Research Article

Efficient Deep Learning Approach for Computational Offloading in Mobile Edge Computing Networks

Xiaoliang Cheng,¹ Jingchun Liu,² and Zhigang Jin ³

¹School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China

²Department of Radiology and Tianjin Key Laboratory of Functional Imaging, Tianjin Medical University General Hospital, Tianjin 300052, China

³School of Electrical and Information Engineering, Tianjin University, Weijin Road Campus: No. 92 Weijin Road, Nankai District, Tianjin 300072, China

Correspondence should be addressed to Zhigang Jin; zgjin@tju.edu.cn

Received 13 November 2021; Revised 1 January 2022; Accepted 13 January 2022; Published 22 February 2022

Academic Editor: Hasan Ali Khattak

Copyright © 2022 Xiaoliang Cheng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The fifth-generation mobile communication technology is broadly characterised by extremely high data rate, low latency, massive network capacity, and ultrahigh reliability. However, owing to the explosive increase in mobile devices and data, it faces challenges, such as data traffic, high energy consumption, and communication delays. In this study, multiaccess edge computing (previously known as mobile edge computing) is investigated to reduce energy consumption and delay. The mathematical model of multidimensional variable programming is established by combining the offloading scheme and bandwidth allocation to ensure that the computing task of wireless devices (WDs) can be reasonably offloaded to an edge server. However, traditional analysis tools are limited by computational dimensions, which make it difficult to solve the problem efficiently, especially for large-scale WDs. In this study, a novel offloading algorithm known as energy-efficient deep learning-based offloading is proposed. The proposed algorithm uses a new type of deep learning model: multiple-parallel deep neural network. The generated offloading schemes are stored in shared memory, and the optimal scheme is generated by continuous training. Experiments show that the proposed algorithm can generate near-optimal offloading schemes efficiently and accurately.

1. Introduction

The rapid development of fifth-generation (5G) mobile communication technology services in recent times has prompted the emergence of compute-intensive applications, such as intelligent driving, ultra-high-definition video, and mobile crowdsensing [1]. The 5G technology is largely characterised by extremely high data rate, low latency, massive network capacity, and ultrahigh reliability; hence, it requires the appropriate architecture to function efficiently. However, traditional centralised cloud computing network architecture is unable to meet the requirements of the 5G software architecture because it is limited by excessive link load and delays in real-time response [2, 3]. Consequently, the European Telecommunications Standards Institute

(ETSI) has proposed a new concept of mobile edge computing (MEC) [4–6]. Users can migrate compute-intensive and delay-sensitive applications from local to edge server to solve the problems of limited computing resources and battery energy [7]. Concurrently, the edge servers can precache some content required by users to reduce access delay and improve user experience [8]. Currently, the research methods of MEC are mainly divided into two categories: offloading method for reducing time delay and offloading method for reducing energy consumption. To reduce time delay, some researchers proposed the IHRA scheme for computations in multiuser situations [9]. It considered the rich computing resources of cloud computing and the low transmission delay characteristics of MEC. The offloading technique enabled part of the

computing tasks to be offloaded to the user terminal device for execution, thereby reducing the execution delay of delay-sensitive applications by 30%. Some studies have proposed the LODCO algorithm [10], a dynamic computing offloading algorithm based on Lyapunov optimisation theory. This method optimises the unloading decision based on two aspects, task running delay and task running fault, to ensure the minimisation of processing delay in the offloading task and the success rate in the data transmission process. Consequently, the probability of offloading failure is reduced. The simulation results showed that the algorithm has excellent advantages in reducing time delay and could shorten the execution time of offloading tasks by 64%. However, this method only focused on the delay condition and failed to consider the energy consumption of the mobile terminal device during the offloading. As a result, the terminal equipment may not operate properly owing to insufficient power, which can have negative impact on the user experience. Therefore, further study is required to discover an offloading technique that can minimise energy consumption. Several studies have been conducted on the optimisation of energy consumption to solve offloading problem in different environment scenarios. Some studies have adopted the artificial fish swarm algorithm to design the offloading scheme for energy consumption optimisation under time delay constraints [11]. Although this method effectively reduces the energy consumption of the task data transmission network by considering the link status in the network, it is characterised by high complexities. In a previous study, a particle swarm task-scheduling algorithm was designed for multiresource matching to minimise energy consumption of edge terminal devices [12]. Furthermore, some studies have investigated partial offloading of computing tasks to minimise the energy consumption of mobile devices. For wireless devices (WDs) with a separable task [13], an energy-saving optimisation problem was proposed, and a greedy algorithm was used to solve it. However, this method only reduced the energy consumption of the mobile terminal and could not minimise delays in the task execution time.

To minimise the calculation delay and energy consumption simultaneously, especially in the environment of multiple MEC servers and multiple terminal users, it is appropriate to realise the offloading of the computational tasks of wireless mobile devices [14]. The use of software defined networks (SDN) for MEC has been adequately exploited to efficiently and effectively deal with the data offloading problem. Specifically, the SDN paradigm transforms the communication networks into a programmable world, where a centralised entity, namely, SDN controller, acquires a global view of the communication links and manages the network traffic efficiently and dynamically [15, 16]. However, the centralised control mode and openness of SDN can pose potential risk to the security of the controller.

Due to these limitations, it avoids the network risk induced by the centralised control mode, while considering the minimisation of computation delay and energy consumption. This study designs a deep learning offloading algorithm that focuses on reducing energy consumption and time

delays. The algorithm is composed of two components, offloading scheme and deep learning, and aims to solve the problem of selective offloading of mobile application components. The innovation of the algorithm is mainly reflected in the following:

- (1) When we establish the system utility model in the MEC network, we mainly weight them based on the two operators of delay communication and residual energy and then obtain the cost of offloading.
- (2) The other innovation of this study is a new type of deep learning model, which uses multiple-parallel deep neural networks [17–19], stores several offloading schemes of computing tasks in a shared memory, and substitutes them into the new deep learning model. After repeated iterations, the optimal offloading scheme is finally obtained. Compared with the traditional single deep neural network, this novel deep learning model has the advantage of facilitating the optimal edge offloading scheme. Thus, the edge offloading convergence is greatly improved.

Experiments show that the scheme has higher accuracy, lower energy consumption, and lower communication delay. The MEC architecture based on the new deep learning model proposed in this study is shown in Figure 1.

2. System Model and Problem Formulation

2.1. System Model. In this study, an efficient mobile edge offloading framework is designed. Suppose that, in the MEC framework, only one edge server, one small cell, and N WDs exist, where N WDs are represented by a set $N = \{1, 2, \dots, N\}$. We assume that the computational task of each wireless terminal device contains C independent subtasks, which are recorded as $A = \{Application_1, Application_2, \dots, Application_c\}$. The work queue of these subtasks is stored in FIFO. For the subtasks in any WD, the offloading scheme is obtained through the offloading algorithm designed in this study. The offloading scheme is expressed in binary; that is, $P_t \in \{0, 1\}$. $P_t = 0$ indicates that WD subtasks are executed locally, and $P_t = 1$ indicates that WD subtasks are offloaded to the edge server for execution. P is the offloading scheme of all subtasks in WD, which is recorded as

$$P = \{P_t \in \{0, 1\}, t \in T\}. \quad (1)$$

The edge computing model of the WDs network is shown in Figure 2.

2.2. Local Execution Model. Assuming that there is a subtask t in WD, to solve the cost function of its local execution, the computational delay and energy consumption must be first determined. Afterwards, the two following operators must be used to solve it. The data size of any WD subtask is marked as d_{nt} . W_t represents the computing resources occupied by subtask t , V represents the number of clocks occupied by the CPU to execute one byte, and f_u represents the CPU operating frequency of the WD [20]. Therefore, the computational delay is

FIGURE 1: The MEC architecture based on the new deep learning model.

FIGURE 2: The edge computing model of the WDs network.

$$T_{l(c)} = \frac{W_t}{f_u}. \quad (2)$$

W_t is determined by d_{nt} and V , which denote the data size of the subtask and the number of clocks occupied by the CPU executing one byte, respectively.

$$W_t = V \times d_{nt}. \quad (3)$$

During local execution, the energy consumed by executing each byte is recorded as e^l , and the energy consumed by executing subtasks is obtained using the following equation:

$$E_{nt}^l = d_{nt} \times e^l. \quad (4)$$

The cost function of local execution is obtained from the computational delay and energy consumption of local execution, which is recorded as $C_l(s)$:

$$C_l(s) = \gamma_1 T_{l(c)} + \gamma_2 E_{nt}^l. \quad (5)$$

In equation (5), γ_1 and γ_2 are weighting coefficients, which are linear with the maximum execution time ($T_{\max}$) and maximum energy consumption ($E_{\max}$) in the task, respectively.

2.3. Mobile Edge Computing. To solve the problems of limited computing resources and battery energy, compute-intensive and delay-sensitive computing tasks in WDs are offloaded locally to the edge server for execution to improve the efficiency of the system. Similar to local execution, computational delay ($T_{r(c)}$) and energy consumption (E_{nt}^r) are crucial operators that affect edge offloading. This study considers the impact of computing resources and communication resources on $T_{r(c)}$ and E_{nt}^r . When the influence of all variables is fully estimated, the most reasonable cost function can be formulated and the optimal offloading scheme can be obtained. The process of offloading the subtasks in WDs to MES involves the following steps. First, the data is offloaded to MES. Second, the task is executed in MES. Subsequently, the third step requires a downlink to a WD, and the last step is to complete decoding in WD.

Consequently, four time constants are generated: the upload time of the subtask, T_{up} ; the execution time of the subtask executing the task in the MES, T_{ex} ; the time of the subtask down to the WD, T_{down} ; and the decoding time, T_d .

$$T_{r(c)} = T_{up} + T_{ex} + T_{down} + T_d. \quad (6)$$

Equation (6) shows that, in the MES, the allocation of communication resources, namely, bandwidth, directly affects T_{up} and T_{down} of the subtask, and the allocation of computational resources, that is, the allocated CPU, directly affects T_{ex} of the subtask. They are denoted by the following equations:

$$\begin{aligned} T_{up} &= \frac{d_{nt}}{r_{ul}}, \\ T_{down} &= \frac{d_{nt}}{r_{dl}}, \\ T_{ex} &= \frac{W_t}{mf_s}. \end{aligned} \quad (7)$$

In equation (7), r_{ul} and r_{dl} represent the transmission rates of the subtask from uplink to MES and downlink to WD, respectively. These two factors are closely related to the allocation of communication resources. W_t represents the number of clocks required to execute subtask T , M represents the number of CPUs allocated to the subtask (the number of M depends on the data structure of the subtask), and f_s represents the CPU working frequency of MES. Bandwidth allocation directly affects the uplink time and downlink time of subtasks [21, 22].

The effects of bandwidth and number of CPUs on the computational delay and transmission delay in offloading are shown in Figure 3.

In this study, emphasis is placed on the method for solving r_{ul} and r_{dl} . The orthogonal frequency division multiple access (OFDMA) technology is used to complete the allocation of communication resources. This technology divides the channel into several orthogonal subchannels and converts the high-speed data signal into parallel low-speed subchannels. Because the orthogonal signals can be separated using correlation technology at the receiving end, it reduces the mutual interference among subchannels and completes the allocation of communication resources. The total network bandwidth B is decomposed into k subcarriers ($K \in 1, 2, 3, \dots, K$), and each subtask t is allocated to several subcarriers. In signal processing, considering that the additive Gaussian white noise signal (AWGN) is easy to analyse and approximate, the actual noise signal can be approximately processed with Gaussian white noise in a certain frequency band for the analysis of the noise performance of the signal processing system [23, 24]. Therefore, the maximum data transmission rates of uplink and downlink are

$$\begin{aligned} r_{ul} &= n \frac{B}{N} \log_2 \left(1 + \frac{P_u |h_{ul}|^2}{\Gamma(g_{ul}) D^\beta N_0} \right), \\ r_{dl} &= n \frac{B}{N} \log_2 \left(1 + \frac{P_u |h_{dl}|^2}{\Gamma(g_{dl}) D N_0} \right). \end{aligned} \quad (8)$$

Here, B is the network bandwidth, D is the distance from WD to MES, N_0 is the power consumption of noise, P_u is the power consumption generated by WD transmission data, and h_{ul} is the attenuation coefficient of uplink channel. β is the path loss index, g_{ul} is the bit error rate of uplink data transmission, and $\Gamma(g_{ul}) = ((-2 \log(5 g_{ul})) / (3))$ represents the stability margin of signal-to-noise ratio, which aims to meet the target of bit error rate.

FIGURE 3: The effects of bandwidth and the number of CPUs on the computational delay and transmission delay in offloading. (a) The effect of bandwidth on transmission delay. (b) The effect of CPU on the computational delay.

Similar to the analysis method for calculating the delay, the energy consumption E_{nt}^r generated by offloading the subtasks to the MES is mainly composed of the energy consumption E_{ex} of the MES execution task and the energy consumption E_d of the decoding in the WD. The energy consumption generated by the data uplink and downlink is ignored.

$$E_{nt}^r = E_{ex} + E_d. \quad (9)$$

The cost function of remote offloading is obtained by computational delay $T_r(c)$ and energy consumption E_{nt}^r of edge offloading, which is denoted as $C_r(s)$:

$$C_r(s) = \gamma_3 T_r(c) + \gamma_4 E_{nt}^r. \quad (10)$$

In equation (10), γ_3 and γ_4 are linearly related to T_{max} and E_{max} , respectively. T_D represents the maximum execution time of the subtask in the MES, and E_M represents the maximum energy consumption in the task.

2.4. Cost Function. WDs require a certain amount of time and energy to perform computing tasks. In this paper, a system utility model S is introduced (equation (11)) and the delay and energy are minimized by analyzing the model. The model contains four key parameters that affect the utility of the system: the number of bytes D of the subtask, the allocated communication resources, K subcarriers, and the allocated computing resources- m CPUs. The energy consumption E_{nt} is employed to execute the task.

$$\{S = (d, k, m, E_{nt}) \in S | d \in D, k \in K, m \in M\}. \quad (11)$$

Based on the established system utility model and the cost analysis of local offloading and edge offloading, the cost function of performing computing tasks is obtained as

$$C(S, P_t) = \begin{cases} C_l(s), & P_t = 0, \\ C_r(s), & P_t = 1. \end{cases} \quad (12)$$

In equation (12), P_t represents the binary offloading scheme of each subtask in WDs. $P_t = 0$ indicates that the subtask of WD is executed locally, and $P_t = 1$ indicates that the subtask of WD is to be offloaded to the edge server for execution.

2.5. Problem Formulation. This study uses the system model S and the cost function $C(S, P_t)$ to find the optimal offloading scheme P_o , which minimises the cost of executing tasks and obtains the optimal solution for MEC.

To obtain the optimal scheme, we input the system utility parameters of the computing task into the DNN [25]. After the identification of the hidden layer, we obtain the offloading scheme at the output layer. The offloading scheme passes through the back-propagation and iteration of the neural network [26], namely, gradient descent algorithm. We obtain the optimisation scheme P_o that is closest to the actual value. The mathematical expression is as follows:

$$P_o = \arg_p \min \left(\sum_{t \in T} C(S, P_t) \right). \quad (13)$$

In equation (13), $\sum_{t \in T} C(S, P_t)$ represents the total cost of performing the calculation task and P represents the most

optimal offloading scheme. We use the DNN to obtain the most approximate value P_o , making P_o infinitely close to P .

3. EDLO Algorithm Design

DNN has a high accuracy rate when training big data [27]. The second aim of this paper is to use multiple-parallel deep neural networks to train samples and obtain the optimal offloading scheme [28, 29]. The input layer of the model is the vector state S^* of several subtasks in the computing task.

$$S^* = \begin{bmatrix} d_1 & d_2 & d_3 & \dots & d_k \\ k_1 & k_2 & k_3 & \dots & k_k \\ m_1 & m_2 & m_3 & \dots & m_k \\ E_1 & E_2 & E_3 & \dots & E_K \end{bmatrix}_{4 \times k}. \quad (14)$$

The output layer is the unloading plan P . This study divides the computing task of WD into several subtasks. The scheme for executing the task is denoted as $P = P_t$, and $c = 1, 2, 3, \dots, |k|$. Each subtask has two possibilities of offloading; that is, it can be executed locally ($P_t = 0$) or using MES ($P_t = 1$). Therefore, they are $2^{|k|}$ offloading schemes for t subtasks.

The detailed steps of energy-efficient deep learning-based offloading (EDLO) designed in this study are listed in Table 1.

3.1. EDLO Algorithm Analysis. The multiple-parallel deep neural network designed in this study increases the number of hidden layers and neuron sums in the network. The number of hidden layers is set to 2, and the number of neurons in each layer is 256 [35]. Based on the corresponding numerical analysis, it was observed that an increase in the number of hidden layers and neurons prompted a corresponding increase in computation accuracy [30]. The sigmoid function is used as the activation function and the value range, which is used for the output of hidden layer neurons, is $[0, 1]$. Finally, the cross entropy loss function is used as the loss function [31]. These matchings are made because the use of the sigmoid function can avoid the loss of mean square error during gradient descent [32, 33], where learning rate is reduced. In other words, the learning rate can be controlled by the output error. The learning process of the DNN is shown in Figure 4.

4. Experiment and Comparative Analysis

4.1. Experiment. In this study, the EDLO algorithm is designed for the optimal offloading scheme problem. The following experiments and simulations were performed to verify the performance of the algorithm. We assume that the edge offloading framework includes three WDs and the computing task of each WD consists of five subtasks, which can be executed by either the edge server or the wireless terminal. The data byte size of each subtask is randomly distributed within 10 M to 20 M [34]; that is, $D \in [10 \text{ M}, 20 \text{ M}]$. The number of clocks occupied by CPU executing one byte is given as $V \in [2000, 10000]$. The available energy E_t in

the WD is 1500 J [35]. The energy consumption of transmitting and receiving data by WDs is $11.3 \times 10^6 \text{ J/bit}$ [36]. The number of CPUs in the MES is denoted as $m \in [0, 10]$. The number of subcarriers is denoted as $n \in [1, 512]$.

In Figure 5, the number of DNNs is different, which affects the convergence of the EDLO algorithm. Because of the input of different state space S , the evaluation function of the system produces different output results. This study uses the EDLO algorithm as the benchmark to ensure that the computation standard of the input quantity is unified. As shown in Figure 5, as the learning steps increase, the EDLO algorithm gradually converges to one. When using ten DNNs, the convergence rate is approximately 98% after 1000 learning steps. When using one DNN, the computation accuracy rate is 75% and cannot converge. When using three DNNs, the computation accuracy rate is approximately 97%. Specifically, the computation accuracy rate is increased by 21% compared to when one DNN is used, and the algorithm converges. As the number of DNNs increases, the convergence speed increases accordingly. The influence of different DNNs on the computation convergence speed is shown in Figure 5.

The EDLO algorithm can generate an optimal unloading decision in less than 0.2 s. Its computational time is a close approximation of the different numbers of DNNs. The computational delay statistics of different numbers of DNNs are shown in Figure 6.

Different learning rates and different data storage spaces have different effects on the convergence speed of EDLO algorithm. The higher the set learning rate is, the faster the convergence speed of EDLO algorithm is. However, as shown in Figure 7, an increase in the learning rate increases the probability of obtaining the optimal solution locally as compared to the probability of obtaining the entire optimal solution. Therefore, it is appropriate to choose the best learning rate according to the actual situation.

Figure 8 shows that different memory spaces affect the convergence speed of the EDLO algorithm. Based on the experiments conducted, it was observed that local convergence is often the first to be established. Therefore, to balance the convergence speed and overall performance of the algorithm in the MES network, we use 2048 memory space to achieve the fastest convergence speed. Figure 8 depicts the effect of the size of memory space on the convergence speed of the algorithm.

4.2. Comparative Analysis. To accurately evaluate the performance of the EDLO algorithm, this study compares the performance of the EDLO algorithm with the processing schemes of four computing tasks. Ten DNNs are used in the EDLO algorithm, the learning rate is defined as 0.01, and the system memory is 2048. Consequently, the EDLO algorithm achieves optimal performance. The four schemes used for comparison are as follows:

- (1) Total local process (TLP): all computing tasks of WD are performed locally.
- (2) Total edge process (TEP): all computing tasks of WDs are executed by the edge server MES [37, 38].

TABLE 1: Algorithm design.

Algorithm: train deep network using EDLO data

BEGIN

(1): **Input:** Input the state vector of the computing task, $S^* = (d, k, m, E_{nt})$ (2): **Output:** Optimal offloading decision, $p_o = \arg_p \min(\sum_{t \in T} C(S, P_t))$ (3): **Initialization:**

(4): Initialize the DNNs with random parameters, weight &paranoid item

(5): Empty the memory structure

(6): **while** $i \leq \text{datasize}$ **do** **while** $j \leq c$ **do** $W_t \leftarrow V_{dc-1;c}$ $Si \leftarrow \{d_i; k_i; m_i; E_i\}$ **if** $p_t = 0$ **then** $C_l(s) = \gamma_1 T_l(c) + \gamma_2 E_{nt}^l$ **else** $C_r(s) = \gamma_3 T_r(c) + \gamma_4 E_{nt}^r$ **end if** $CS, P_t = \begin{cases} C_l(s), & P_t = 0, \\ C_r(s), & P_t = 1, \end{cases}$ **end while** $Ci \leftarrow C(S, P)$ $P^* \leftarrow \arg_p \min(\sum_{t \in T} C(S, p_t))$ **end while**(7): **Store into the memory structure;**

(8): Train the DNNs and update weight &paranoid item of DNNs;

(9): The optimal unloading scheme is solved

End

FIGURE 4: The learning process of the DNN.

FIGURE 5: The influence of different DNNs on the computation convergence speed.

FIGURE 6: The computational delay statistics of different numbers of DNNs.

FIGURE 7: The effect of different learning rates on convergence.

FIGURE 8: Different memory spaces affect the convergence speed of the EDLO algorithm.

TABLE 2: Network parameters.

Parameter	Value	Parameter	Value
B	100	M	10
β	2	g_{ul}	10^{-3}
D	10	g_{dl}	10^{-3}
f_s	10^{10} cycle/s	P_u	0.01
γ_1	0.5	N_0	5×10^{-5}
γ_2	0.5	T_d	0.1
γ_3	0.5	γ_4	0.3

FIGURE 9: Communication delay between different algorithms in offloading process (T_{ud}). FIGURE 10: Computational delay between different algorithms in offloading process (T_{ex}).

FIGURE 11: Comparison of the total computational delay of EDLO and other algorithms.

FIGURE 12: Comparison of the offloading accuracy of EDLO and other algorithms.

- (3) Random offloading scheme (ROS): regardless of the state space S of the computing task, the subtasks are randomly arranged on the edge server or local WDS for processing.
- (4) Deep learning-based offloading (DLO) scheme: the input state of the algorithm does not include the remaining energy of the WDS and the energy

TABLE 3: Offloading accuracy.

Scheme	Accuracy (%)
EDLO	99.67
TEP	62.16
ROS	0.05
DLO	91.23
TEP	38.01

FIGURE 13: Comparison of energy consumption of EDLO and the other algorithms.

FIGURE 14: Comparison of cost of EDLO and the other algorithms.

consumption of executing the program [39]. The neural network of the algorithm contains two hidden layers, each with 128 neurons.

In this paper, the EDLO algorithm and the other four algorithms are compared and tested based on the following: (1) the impact of communication resource on offloading delay, that is, the impact of bandwidth B on $T_{ud} = T_{up} + T_{down}$; (2) the impact of computing resources on edge computing latency, that is, the impact of the number of CPUs m on T_{ex} ; (3) the total computational delay generated during task execution, that is, the total delay consumption of task execution; (4) computational accuracy; (5) energy consumption; and (6) computational cost. Based on the results, it was proved that the EDLO algorithm provides optimal performance. Network parameters in this study are listed in Table 2.

Owing to the TLP, the computing and communication resources of MES are not used for local data processing. In this study, when the network bandwidth is 100 Mbps, the number of cores in MES is 10 and the computing task size is 15 M. The effects of the four algorithms on communication and computational delays are compared with those of the EDLO algorithm. The communication and computational delays of EDLO algorithm are 93 ms and 102 ms, respectively. The delay time of EDLO algorithm is substantially lower than those of the TLP and ROS algorithms and

moderately lower than that of the traditional deep learning algorithm, DLO. Compared to the DLO, the communication delay time and computational delay time of EDLO are 30.76% and 31.58% lower, respectively. Thus, the computational efficiency of the proposed algorithm is significantly improved because we consider the impact of bandwidth and CPU on delay. The communication delay T_{ud} and computational delay T_{ex} of all the algorithms in the offloading process are shown, respectively, in Figures 9 and 10.

Figure 11 presents a comparison of the total computational delay of EDLO and other algorithms. Based on the experimental results, the total computational delay of EDLO is 860 ms, which is 44.19% lower than that of the DLO algorithm. Thus, the real-time performance of computation is improved greatly.

Figure 12 shows a comparison of the offloading accuracy of EDLO and other algorithms. Based on the results, the offloading accuracy of EDLO is significantly higher than those of TIP and TEP and slightly higher than that of DLO. However, the offloading accuracy of the ROS algorithm is approximately zero. Similar to Figure 11, Table 3 shows that the EDLO algorithm is reliable.

Figure 13 shows the comparison of energy consumption of different algorithms. The energy consumption generated by EDLO is lower than that generated by DLO, which is

9.82% and significantly lower than those of the other three traditional data processing methods.

Figure 14 presents a comparison of cost of EDLO and the other algorithms. Based on the experimental results, the final cost of the EDLO algorithm is 38.32%, which is lower than that of the DLO and significantly lower than those of the other three traditional data processing methods.

It can be observed that the EDLO Algorithm exhibits superior performance compared to the three schemes of TLP, TEP, and ROS. The performance of the EDLO algorithm is higher than that of the DLO algorithm.

5. Conclusion

In this study, we propose a deep learning offloading algorithm, EDLO, to reduce energy consumption and time delay. To realise this, we consider the energy consumption, computing delay, computing resources, and communication resources of the applied system. Distributed management avoids the hidden dangers of network security caused by centralised control. The algorithm introduces multiple-parallel DNNs, which can generate the optimal solution without manually labelling the data, and the numerical results verify the accuracy and performance of the algorithm. We envisage that our proposed framework can be applied in subsequent advances of MEC network to achieve optimised real-time offloading.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Xiaoliang Cheng and Jingchun Liu have contributed equally to this work and share first authorship.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (81601467, 81601472, and 81871327).

References

- [1] D. Zhang, H. Xiong, L. Wang, and G. Chen, "Crowdrecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint," in *Proceedings of the ACM Conference on Ubiquitous Computing*, pp. 703–714, ACM, New York, NY, USA, 2014.
- [2] X. Fu, S. Secci, D. Huang, and R. Jana, "Mobile cloud computing [guest editorial]," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 61–62, 2015.
- [3] 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 *Proceedings of the 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, pp. 59–66, Pavia, Italy, March 2019.
- [4] Etsi, "Mobile edge computing (mec)," 2014, <http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing>.
- [5] S. Yu, R. Langar, W. Li, and X. Chen, "Coalition-based energy efficient offloading strategy for immersive collaborative applications in femto-cloud," in *Proceedings of the IEEE International Conference on Communications, (ICC'16)*, Kuala Lumpur, Malaysia, May 2016.
- [6] S. Yu, R. Langar, and X. Wang, "A d2d-multicast based computation offloading framework for mobile edge computing," in *Proceedings of the IEEE Global Communication Conference, (GLOBECOM'16)*, Washington, DC, USA, December 2016.
- [7] M. Aloqaily, I. A. Ridhawi, H. B. Salameh, and Y. Jararweh, "Data and service management in densely crowded environments: challenges, opportunities, and recent developments," *IEEE Communications Magazine*, vol. 57, no. 4, pp. 81–87, 2019.
- [8] E. Ahmed, A. Ahmed, I. Yaqoob et al., "Bringing computation closer toward the user network: is edge computing the solution?" *IEEE Communications Magazine*, vol. 55, no. 11, pp. 138–144, 2017.
- [9] Z. Ning, P. Dong, and X. Kong, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2018.
- [10] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, 2016.
- [11] H. Zhang, J. Guo, L. Yang, L. Xi, and J. Hong, "Computation offloading considering front haul and backhaul in small-cell networks integrated with MEC," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 115–120, IEEE, Atlanta, GA, USA, May 2017.
- [12] Xu Jia, Li Xue-Jun, and R.-M. Ding, "Energy efficient multi-RE-source computation off loading strategy in mobile edge computing," *Computer Integrated Manufacturing Systems*, vol. 25, no. 4, pp. 954–961, 2019.
- [13] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Communications*, vol. 15, no. 11, pp. 149–157, 2018.
- [14] J. Wang and D. Li, "Adaptive computing optimization in software-defined network-based industrial internet of things with fog computing," *Sensors*, vol. 18, no. 8, p. 2509, 2018.
- [15] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633, 2019.
- [16] 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 *Proceedings of the 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, pp. 59–66, Pavia, Italy, Mar. 2019.
- [17] F. Giust, L. Cominardi, and C. Bernardos, "Distributed mobility management for future 5G networks: overview and analysis of existing approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 142–149, 2015.
- [18] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: a comprehensive survey," *IEEE*

- Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.
- [19] “IEEE network special issue: exploring deep learning for efficient and reliable mobile sensing,” 2017, <http://www.comsoc.org/netmag/cfp/exploring-deep-learning-efficient-and-reliable-mobile-sensing>.
 - [20] R. Socher, Y. Bengio, and C. D. Manning, “Deep learning for NLP (without magic),” in *Tutorial Abstracts of ACL*, 5, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012.
 - [21] B. G. Ryder, “Constructing the call graph of a program,” *IEEE Transactions on Software Engineering*, vol. SE-5, no. 3, pp. 216–226, 1979.
 - [22] 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 *Proceedings of the IEEE International Conference on Communications*, pp. 2026–2031, Ottawa, ON, Canada, Jun. 2012.
 - [23] S. Yu, X. Wang, and R. Langar, “Computation offloading for mobile edge computing: a deep learning approach,” in *Proceedings of the Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, pp. 1–6, October 2017.
 - [24] L. Lei Lei, Z. Zhangdui Zhong, C. Chuang Lin, and X. Xuemin Shen, “Operator controlled device-to-device communications in LTE-advanced networks,” *IEEE Wireless Communications*, vol. 19, no. 3, pp. 96–104, Jun. 2012.
 - [25] N. F. Hordri, A. Samar, S. S. Yuhaniz, and S. M. Shamsuddin, “A systematic literature review on features of deep learning in big data analytics,” *International Journal of Advances in Soft Computing and Its Applications*, vol. 9, no. 1, 2017.
 - [26] H. Park, J. Sun, S. Pekarek et al., “Real-time model predictive control for shipboard power management using the ipa-sqp approach,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2129–2143, 2015.
 - [27] J. Li, H. Gao, T. Lv, and Y. Lu, “Deep reinforcement learning based computation offloading and resource allocation for mec,” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, Barcelona, Spain, April 2018.
 - [28] C. You, K. Huang, H. Chae, and B. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411.
 - [29] H. Zhang, H. Liu, J. Cheng, and V. C. M. Leung, “Downlink energy efficiency of power allocation and wireless backhaul bandwidth allocation in heterogeneous small cell networks,” *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1705–1716, 2018.
 - [30] W. Zhou, S. Newsam, C. Li, and Z. Shao, “PatternNet: a benchmark dataset for performance evaluation of remote sensing image retrieval,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 145, pp. 197–209, 2018.
 - [31] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005.
 - [32] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, “Learning to optimize: training deep neural networks for wireless resource management,” in *Proceedings of the 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–6, Sapporo, Japan, July 2017.
 - [33] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, “Towards wearable cognitive assistance,” in *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 68–81, Bretton Woods NH, USA, June 2014.
 - [34] Y. Cao, T. Jiang, and C. Wang, “Optimal radio resource allocation for mobile task offloading in cellular networks,” *IEEE Network*, vol. 28, no. 5, pp. 68–73, 2014.
 - [35] I. Loshchilov and F. Hutter, “Online batch selection for faster training of neural networks,” 2015, <https://arxiv.org/abs/1511.06343>.
 - [36] G. Alain, A. Lamb, C. Sankar, A. Courville, and Y. Bengio, “Variance reduction in SGD by distributed importance sampling,” 2015, <https://arxiv.org/abs/1511.06481>.
 - [37] H. Park, J. Sun, S. Pekarek et al., “Real-time model predictive control for shipboard power management using the ipa-sqp approach,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2129–2143, 2015.
 - [38] G. E. Dahl, D. Dong Yu, L. Li Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.
 - [39] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, “Intelligent dynamic data offloading in a competitive mobile edge computing market,” *Future Internet*, vol. 11, no. 5, p. 118, 2019.