Journal of Computers Vol. 32 No. 3, 2021, pp. 289-299
doi:10.3966/199115992021063203021

An Edge Resource Offloading Algorithm Model Based on Deep Learning

Ying-zhan Kou[1], Cai-sen Chen[1*], Yang-xia Xiang[2], Fang Liu[1]

[1] Center of Exercise and Training, Academy of Army Armored Force, Beijing 100072, China
caisenchen@163.com

[2] Department of Information and Communication Army Academy of Armored Forces, Beijing, 100072, China
kyzh0323@sina.com, elephant_187@163.com, 3074002466@qq.com

Received 17 April 2021; Revised 1 May 2021; Accepted 17 May 2021

Abstract. With regard to the limited computing resource and the constraints of time delay of mobile edge computing (MEC) server, how to offload the complex computing tasks to mobile edge computing servers reasonably to carry on the data storage and calculation processing so as to both shorten completion time and reduce terminal energy consumption is the significant research contents of resource offloading algorithm. In this paper, a resource offloading algorithm aimed at the single cell multi-user scenario in edge computing with deep learning theory is proposed, considering the case that computing resources can be divided arbitrarily to decide whether to offload or not, in which the system models are established respectively taking delay and energy consumption as optimization objectives. The simulation results show that the resource offloading delay and energy consumption of this algorithm are significantly improved compared with the traditional resource offloading algorithm.

Keywords: mobile edge computing, resource offloading, deep learning

1 Introduction

With the development of technology, the intelligent terminal has become an indispensable part of modern life. Especially after the rise of 5G technology, people began to carry out new services such as high-definition video live broadcast and augmented reality on intelligent devices to facilitate our life, in which case, the traditional operation mode with cloud data center as the core needs to carry more and more data per unit time. On the one hand, because the business data interaction needs to be transmitted through the core network, it will produce a great load pressure on the core network in the peak period of the network. On the other hand, it will produce a large network delay between the relatively distant intelligent devices and the cloud data center, which seriously affects the user experience.

The concept of mobile edge computing was proposed to meet the above challenges. The small-scale data center is deployed in network edge nodes, such as base stations and wireless access points) to marginalize and localize computing resources and cache resources. Compared with traditional cloud data center computing, MEC has the following advantages: First, it can provide cloud computing services for mobile users in nearby areas, shorten the transmission distance effectively reduce the network delay; Second, there is no need to offload all the data to the cloud computing center to save communication consumption and reduce the congestion pressure of the network center; Third, it ca protect user's privacy better to complete the computing task on the server close to the edge node.

Although MEC enhances the computing power of intelligent devices and relieves the network pressure of the core network, how to make a reasonable offload decision and resource allocation to balance the load of each network node has become an important problem to be considered in MEC environment due

* Corresponding Author

to the limited computing power of the edge server.

Compared with the traditional machine learning technology, deep learning only needs to transfer the data directly to the network, which avoids a lot of feature processing process. At the same time, it can automatically extract new features for different problems. Through multi-layer stacking, it can express the input information hierarchically and train it layer by layer, which is more suitable for the processing of large and complex data. Deep learning has played an significant role in the Internet of Things (IoT) services as a big analysis tool at present. Because the processing of edge computing is performed in the edge layer, only the intermediate data or results need to be transferred from the device to the cloud server. Therefore, the combination of deep learning technology and MEC can further reduce the data transmission and reduce the network pressure. Therefore, an edge resource offloading strategy is developed in this paper, which makes use of the advantages of deep learning speed, good adaptability, strong learning ability and fast processing speed to reasonably allocate computing tasks, so as to solve the problem of what to unload, how much to unload and where to unload. Under the condition of constraining energy consumption cost, it can effectively reduce the delay of intelligent devices and service nodes, and improve the utilization of the whole MEC platform.

2 Related Work

With the continuous development of information technology and the explosive growth of data in information network, the application of edge computing has gradually become the mainstream, and the resource offloading of MEC has also been widely concerned and studied in recent years. A task offloading and resource allocation algorithm based on data security combined with deep reinforcement learning is proposed under multiple constraints in reference [1]. Compared with the classical algorithm, it effectively improves the success rate of task offloading and task successful execution rate, and better meets the QoS requirements of users. In [2], task offloading and resource allocation in MEC environment and cloud edge collaborative computing environment are studied respectively. In order to solve the problem of task offloading and resource allocation in MEC environment, an adaptive algorithm based on deep Q-learning is proposed. The algorithm has the ability of self-learning, which can determine whether the task needs to be offloaded and allocate the appropriate computing nodes for the task. During the algorithm training process, it continuously learns to improve the accuracy of decision-making. Aiming at the cloud edge collaborative environment, an adaptive task offloading and resource allocation algorithm with resource and reliability constraints is proposed while considering the reliability of computing nodes. In [3], a DRL based dynamic computing offload scheduling algorithm for complex task queue is designed, which jointly solved “where” and “when” to offload each task in the task queue, so as to obtain the optimal long-term tradeoff between task execution delay and energy consumption in this complex environment. By comparing with six baseline algorithms, the excellent performance and stability of the proposed algorithm are verified. [4] proposed a dynamic offloading and resource allocation scheme to minimize the execution delay of all tasks aiming at the single cell multi-user scenario and the case network transmission and computing offloading model. Then, aiming at the multi-user multi-cell scenario and the resource constrained situation of MEC service nodes, a joint optimization scheme of computing offloading, user access and resource allocation is proposed, which can finally achieve the goal reducing the total task execution delay of all users in the network under the condition of making full use of limited resources. The optimal computing offload problem is modeled as a Markov decision process in [5], and a candidate network optimization ECOO algorithm (Edge Computing Optimize Offloading) based on DDPG is proposed. According to the queue state, energy queue state and channel quality between mobile users and BS, the offloading decision is made. Simulation results show that ECOO algorithm is superior to some deep reinforcement learning algorithms in energy consumption and delay, and it is more effective in dealing with high-dimensional problems and the effect is remarkable. In [6], the computational power consumption of deep learning task is modeled based on the architecture of deep neural network dedicated accelerator chip Eyeriss, and a three-stage group sparse beamforming framework (GSBF) is proposed. Through careful design of computational task priority, redundant computing tasks at the base station are deleted as much as possible, optimizing the whole network power consumption including transmission power loss and calculation power loss.

Based on above research, compared with the assumption of prior distribution or heuristic algorithm, deep learning has the characteristics of self-learning and self-adaptive, which can solve more complex,

high latitude and more realistic task scenarios, and single cell multi-user scenario is a more common scenario model. Therefore, this paper focuses on the edge computing resource offloading scheme based on deep learning in this scenario.

3 System Model

In this section, we will elaborate the modeling process of MEC resource offloading scheduling optimization model based on deep learning, mainly from two aspects: the application scenario and the establishment of the model.

3.1 System Architecture

A typical MEC system architecture is shown in Fig. 1. MEC servers are deployed to the edge of network to provide low latency computing service for mobile users, which allocates special software and hardware service resources for each user, and isolates resources based on Virtualization Technology (such as virtual machine, cloudlet or cloudclone, etc.) to ensure service quality and user privacy. On the user side, the computing tasks in the mobile application can be executed directly on the CPU of the user's device, or sent to the MEC server through the data transmission unit (DTU) for remote offloading and execution by the corresponding service instance of the user. The offloading scheduling module makes scheduling decisions for all tasks in the user equipment, and determines the execution mode (local execution or offloading execution) and scheduling order of the task.

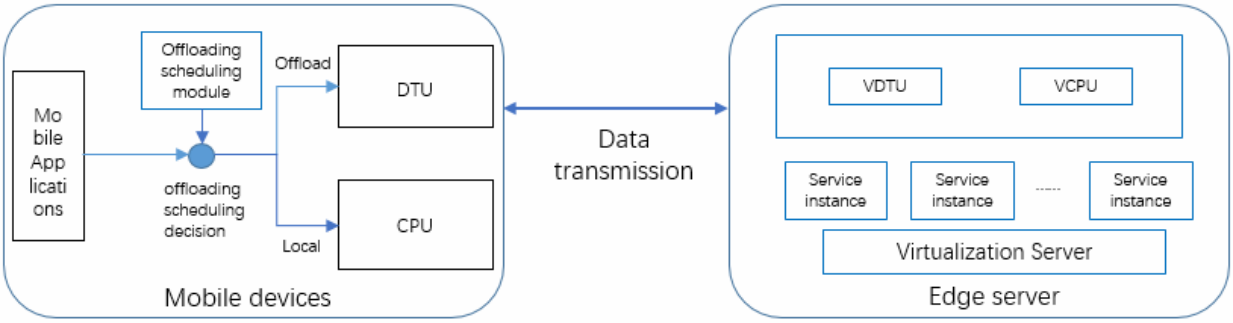

Fig. 1. MEC task offloading architecture

3.2 Network Model

A cell scenario with a base station (BS) and N mobile devices (MD) is considered，shown as Fig. 2. The MEC server and base station are deployed together, and each device can upload data through the wireless channel to realize resource offloading. Whether to offload depends on the overall offloading decision. It is assumed that every member of the set of MD, I = {1, 2, …, N}, has a computing intensive task $A_i \triangleq (X_i, L_i, \tau_i^d)$ needed to be processed. X_i in bit here means the data size needed to calculate A_i. And L_i in CPU cycles needed to calculate each bit of data reflects the calculation intensity. Therefore, $X_i L_i$ can be used to represent the amount of computing resources needed to complete task A_i. In order to avoid the impact of different types of edge devices or servers on the amount of computing tasks, wo assume that the size of L_i has nothing to do with the execution device of task A_i. Since the total time consumption of each task cannot be increased infinitely, τ_i^d is used to indicate the time limit for completing the task, including the time for executing the calculation, the time for offloading data and the possible waiting time. At present, with the development of code decomposition technology, many applications can be divided into two parts for processing, that is, the local execution part and the server execution part. Therefore, we assume that computing task A_i can be arbitrarily divided into two parts for processing at the same time. If it cannot complete the whole task within the allotted time, MDi is allowed to offload part of the data to be processed to MEC server to perform the calculation. We use $a_i \in [0,1)$ to

present the offload rate of MDi, that is, the percentage of offload processing data, and finally use vector $\alpha = [\alpha_1, \alpha_2, ..., \alpha_N]$ to represent the offload rate of all user devices. By splitting the computing tasks, the delay and energy consumption can be reduced.

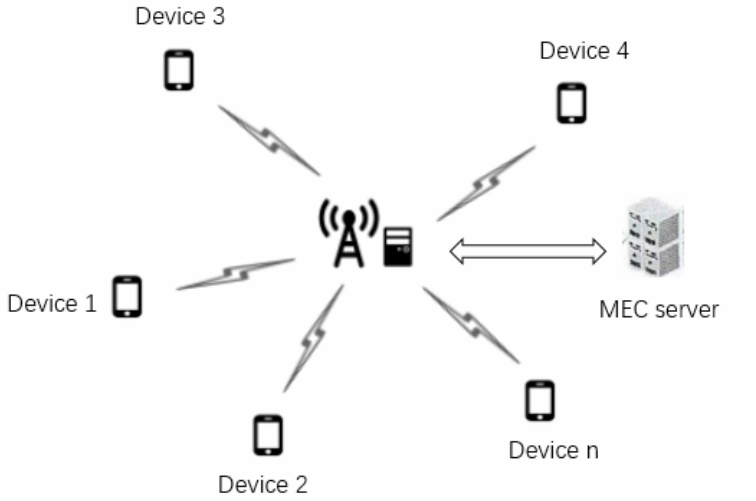

Fig. 2. Network system

W is defined as the bandwidth of wireless channel. In the cell with single base, the inter area interference can be ignored. According to reference [7], the data offload rate R_i can be achieved by MDi is:

$$R_i = W \log(1 + \frac{P_i h_i}{\sigma^2}). \tag{1}$$

Among them, P_i is the transmission power of data offloaded by MDi, and σ^2 is the variance of complex Gaussian white channel noise.

3.4 Calculation Execution Model

In BARaft, consensus process includes four parts: node status broadcasting, leader election, outsider election and log replication.

Local Processing Mode: For the local computing part of MDi, t_i^l is defined as the local execution delay of MDi, that is, the processing time of local device CPU. f_i^l is defined as the CPU frequency of the local device. To simplify the problem, we assume that the maximum CPU frequency of each MD is the same. Local execution delay t_i^l can be expressed as follows:

$$t_i^l = \frac{(1-\alpha_i)L_i X_i}{f_i^l}. \tag{2}$$

At the same time, we define e_i^l as the energy consumption of the local execution part corresponding to MDi, which according to [8] can be expressed as:

$$e_i^l = z_i (f_i^l)^2 (1-\alpha_i) L_i X_i. \tag{3}$$

z_i in this formula is the energy density, which represents the energy consumed by the CPU in each cycle when the local device performs the calculation.

Offloading processing model: calculation of MDi offloading ca be divided into three steps. In the first step, MDi uploads the offloaded part $\alpha_i X_i$ to the base station through the wireless access network, after that the base station forward this part of data to the MEC server. In the second step, the MEC server allocates the corresponding computing resources to perform the computing processing of the offloading

part. According to the above processes, the delay of the first step of the offloading process can be described as:

$$t_{i,t}^{o} = \frac{\alpha_i X_i}{R_i}. \tag{4}$$

The transmission power of MDi is recorded as P_i in this process and the energy consumed in this transmission process is calculated as:

$$e_{i,t}^{o} = P_i t_{i,t}^{o} = \frac{P_i \alpha_i X_i}{R_i}. \tag{5}$$

Define the computing resource allocated to MDi by MEC server in the second step as F_i, then the vector $F_i = [F_1, F_2, ..., F_N]$ represents the resource allocation vector, and the computing time delay $t_{i,p}^{o}$ of device MDi ca be expressed as:

$$t_{i,p}^{o} = \frac{\alpha_i L_i X_i}{X_i}. \tag{6}$$

It should be noted that the total amount of computing resources allocated for each MD cannot exceed the upper limit due to the limited available computing resources $F_{\max}$ of MEC server, that is, $\sum_{i=1}^{N} F_i \le F_{\max}$. In this process, the user device is always waiting to receive. If the power of MDi is defined as P_i^w, the energy consumed by the user device can be described as:

$$e_{i,p}^{o} = \frac{P_i^w \alpha_i L_i X_i}{X_i}. \tag{7}$$

According to reference [9], the return rate of wireless network is very fast, and the size of the return data is much smaller than the uploaded data, so the delay and energy consumption are so small that they can be ignored.

Based on the above formulas, we can calculate the energy consumption of MDi in the whole process of offloading in the following way:

$$e_i = e_{i,t}^{o} + e_{i,p}^{o} = \frac{P_i \alpha_i X_i}{R_i} + \frac{P_i^w \alpha_i L_i X_i}{X_i}. \tag{8}$$

3.3 Problem Modeling

From the previous description, for each user device the energy consumption of MDi in the whole calculation process can be obtained by adding the energy consumption of local processing and offloading processing, which is expressed as follows:

$$e = z_i (f_i^l)^2 (1-\alpha_i) L_i X_i = \frac{P_i \alpha_i X_i}{R_i} + \frac{P_i^w \alpha_i L_i X_i}{X_i}. \tag{9}$$

The purpose of building the mathematical model of this problem is to make clear how to minimize the sum of energy consumption of all users in MEC system under the partial offloading model. Under the condition of limited time and computing resources, this problem can be formulated as follows:

$$\min \sum_{i=1}^{N} e_i.$$

$$\text{s.t. C1: } \alpha_i \in [0,1), \forall i \in N.$$

$$\text{C2: } t_i^l = \frac{(1-\alpha_i)L_i X_i}{f_i^l} \le \tau_i^d, \forall i \in N.$$

$$\text{C3: } t_i^o = \frac{\alpha_i X_i}{R_i} + \frac{\alpha_i L_i X_i}{X_i} \le \tau_i^d, \forall i \in N. \tag{10}$$

$$\text{C4: } 0 \le F_i \le F_{\max}, \forall i \in N.$$

$$\text{C5: } \sum_{i=1}^{N} F_i \le F_{\max}, \forall i \in N.$$

The objective of the optimization problem is to minimize the total energy consumption of all MD, and the required solutions are α and F. The constraint C1 ensures that each MD can be calculated locally or partially offloaded; The constraint C2 indicates that the local processing time delay cannot exceed the time limit; The constraint C3 ensures that the time delay of the offloading process does not exceed the time limit so that the whole execution process is constrained by the time limit; the constraint C4 guarantees that the computing resources allocated to each MD cannot exceed the maximum available capacity of the MEC server; and the constraint C5 guarantees that the comprehensive computing resources allocated to the offloading MD cannot exceed the maximum available capacity of the MEC server.

4 Solution and Optimization Scheme

4.1 Scheme Conception

Among the existing resource offloading algorithms, many algorithms based on numerical optimization have been used, such as steepest descent method, Newton's method, quasi Newton methods and so on. In the meantime, scientists are constantly studying more low-cost and high-performance new algorithms. In real-time MEC system, the process ofoffnloading decision and resource allocation needs to be completed in milliseconds due to the rapid changes of channel conditions, number of users and other system parameters. Particle swarm optimization (PSO) which can basically solve numerical optimization problems is a kind of stochastic optimization technology based on population with the advantages of fast convergence, simple algorithm and easy programming, proposed by Eberhart and Kennedy in 1995. However, PSO and many optimizations based algorithms can get results only after multi-step iteration, which leads to high cost of cloud travel time. To solve this problem, we consider using deep neural network (DNN) to deal with the problem of MEC computing resource offloading.

The steps of deep neural network training are shown in Fig. 3. In the process of training, we first choose an existing optimization based algorithm to get several groups of optimal solutions of the original problem. Here we choose the above-mentioned PSO algorithm as the traditional optimization algorithm. Then we use these parameters and optimization results as the training data of the neural network, and constantly update the network weights until the training is completed. Compared with the traditional optimization algorithm, the trained DNN neural network has higher real-time operation efficiency. When we run a trained DNN, it only needs some simple mathematical calculation or nonlinear transformation, such as Sigmond activation function and Relu activation function, to get the final result. Therefore, if we can better train a DNN to ensure that it is highly similar to the accuracy of the traditional optimization algorithm, we can get a deep learning algorithm which runs much faster than the traditional optimization algorithm, better adapting to the high real-time requirements of MEC system.

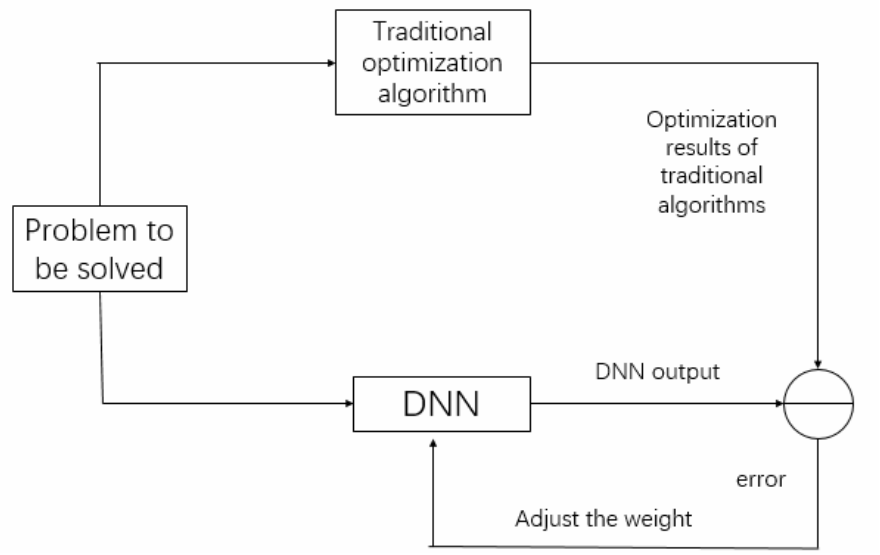

Fig. 3. Training steps

4.2 Particle Swarm Optimization Algorithm

Problem (10) is a nonlinear programming problem with constraints. PSO algorithm can solve small and medium-sized nonlinear optimization problems which has the advantages of good convergence and simple algorithm compared with other optimization algorithms. Therefore, we choose PSO as the traditional optimization algorithm in the training process. The general optimization process of PSO is as follows: Firstly, set the maximum number of iterations, the number of independent variables of the objective function, the maximum speed and position information of particles as the whole search space, initialize the speed and position randomly in the speed range and search space, and initialize a speed randomly for each particle in the particle swarm; Define a fitness function. The individual extremum is the optimal solution found by each example. A global value is found from these optimal solutions, which is recorded as the current global optimal solution. Compared with the historical global optimal solution, it is updated. When the number of iterations or the difference between algebras meets the minimum limit, the algorithm is completed. The specific process is reflected in the following algorithm (Table 1):

Table 1. Particle swarm optimization algorithm

Algorithm 1. Particle swarm optimization algorithm

```
Generate initial population
for k=1 to population size
   particle [k].best=current position
   particle [k].bestfitness=current fitness
end for
gbest=particle.best with lowest fitness
for t=1 to MAX_ITERATION
  for k=1 to population size
      v_k^{t+1} = χ(v_k^t + Σ_{p_i∈N_k} c_i γ_i (p_i^t − x_k^t))
      x_k^{t+1} = x_k^t + v_k^{t+1}
      if current fitness < particle [k].best fitness
        particle [k].best=current position
        particle [k].bestfitness=current fitness
      end if
    end for
    gbest=particle.best with lowest fitness
 end for
 return best
```

The scheduling process of PSO algorithm in resource offloading is as follows (Table 2):

Table 2. The scheduling process of PSO algorithm in resource offloading

Algorithm 2. The scheduling process of PSO algorithm in resource offloading

```
ready={tasks with no precedence}
for each instance i
    idle_i = 0
while ready is not empty
    t =task in ready list with least EST and FET
    assign_t = allocation[t]
    start_t = max(idle_allocation[t], EST)
    finish_t = start_t + exetime_t
    idle_allocation[t] = finish_t
    d ={descendence oft}
    for each d
        if all precedents of d have been executed
           add to ready
        Endfor
 Endwhile.
```

4.3 Deep Neural Network

Fig. 4 shows the structure of the designed fully connected deep neural network, which has an input layer, three hidden layers and an output layer. The input parameter of the neural network is the data size $\{x_i\}$ needed to complete each task, and the output parameter of the neural network is the resource allocation $\{F_i\}$. Therefore, the number of input neurons and output neurons depends on the number of MD. The number of neurons in the three hidden layers is 200, 120 and 80, respectively, in which we use the Relu function $y = \max\{x, 0\}$ as the activation function of the hidden layer.

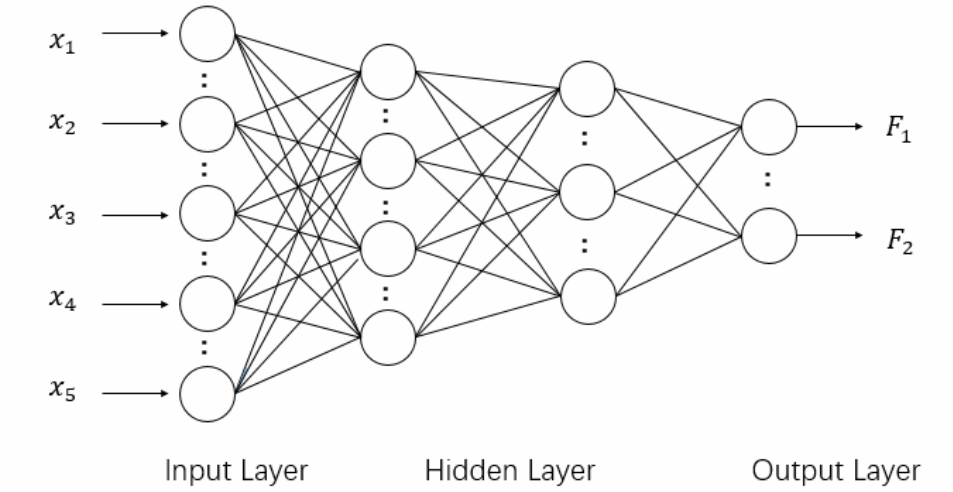

Fig. 4. Deep neural network

In the training process, we use the label $\{F_i^{(k)}\}$ of the training data and the mean square error before the actual output of the neural network as the loss function, and then use the adaptive matrix estimation method (Adam), which can adjust the gradient descent learning rate adaptively and converge more easily, as the optimization algorithm of the network to avoid the process of manually adjusting the learning rate parameters. In the process of testing, we input $\{x_i^{(k)}\}$ of the test data into the trained DNN, and record the running time of each algorithm, so as to observe the efficiency of our trained DNN algorithm.

5 Simulation and Performance Analysis

5.1 Experimental Setup

Suppose that the wireless network scenario is a single cell with bandwidth $W = 20MHz$, and a base station and MEC server are set in the center of the cell. The user equipment MD is randomly distributed in the space of 100 meter away from the base station, and the number of user devices in each time slot is a random number between 3 and 10. The ultimate computing power of MEC server is $F_{\max} = 50GHz/s$. In order to simplify the problem, the difference of CPU between different devices is ignored, and the maximum CPU frequency of each MD is $F_{\max}^{l} = 2GHz/s$. According to reference [10], the transmission power and waiting power of MD are set as $P_i = 100mw$ and $P_i^w = 10mw$ respectively. Assuming that the data size $\{X_i\}$ of each device is evenly distributed between (100, 2000), an experiment is conducted in each time slot, and the average value of the output performance index is taken as the final result. Because the number of user devices in each time slot is different, the number of input and output neurons of the corresponding neural network is also different. For the training data of different number of users, the number of input and output neurons of the neural network should be consistent with the number of users.

5.2 Experimental Result

As shown in Fig. 5, the abscissa represents the number of user equipment, and the ordinate is the total energy consumption of all user equipment (the average value after multiple experiments). We compare the effect of the proposed DNN with that of the PSO, and the results show that the PSO algorithm has a better effect, while the method based on DNN is only 3.8%~7.2% more than that of the PSO algorithm based on the training data source, and the overall effect is consistent, which shows that the neural network can be well similar to the PSO algorithm for optimization process after training.

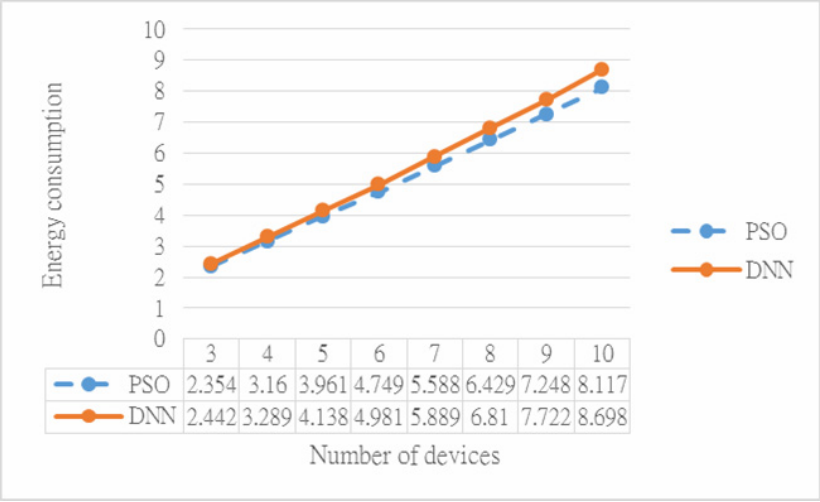

Fig. 5. Energy consumption

As shown in Fig. 6, the resource offloading algorithm based on DNN achieves good performance in terms of computing efficiency of respective N-user systems. For example, compared with the PSO algorithm with n = 3, the total energy consumption index of DNN is increased by 3.74%, but the running time is only 83.7% of the PSO algorithm, which improves the speed by nearly 20%; in the experiment with n = 10, the running time of DNN is 81.1% of the PSO algorithm, and the additional total energy consumption is only increased by 7.2%. The difference between the two methods in the total energy consumption performance is small, which shows that the DNN designed can well approximate the optimization result produced by PSO. With the increase of the number of user devices, the running time of PSO algorithm increases rapidly, and may even exceed the length of a resource allocation slot interval. Therefore, the algorithm is not suitable for real-time MEC system in fact. In contrast, the advantage of running efficiency makes the deep neural network DNN in MEC system more suitable for PSO algorithm.

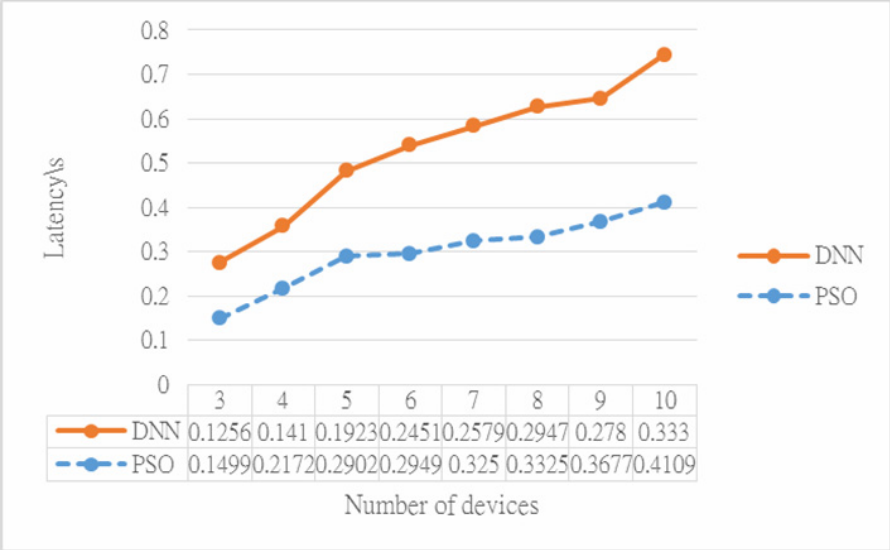

Fig. 6. Latency

6 Conclusion

In this paper, a MEC resource offloading algorithm based on deep learning theory is proposed for real-time allocation of computing tasks under the condition of partial offloading for the single cell multi-user scenario in edge computing. The two kinds of problems are modeled respectively with delay and energy consumption as optimization objectives. Firstly, a system model with total energy consumption constraint is planned, and then the traditional PSO optimization algorithm is used to solve the problem, the results of which are used to train the neural network. In the future, we can continue to study whether we can design more complex MEC application scenarios for the proposed DNN framework, and further improve the network structure to improve the performance of deep neural network. The simulation results show that DNN can approach the PSO algorithm well, and can improve the delay efficiency by nearly 20% when the total energy consumption is less than 10%.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable comments. This research was supported by the National Natural Science Foundation of China under Grant No. U1836101, and Fund projects in the technical field of the basic strengthening plan of the science and Technology Commission of the Military Commission under Grant No. 2019-JCJQ-JJ-031.

Reference

[1] Z. Tong, F. Ye, B. Liu, X. Deng, J. Mei, H. Liu, A task offloading and resource allocation algorithm under multiple constraints in mobile edge computing, Computer engineering and science 42(10)(2020) 1869-1879.

[2] X. Deng, Research on task offloading and resource allocation algorithm based on mobile edge computing, [master dissertation], Hunan Normal University, 2020.

[3] W. Zhan, Computation offloading scheduling and resource management strategy in mobile edge computing, [doctoral dissertation], University of Electronic Science and technology, 2020.

[4] B. Cheng, Research on MEC computing offloading and resource allocation based on deep reinforcement learning, [master dissertation], Beijing University of Posts and telecommunications, 2019.

[5] L. Huang, X. Feng, C. Zhang, L. Qian, Y. Wu, Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing, Digital Communications and Networks 5(1)(2019) 10-17.

[6] G. Yin, Y. Shi, Deep learning task offloading scheme in mobile edge network, Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition) 32(1)(2020) 38-46.

[7] K. Kumar, Y. H. Lu, Cloud computing for mobile users: Can offloading computation save energy?, Computer 43(4)(2010) 51-56.

[8] Z. Xu, Y. Wang, J. Tang, J. Wang, M.C. Gursoy, A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans, in: Proc. 2017 IEEE International Conference on Communications (ICC), 2017.

[9] Q. Shi, M. Razaviyayn, Z.Q. Luo, C. He, An iteratively weighted mmse approach to distributed sum-utility maximization for a mimo interfering broadcast channel, in: Proc. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011.

[10] Y. Cao, T. Jiang, C. Wang, Optimal radio resource allocation for mobile task offloading in cellular networks, IEEE Network 28(5)(2014) 68-73.