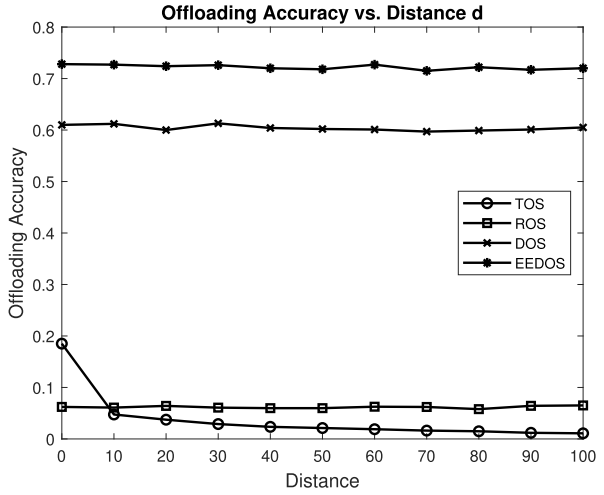


FIGURE 7. Comparison of offloading accuracy with respect to distance for TOS, ROS, DOS, and EEDOS.

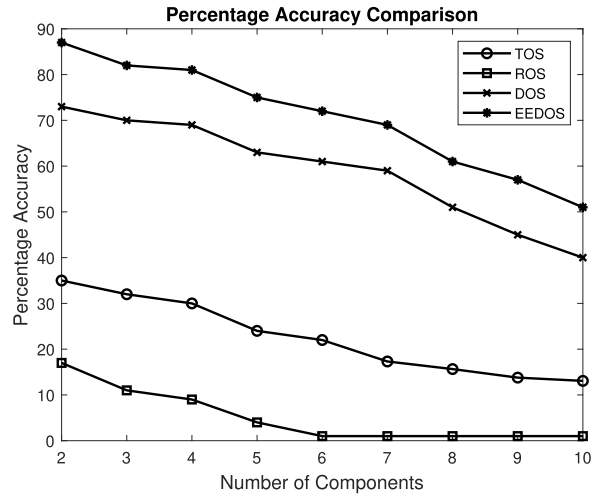


FIGURE 8. Comparison of offloading accuracy with respect to distance for TOS, ROS, DOS, and EEDOS.

DOS mainly focuses on the delay constraints while EEDOS also considers the energy consumption dependency on the cost function. It is important to note that although energy consumption was not taken into account originally by the DOS approach, the overall cost was recalculated for the sake of this plot, taking account of the energy consumption for all the schemes.

Figure 7 displays the comparison of offloading accuracies with respect to distance from MES. As the UE moves away from MES, TOS is affected the most because of all components offloading to MES at various distances. However, EEDOS considers the distance dependency on cost function in generating data set for DNN. Therefore, EEDOS maintains its accuracy and allows the UE to move around while offloading computation-intensive components of an application to MES.

Figure 8 illustrates the results of offloading accuracy for all the schemes as the number of components varies. As expected, the offloading accuracy decreases as the number of components of a task increases. Increasing the number of components raises the complexity of the decision boundaries for the DNN, hence the decline in accuracy for the same number of samples. However, the performance of EEDOS remains consistently higher than other schemes for a small number of components as well as for a large number of components.

Comparison of offloading accuracy of different schemes with respect to number of data samples, number of application components, and distance of the UE from MES is provided in the previous section. We conclude from the figures that the accuracy of the deep learning approaches increases by using a larger dataset, while the accuracy of the TOS and ROS schemes is not affected by the size of data since there is no learning involved. Dividing an application into more number of components decreases the accuracy of all the schemes. The accuracy of TOS and ROS decreases because the increment in the number of components increases the number of offloading policies and the likelihood of the optimal policy being TOS (or ROS) is decreased. The accuracy of the deep-learning-based approaches is decreased because of the decision boundaries becoming more complex with the increase in the number of components. The distance only affects the TOS scheme since at zero distance the accuracy of TOS is likely to be more than its accuracy at other distances (all components are more likely to be offloaded if the distance is very small, making the cost of offloading very small). All the simulation results show a better performance of EEDOS because of its comprehensive mathematical model. For the cost function, all the important and realistic parameters are considered. The accuracy of DNN is improved because the data has a clearer underlying pattern. The proposed approach comprehensively models the real environment and is better suited for implementation in practical scenarios). Our proposed work selects the offloading policy with minimum cost. It means that the optimal offloading policy selected by our proposed work will consume minimum energy and take a minimum delay for the execution of a task. A limitation of our proposed model is that it considers a single user and the application components are assumed to be executed sequentially (linear call graph model). If an application has several components being executed in parallel, the results of which are being used subsequently by other components, or if the application has callbacks or loops through previous components (non-linear call graph), then our proposed approach cannot handle that scenario. All loops, callbacks, and parallel executions are merged into a single component that will either be executed locally or offloaded to MES and the overall application is always divided into components that can be executed sequentially. Such division can sometimes result in some very large application components, and the offloading scheme can be further optimized by allowing the subdivisions of such components into smaller components with loops, callbacks, and parallel execution. However, that is beyond the scope of this paper and should be considered

as an extension of this work in the form of a proposed future research challenge.

V. CONCLUSION AND FUTURE WORK

In this paper, we demonstrated a novel approach to intelligently offload application components to cloudlets using comprehensive mathematical modeling and deep learning approach named as EEDOS. We modeled a cost function for the application execution on UEs as well as on the cloud server under the constraints of energy consumption, network conditions, delays, and available computation resources. Due to the consideration of these parameters in the cost function, our proposed work (EEDOS) is more comprehensive and high accuracy for optimal decision making for the offloading problem in MEC. Through an exhaustive analysis of cost, accuracy, and energy consumption we showed that EEDOS is more comprehensive and accurate. To avoid the exhaustive calculation and make the decision-making process faster we trained a DNN. The data set for training the DNN is generated from the derived mathematical model in which we consider all the important parameters in the cost function derivation. We achieved upto 3% decrease in the energy consumption and 2% decrease in the overall cost as compared with the previous methods. We also achieved 12% increase in the accuracy of the DNN, with fewer neurons.

This proposed work considers a single-user scenario, in which the total task of a single user is divided into multiple components. These multiple components are then executed sequentially. Therefore, we consider a linear directed graph for problem formulation. For future work, the multi-user scenario can be considered to generate the training data set for DNN. For multi-user scenario the problem can be considered as a non-linear graph.

REFERENCES

[1] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, Jun. 2014, pp. 68–81.

[2] T. Langlotz, D. Wagner, A. Mulloni, and D. Schmalstieg, "Online creation of panoramic augmented reality annotations on mobile phones," *IEEE Pervas. Comput.*, vol. 11, no. 2, pp. 56–63, Feb. 2012.

[3] I. Al Ridhawi, M. Aloqaily, Y. Kotb, Y. Jararweh, and T. Baker, "A profitable and energy-efficient cooperative fog solution for IoT services," *IEEE Trans. Ind. Informat.*, to be published.

[4] M. Quwaider and Y. Jararweh, "A cloud supported model for efficient community health awareness," *Pervas. Mobile Comput.*, vol. 28, pp. 35–50, Jun. 2016.

[5] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[6] T. Dreibholz, S. Mazumdar, F. Zahid, A. Taherkordi, and E. G. Gran, "Mobile edge as part of the multi-cloud ecosystem: A performance study," in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Pavia, Italy, Mar. 2019, pp. 59–66.

[7] X. Fu, S. Secci, D. Huang, and R. Jana, "Mobile cloud computing [Guest Edotorial]," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 61–62, Mar. 2015.

[8] I. Al Ridhawi, M. Aloqaily, B. Kantarci, Y. Jararweh, and H. T. Mouftah, "A continuous diversified vehicular cloud service availability framework for smart cities," *Comput. Netw.*, vol. 145, pp. 207–218, Nov. 2018.

[9] M. Aloqaily, I. Al Ridhawi, H. B. Salameh, and Y. Jararweh, "Data and service management in densely crowded environments: Challenges, opportunities, and recent developments," *IEEE Commun. Mag.*, vol. 57, no. 4, pp. 81–87, Apr. 2019.

[10] 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 6th Int. Conf. Utility Cloud Comput.*, Dresden, Germany, Dec. 2013, pp. 17–25.

[11] H. Eom, R. Figueiredo, H. Cai, Y. Zhang, and G. Huang, "MALMOS: Machine learning-based mobile offloading scheduler with online training," in *Proc. 3rd IEEE Int. Conf. Mobile Cloud Comput., Services, Eng.*, San Francisco, CA, USA, Apr. 2015, pp. 51–60.

[12] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach," in *Proc. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

[13] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[14] G. Orsini, D. Bade, and W. Lamersdorf, "CloudAware: A context-adaptive middleware for mobile edge and cloud computing applications," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FAS W)*, Augsburg, Germany, Sep. 2016, pp. 216–221.

[15] M. Al-Khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, and A. Hussain, "IoT-Fog optimal workload via fog offloading," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, Zurich, Switzerland, Dec. 2018, pp. 359–364.

[16] H. Mazouzi, N. Achir, and K. Boussetta, "DM2-ECOP: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment," *ACM Trans. Internet Technol. (TOIT)*, vol. 19, no. 2, p. 24, Apr. 2019.

[17] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[18] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[19] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 84–91, May. 2016.

[20] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[21] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, and A. Abraham, "Hybrid job scheduling algorithm for cloud computing environment," in *Proc. 5th Int. Conf. Innov. Bio-Inspired Comput. Appl. (IBICA)*. Ostrava, Czech Republic: Springer, 2014, pp. 43–52.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[23] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 160–167.

[24] G. Bradski, A. Kaehler, and V. Pisarevsky, "Learning-based computer vision with intel's open source computer vision library," *Intel Technol. J.*, vol. 9, no. 2, pp. 119–130, May 2005.

[25] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. ASP-DAC*, Jan. 2018, pp. 129–134.

[26] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* Montreal, QC, Canada: MIT Press, 2014, pp. 2924–2932.

[27] B. G. Ryder, "Constructing the call graph of a program," *IEEE Trans. Softw. Eng.*, vol. SE-5, no. 3, pp. 216–226, May 1979.

[28] M. Yang, Y. Wen, J. Cai, and C. H. Foh, "Energy minimization via dynamic voltage scaling for real-time video encoding on mobile devices," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Ottawa, ON, Canada, Jun. 2012, pp. 2026–2031.

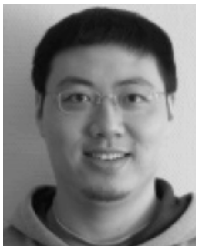
[29] L. Lei, Z. Zhong, C. Lin, and X. Shen, "Operator controlled device-to-device communications in LTE-advanced networks," *IEEE Wireless Commun.*, vol. 19, no. 3, pp. 96–104, Jun. 2012.

[30] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electron*, vol. 8, no. 3, p. 292, Mar. 2019.

[31] W. Zhou, S. Newsam, C. Li, and Z. Shao, "PatternNet: A benchmark dataset for performance evaluation of remote sensing image retrieval," *ISPRS-J. Photogramm. Remote Sens.*, vol. 145, pp. 197–209, Nov. 2018.

[32] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.
[33] D. Gao and Y. Ji, "Classification methodologies of multilayer perceptrons with sigmoid activation functions," *Pattern Recognit.*, vol. 38, no. 10, pp. 1469–1482, 2005.
[34] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *Proc. Multiple Classifier Syst. (MCS)*, T. Windeatt and F. Roli, Eds., Jun. 2003, pp. 125–134.
[35] P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Ann. Oper. Res.*, vol. 134, no. 1, pp. 19–67, 2005.

ZAIWAR ALI received the B.S. degree in electronics engineering from the COMSATS Institute of Information Technology at Abbottabad, Pakistan, in 2012, and the M.S. degree in electronics engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, in 2015, where he is currently pursuing the Ph.D. degree in electrical engineering with the Telecommunications and Networking (TeleCoN) Research Laboratory. From 2013 to 2015, he was a Graduate Assistant with the Faculty of Electrical Engineering, GIK Institute, where he has been a Research Associate, since 2016. His current research interests include stochastic processes, the IoT, machine learning, edge computing, and wireless networks. He was a recipient of the Highest Level of Merit Scholarship.



LEI JIAO received the B.E. degree in telecommunication engineering from Hunan University, China, the M.E. degree in communication and information system from Shandong University, China, in 2005 and 2008, respectively, and the Ph.D. degree in information and communication technology from the University of Agder (UiA), Norway, in 2012, where he is currently an Associate Professor with the Department of Information and Communication Technology. His current research interests include mobile communications and artificial intelligence.

THAR BAKER received the Ph.D. degree in autonomic cloud applications from LJMU, in 2010. He is currently a Reader in cloud engineering and the Head of the Applied Computing Research Group (ACRG), Faculty of Engineering and Technology, Liverpool John Moores University (LJMU), U.K. He has published numerous refereed research articles in multidisciplinary research areas including big data, algorithm design, green and sustainable computing, and energy routing protocols. He has been actively involved as a member of Editorial Board and Review Committee for a number of peer-reviewed international journals, and is on program committee for a number of international conferences. He was a Senior Fellow of the Higher Education Academy (SFHEA), in 2018. He is currently an Associate Editor of *Future Generation Computer System*. He is an Expert Evaluator of Newton Funds, ICTFund, and British Council.

GHULAM ABBAS received the B.S. degree in computer science from the University of Peshawar, Pakistan, in 2003, and the M.S. degree in distributed systems and the Ph.D. degree in computer networks from the University of Liverpool, U.K., in 2005 and 2010, respectively. From 2006 to 2010, he was a Research Associate with the Liverpool Hope University, U.K., where he was associated with the Intelligent and Distributed Systems Laboratory. Since 2011, he has been with the Faculty of Computer Sciences and Engineering, GIK Institute of Engineering Sciences and Technology, Pakistan. He is currently an Associate Professor and the Director of the Huawei Network Academy. He is also a Co-Founding Member of the Telecommunications and Networking (TeleCoN) Research Laboratory, GIK Institute. His current research interests include computer networks and wireless and mobile communications. He is a Fellow of the Institute of Science and Technology, U.K., British Computer Society.

ZIAUL HAQ ABBAS received the M.Phil. degree in electronics from Quaid-i-Azam University, Pakistan, in 2001, and the Ph.D. degree from the Agder Mobility Laboratory, Department of Information and Communication Technology, University of Agder, Norway, in 2012. In 2012, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Minnesota, USA. He is currently an Assistant Professor with the GIK Institute of Engineering Sciences and Technology. His current research interests include energy efficiency in hybrid mobile and wireless communication networks, 5G and beyond mobile systems, mesh and ad hoc networks, traffic engineering in wireless networks, the performance evaluation of communication protocols and networks by analysis and simulation, the quality-of-service in wireless networks, green wireless communication, and cognitive radio.

SADIA KHAF received the B.E. degree from the National University of Sciences and Technology, Islamabad, in 2015, and the M.S. degree in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2018. From 2015 to 2018, she was a Research Assistant with IONOLAB, Turkey. She is currently with the Faculty of Electrical Engineering, GIK Institute of Engineering Sciences and Technology, Pakistan, as a Research Associate. She is also a Graduate Teaching Assistant with Bilkent University. Her current research interests include machine learning, edge computing, and radio resource management. She was a recipient of highest level of merit scholarship.