An Adaptive Unloading Algorithm of Computing Tasks Based on Edge Cloud Collaboration Scenario for Internet of Things

Feilong Zhang, Yueyang Xiong(✉), and Yonghua Li

Beijing University of Posts and Telecommunications, Beijing 100876, China
{xyueyang,liyonghua}@bupt.edu.cn

Abstract. In recent years, the analysis of big data in the realm of the Internet of Things (IoT) has garnered increasing attention. Several cloud platforms offer pre-trained machine learning models to comprehend IoT data. Nonetheless, to utilize these cloud services, the transmission of personal data is necessitated, and network issues may impede clients from obtaining timely analytical results. To address these challenges, the migration of data and analytical tasks to edge platforms is gaining momentum. However, the majority of edge devices lack the capacity required to process and train on vast datasets. In response to the substantial computational demands in IoT scenarios, we propose an adaptive task offloading algorithm within an edge-cloud collaborative system. Leveraging Deep Belief Networks (DBN) technology, data is hierarchically processed, and the performance of both computing devices and the edge-cloud collaborative system's network is meticulously assessed. Our proposed adaptive task offloading algorithm is underpinned by a meticulously designed processing time model. Experimental results unequivocally demonstrate the algorithm's pronounced advantages in enhancing the overall response time within the context of edge-cloud collaboration.

Keywords: Edge cloud collaboration · DBN · Adaptive unloading of computing tasks

1 Introduction

In the interconnected world of IoT, the proliferation of diverse terminal sensing devices has led to a surge in complex data. Real-time processing has become imperative as data volumes surpass Zeta Bytes (ZB) [1]. To address this, researchers advocate integrating edge computing with cloud computing to alleviate computational loads [2]. This entails offloading cloud-assigned tasks to edge devices, aiming to enhance data transmission and real-time processing while reducing the cloud's computational burden [3].

Supported by National Key Research and Development Program of China 2022YFC2806300.

F. Gao et al. (Eds.): Chinacom 2023, LNICST 590, pp. 508–527, 2024.
https://doi.org/10.1007/978-3-031-67162-3_32

Despite potential benefits, edge-cloud collaboration has limitations, particularly in resource-constrained scenarios. The absence of a closed-loop optimization mechanism among terminals, edge servers, and the cloud can hinder the full potential of edge-cloud collaboration [4]. Researchers have explored energy-efficient solutions, leveraging edge devices to migrate computing tasks from the cloud [5–9].

Efficient computing task offloading algorithms are pivotal to allocate tasks between edge devices and cloud centers, optimizing energy consumption. This paper introduces a delay-optimized task unloading algorithm based on task priority classification [10]. Simulation results demonstrate its effectiveness in enhancing system revenue and reducing task delays.

Additionally, a joint optimization scheme addressing task unloading, computing, and communication resources under multi-user wireless interference is proposed [11]. This scheme enhances system utilization, reduces task execution delays, and lowers energy consumption. In industrial IoT, a multi-user and multi-access edge computing (MEC) delay optimization problem is tackled [12]. A computational unloading strategy based on particle swarm optimization yields promising results in reducing the overall network cost.

To address these challenges, this paper introduces a novel approach that combines deep learning using Deep Belief Networks (DBN) with edge-cloud collaboration. Hierarchical data processing using DBN and an adaptive task offloading algorithm are key components. Simulations evaluate DBN's advantages in reducing intermediate data and analyze task offloading proportions under varying conditions of data volume, network bandwidth, edge device load, and network delay.

The main contributions of this paper are as follows:

- Application of the DBN Algorithm: To facilitate more convenient and flexible data processing, we introduce the Deep Belief Networks (DBN) algorithm. DBN is employed for the hierarchical processing of input data slated for computation, streamlining subsequent data unloading tasks with greater flexibility.
- Development of an Adaptive Task Offloading Algorithm: This paper presents the creation of an adaptive task offloading algorithm tailored for computing tasks. This innovative algorithm incorporates Performance of CPU and Memory (PCM) metrics to quantitatively assess equipment performance. Moreover, it conducts an in-depth analysis and optimization of network performance by integrating vital parameters like network bandwidth and delay. These insights enable the quantitative modeling of computing task processing times. We further leverage the breadth-first search (BFS) algorithm to minimize the processing time of computing tasks, thereby realizing an adaptive task offloading algorithm that enhances overall system performance.
- Experimental Validation: To validate the practical advantages of our proposed algorithm, we design and conduct experiments. These experiments simulate the application of the computing task adaptive offloading algorithm and provide a comprehensive analysis of its performance in comparison to traditional cloud computing center or edge device computing models.

The rest of this paper is organized as follows: the second section reviews the collaborative model, mainly introduces the local computing model and cloud computing model. The third section introduces the details of computing task adaptive unloading algorithm. The fourth section analyzes and verifies the design simulation experiment of computing task adaptive unloading algorithm. The fifth section gives the simulation conclusion and analyzes and verifies it.

2 Related Work

Edge cloud collaboration technology, a pivotal IoT component, has gained extensive attention in academia and industry. In [13], a local outlier factor algorithm based on affinity propagation clustering is proposed to verify the network connectivity of edge cloud cooperation. A closed optimal task segmentation strategy based on standardized backhaul communication and cloud computing capability is derived in [14], but these works only focus on the control of service delay and ignore other important performance indicators, such as energy consumption.

Recent work is devoted to joint communication and computing resource management in order to balance service delay and energy consumption. An algorithm based on alternating convex search is proposed in [15], which minimizes the weighted sum of delay consumption and energy consumption by jointly optimizing local computing frequency, task segmentation and transmission power, while ensuring strict delay requirements and residual energy constraints. A new decentralized algorithm to jointly allocate communication and computing resources to meet the requirements of service delay and energy consumption is designed in [16]. However, the research scenarios of these two papers are in quasi-static networks. In this scenario, the computing task set and channel state are assumed to remain unchanged without considering the potential time dynamics.

To apply to dynamic industrial scenarios, some work focuses on edge cloud cooperation, as well as the dynamic changes of wireless channel status and data arrival. For example, in [17], the dynamic channel state is described as a Markov decision process, and deep learning is applied to solve the problem of optimal communication and computing resource allocation. In [18,19], an online algorithm is proposed, which can adjust the transmission power and CPU calculation speed of the edge server to solve the problems of dynamic time-varying channel state and data arrival. However, all these works adopt a simple task/data processing model, ignoring the impact of its detailed performance (such as processing accuracy) on the IoT system.

Some studies have begun to analyze the impact of the industrial environment on data accuracy. In [20], the influence of industrial noise on data quality is studied, and an improved wavelet denoising algorithm is proposed. However, it is not considered that the denoising algorithm will occupy the corresponding computing resources, which may affect the computing delay and energy consumption. In [21], the influence of physical layer sampling rate on data accuracy is considered, and the influence of sampling rate on the performance of IoT

system is analyzed. However, the necessity of data preprocessing and the impact of different preprocessing methods on system performance are not considered.

In short, what distinguishes this paper from all existing research work is that it introduces DBN into the research of edge cloud collaborative computing in the IoT scenario, and puts forward a computing task processing time model to segment computing tasks.

- The DBN is introduced into the edge cloud collaboration system. Hierarchical processing of IoT data facilitates the subsequent design of a computing task segmentation algorithm.
- Design and calculate the task processing time model. Firstly, PCM algorithm is used to evaluate the performance of computing equipment, and the load capacity and processing efficiency of the equipment are analyzed; Secondly, the network performance of the edge cloud system is analyzed. The network performance is the key to the data transmission between the edge cloud servers and plays a vital role in the overall response time of the system; Finally, the processing time model of computing task is constructed according to the processing capacity of equipment and network performance.
- Based on the computing task processing model, an adaptive unloading algorithm of computing task is proposed based on BFS algorithm, which is committed to obtaining the shortest response time.

3 Model and Related Algorithm

The substantial volume of data imposes significant computational load on the cloud computing center. To address these challenges, this paper developed an adaptive task unloading algorithm designed for managing the extensive computing tasks. This algorithm aims to facilitate collaborative processing between edge devices and the cloud computing center.

3.1 System Model

The system model, as illustrated in Fig. 1, typically comprises four layers in an IoT setup. At the base, there's the device layer, primarily consisting of edge devices and sensors responsible for data collection required for applications. These devices connect to the gateway for data transmission over the network. Once uploaded to the network, the data undergoes processing and forwarding through common middleware devices, eventually reaching the corresponding application where it is retrieved and processed. The computing task adaptive unloading algorithm is designed within the framework of this system model.

3.2 Hierarchical Data Processing Based on DBN

The Deep Belief Network (DBN), originally proposed by Hinton et al., was developed to address the challenge of rapid and automated feature learning [22].

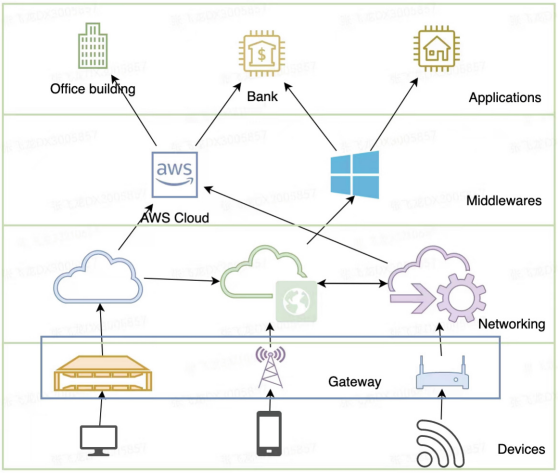


Fig. 1. System model

Hinton and his colleagues introduced novel insights into understanding and analyzing the DBN model, which significantly expanded its applicability across various domains.

Figure 2 illustrates the DBN's structure, composed of multiple layers stacked with Restricted Boltzmann Machines (RBMs). RBMs are trained to learn diverse parameters within the deep network using the Contrastive Divergence algorithm (CD). This section will provide a detailed overview of RBM's structure and the Contrastive Divergence algorithm.

- RBM

RBM is one of the basic modules of DBN. The two parts of visual layer (V) and hidden layer (H) together constitute the RBM, and the neurons between them are connected by weight. The process of training each DBN in DBN consists of two steps: mapping and reconstruction. The first step is the mapping process. The data needs to be transmitted to the visual layer V1 of RBM1. The information contained in the visual layer V1 is sampled by Gibbs and the sampled information is transmitted to the hidden layer H1. Secondly, it enters the reconstruction link. In the reconstruction link, Gibbs samples the information contained in the hidden layer H1 again, and then transmits the sampled data back to the visual layerV1. The function of mapping and reconstruction is called recursively, and the training of RBM is considered to be completed until the calculated error meets the specified value. In the training of the second RBM, the information contained in the hidden layer V1 of the front RBM is taken as the input of the visual layer V2 of the rear RBM, and the mapping and reconstruction process are repeatedly executed, and so on, until all the training processes of the RBM are completed.

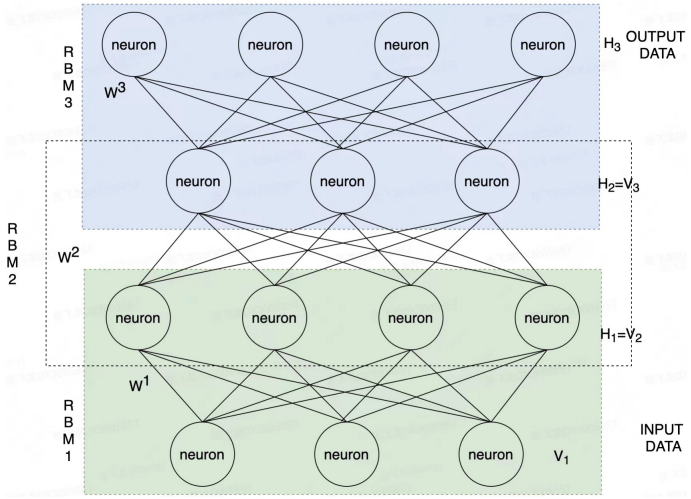


Fig. 2. Structure diagram of DBN

For the state (V, H) given by each group of RBM, the energy contained can be formulated as

$$E(v,h) = -\sum_{i=1}^{I}\sum_{j=1}^{J} v_i w_{ij} h_j - \sum_{i=1}^{I} b_i v_i - \sum_{j=1}^{J} a_j h_j \tag{1}$$

where V_i means the state of the ith visual unit, H_i is the state of the ith hidden unit, w_ij stands for visual unit v_i and hidden unit H_j corresponding symmetrical connection weight.

The calculation method of hidden layer unit state can be formulated as

$$P\left(H_j = 1 \mid v\right) = \sigma\left(\sum_{i=1}^{m} w_{ij} + a_i\right) \tag{2}$$

where $P(h_j = 1|v)$ represents the activation probability of the jth unit of the hidden layer, and $P(v_i = 1|h)$ represents the activation probability of the ith unit of the visible layer, σ activate the function for sigmoid.

The state of each visible layer unit is reconstructed as

$$P\left(V_i = 1 \mid h\right) = \sigma\left(\sum_{j=1}^{n} w_{ij} h_j + b_j\right) \tag{3}$$

when the RBM is trained, it is brought into the whole DBN, and then the gradient descent algorithm is used to reconstruct the whole network.

The input layer is used to input data. The nonlinear mapping process of data from the input layer x to the hidden layer Z is called mapping, and the hidden

layer Z is remapped to the visual layer $\hat{x}$ The nonlinear mapping of is called reconstruction. The mapping and reconstruction process can be formulated as

$$z = f\left(w_z x + b_z\right) \tag{4}$$

$$\hat{x} = f\left(w_{\hat{x}} Z + b_{\hat{x}}\right) \tag{5}$$

where the weights of input layer and hidden layer pass through W_z for $W_{\hat{x}}$. It means the weight from hidden layer to reconstruction layer B_z ,$b_{\hat{x}}$ offset of hidden layer and output layer respectively. In addition, the $f(\cdot)$ function represents the mapping relationship, while the relationship of W_z and $W_{\hat{x}}$ is as follows: $w_z = w_{\hat{x}}^T = w$, where T represents transpose.

The gradient descent method is used in the whole network parameter update link of the DBN [23]. The specific method is as follows: suppose the input data $\hat{x}$. The reconstruction error function between and the reconstructed data x is $e(x, \hat{x})$. Then the optimal network parameters w ,B_z ,$b_{\hat{x}}$ can be calculated by solving the error function. The error function formula is formulated as

$$T = \text{argmin}[e(x, \hat{x})] \tag{6}$$

According to the gradient descent algorithm, the parameters can be formulated as

$$w = w - \eta \frac{\partial e(x, \hat{x})}{\partial w} \tag{7}$$

$$b_z = b_x - \eta \frac{\partial e(x, \hat{x})}{\partial b_z} \tag{8}$$

$$b_{\hat{x}} = b_{\hat{x}} - \eta \frac{\partial e(x, \hat{x})}{\partial b_{\hat{x}}} \tag{9}$$

where η represents the learning rate, and the parameters will be updated continuously according to the related rules. When the reconstruction error meets the convergence standard, the RBM is constituted by the parameters at this time. So far, a training process of the RBM has been completed. In addition, the value of hidden layer can be used not only as the extracted feature, but also as the input value of the next layer RBM.

- Contrast Divergence Algorithm

Contrast divergence algorithm is one of the fast learning algorithms of RBM. There are obvious differences between the contrast divergence algorithm and the common Gibbs sampling algorithm. The algorithm inputs the training sample data as input parameters into the visual layer of the first RBM, the probability that the unit value of the first RBM is 1, which can be calculated by Eq. (10). Secondly, the probability that the *ith* visual unit is 1, which can be calculated by Eq. (11), and then a process of the visual layer is reconstructed. When using contrast divergence algorithm to train data, the update rules of each parameter are as follows:

Firstly, for each RBM, the contrast divergence algorithm is used to update its parameters. The utilization conditions of each visual node are randomly selected

from 0,1. So far, $v^{(0)}$, $v^{(0)} \leftarrow v$ is obtained, and the initialization parameter increment is $\Delta w_{ij} = \Delta a_i = \Delta b_i = 0$, m samples are taken between the visible layer and the hidden layer of each RBM, and is used to represent Gibbs sampling, M represents the sampling times, and the value range of $m = 0, 1, \ldots, M-1$, which is formulated as

$$h_j^{(m)} \sim P\left(h_j = 1 \mid v^{(m)}\right) \tag{10}$$

$$v_i^{(m+1)} \sim P\left(v_i =\mid h^{(m)}\right) \tag{11}$$

Secondly, update the weight of the RBM according to Eq. (12), where $m = 0, 1, \ldots, M-1, \Delta w'_{ij}$ represents the weight parameter of the $k-th$ layer of the hidden layer of the updated RBM. In addition, the probability that the hidden layer output is 1 after the visual layer of the RBM is initialized, which can be expressed by the formula $P\left(h_i = 1 \mid v_i^{(0)}\right) v_i^{(0)}$, where $P\left(h_i = 1 \mid v_i^{(m)}\right) v_i^{(m)}$ represents the probability that the hidden layer output is 1 after M sampling of the visual layer of the RBM.

$$\Delta w'_{ij} = \Delta w_{ij} + \left[P\left(h_j = 1 \mid v_i^{(0)}\right) v_i^{(0)} - P\left(h_j = 1 \mid v_i^{(m)}\right) v_i^{(m)}\right] \tag{12}$$

After that, the offset parameters of the visual layer for the RBM are formulated as

$$\Delta a'_i = \Delta a_i + \left[v_i^{(0)} - v_i^{(m)}\right] \tag{13}$$

where $m = 0, 1, \ldots, M-1, \Delta a_i$ represents the offset parameter of the visual layer of the updated RBM. In addition, $P\left(h_j = 1 \mid v_i^{(0)}\right)$ indicates the output after the initialization of the visual layer, $P\left(h_j = 1 \mid v_i^{(m)}\right)$ represents the output of the visual layer after M samples.

• SOFTMAX Classifier

The logistic regression model is only for the problem of two classification is pointed out in [24], while the SOFTMAX classifier is more universal for multiple classification application scenarios. The structure diagram of SOFTMAX classifier is shown in Fig. 3, where $s(\cdot)$ is the activation function. Which is to ensure that the value range of input vector is within (0,1) and the sum of output data is 1, because the activation function in the SOFTMAX distributor is similar to the sigmoid in the neural network, the SOFTMAX classifier is generally used at the output of the deep learning network and connected with the hidden layer of the lowest layer of the deep learning network. Such a complete deep learning network structure is formed, which can further optimize the processing results of the deep learning network.

Suppose $Q = [q_1, q_2, \ldots, q_i]$ is the input vector of softamx classifier, $P = [p_1, p_2, p_3, \ldots, p_N]$ is the output vector of classifier, and N represents the dimension of the last hidden layer. In addition, W and B represent the weight and bias of the classifier, which is formulated as

$$q_i = p(Y = i \mid P; w, b) = s(wP + b) = \frac{e^{w_i P + b_i}}{\sum_{k=1}^{3} e^{w_k P + b_k}} \tag{14}$$

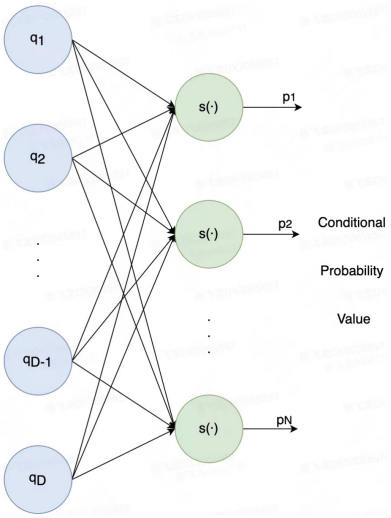


Fig. 3. Structure diagram of SOFTMAX classifier

3.3 Adaptive Algorithm of Data Computing Task Unloading

Efficient data collaboration is at the core of edge-cloud cooperation, with a pivotal focus on rationalizing data transmission and computing task allocation between edge devices and cloud computing centers. Prior research in the realm of edge-cloud collaborative data and computing task offloading has been extensive. For instance, a collaborative model for intelligent machine tools in the context of "Industry 4.0" is presented in [25]. This model introduces a method for regularly offloading data with low real-time demands to the cloud computing center, effectively alleviating equipment load pressures and enhancing the intelligence of edge devices.

In this section, building upon the hierarchical data processing using DBN as discussed in Section A, we will design an adaptive algorithm for unloading data computing tasks to enable real-time and flexible data processing. However, the intricacies of the network environment and the diverse performance requirements for data processing necessitate careful decision-making in data computing task offloading. Inadequate design decisions can lead to network load imbalances, potentially resulting in system failures under peak or specialized conditions. To address these concerns, our algorithm takes into account a comprehensive range of factors, including computing task loads, equipment performance, network bandwidth throughput, latency, and network stability. It models the data processing time per unit size and constructs a Weighted Directed Graph (WDG) based on data volume and processing time. Once this WDG structure is established, we employ the Breadth-First Search (BFS) algorithm to partition computing tasks, minimizing the overall system response time.

Computing Device Performance Based on PCM Algorithm. Firstly, according to the PCM (performance CPU memory) algorithm, the performance of edge devices and cloud computing center are dynamically obtained. Suppose the device group is $D = \{D_0, D_1, \ldots, D_{n-1}\}$, where D_i represents the ith device node. Use $P(D_i)$ to represent the current performance status of the equipment, W_i is used to represent the specific gravity coefficient corresponding to the performance index of each equipment, and $\sum_{i=0}^{1} W_i = 1$. In addition, use F_c represents the frequency of the device CPU, T_m represents the total memory usage. Therefore, the current equipment performance is formulated as

$$P(D_i) = W_0 * F_c(D_i) + W_1 * T_m(D_i)) \tag{15}$$

Suppose $L(D_i)$ represents the equipment load condition, K_i represents the specific gravity coefficient corresponding to the load capacity index of each equipment, and $\sum_{i=0}^{1} K_i = 1$. Simultaneous use of U_c represents the current CPU utilization, U_m represents the current memory usage. Then $L(D_i)$ can be is formulated as

$$L(D_i) = K_0 * U_c(D_i) + K_1 * U_m(D_i) \tag{16}$$

For the formula used to calculate the performance of the above equipment, it is necessary to quantify the utilization of CPU. The parameters file is used to obtain the device information. The main information includes the user's occupation time, the time when the system kernel is used and the time when idle processes are used. Using the time when idle processes are used, it is convenient to calculate the ratio of unused CPU of the device, so as to obtain the CPU utilization of the device. After obtaining the CPU utilization of the device, the CPU utilization can be calculated.

Arbitrary selection t_1, t_2 two moments, by F_i stands for t_i Unoccupied process at time i, C_i stands for t_i the state of the CPU at the time. The CPU unutilization rate can be calculated from the proportion of unused processes

$$F = \frac{F_2 - F_1}{C_2 - C_1} * 100\% \tag{17}$$

CPU utilization U_c can be obtained through $U_c = (1 - F) * 100\%$.

Memory usage information can be read from memory parameters, and memory utilization can be calculated based on Memtotal and Memfree. The memory usage is formulated as

$$U_m = \frac{\text{Memtotal-Memfree}}{\text{Memtotal}} * 100\% \tag{18}$$

The equipment weight is calculated by using the equipment performance and equipment load capacity. The calculation is formulated as

$$Q(D_i) = L(D_i)/P(D_i) \tag{19}$$

where the equipment weight Q represents the load capacity of the current equipment. The greater the weight Q, the greater the load borne by the current equipment, the less corresponding calculation tasks assigned to the equipment.

At the same time, in order to improve the robustness of the system load capacity, the load redundancy value will be introduced to a certain extent, and this index will be used to judge the capacity of a device node when adding load at a certain time. Assuming that the equipment operation cycle can be expressed in T, the load redundancy value is formulated as

$$M\left(D_i\right) = T * P\left(D_i\right) / L\left(D_i\right) \tag{20}$$

Network Performance Analysis of Edge Cloud System. In the investigation of collaborative computing within the edge-cloud scenario, the allocation of computing tasks necessitates a comprehensive consideration of factors like network bandwidth and delay [26]. Following the analysis and assessment of equipment performance using the PCM algorithm as discussed earlier, the next crucial aspect in edge-cloud collaboration is the evaluation of network performance. The comprehensive design concept is illustrated in Fig. 4.

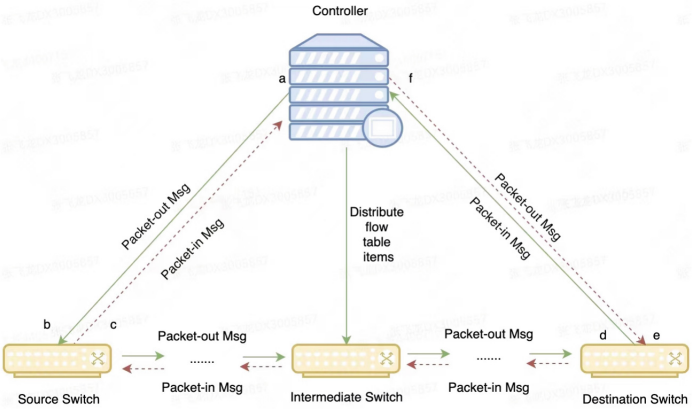


Fig. 4. Overall design drawing of time delay measurement

The data flow table item is initially transmitted from the source switch to the destination switch and vice versa by the controller. Upon the installation of the intermediate switch, the flow table item triggers a packet-out message sent to the source switch. This message's data segment includes a timestamp, indicating when the controller sent the request message. The action within the packet-out message instructs the source switch to forward the data packet to the destination switch. If the destination switch receives a data packet from the preceding switch but cannot find a proper match in its flow table item, it sends a packet-in message to the controller and returns the data packet. Upon receiving the data packet from the switch, the controller calculates the time difference between the data packet's reception time and the timestamp in the message, resulting in the overall time difference denoted as T_1. This encompasses the delays from the controller

to the source switch, from the source switch to the destination switch, and from the destination switch back to the controller. Following this, the reply process ensues, and the time difference for the entire reply process is denoted as T_2. If we define the round-trip delays in the transmission process from the source switch to the destination switch as T_s and T_d, respectively, the overall round-trip delay from the source switch to the destination switch can be expressed as

$$\text{Delay} = T_1 + T_2 - T_s - T_d \tag{21}$$

However, when the number of intermediate switches involved in the entire test link exceeds a certain threshold, the measurement time becomes excessively long. This extended duration poses challenges for the controller in accurately tracking the forwarding process and consumption time of the intermediate routers. In such cases, the controller must rely on the data packet's return to determine the timing. If a data packet is directly forwarded to the controller without traversing the destination switch, it can introduce significant errors in the experimental measurement results. Given that the controller has the capability to transmit multiple data packets simultaneously, a preventive measure involves dividing the n switches in the link into n sub-links. By separately measuring the delay of each individual sub-link, the overall delay can be computed as the sum of each delay component

$$\text{Delay} = \sum_{i=1}^{n} \text{SubDelay}(i) \tag{22}$$

Secondly, network stability will be assessed through an active measurement method to calculate the network packet loss rate. This involves counting the number of packets sent, denoted as S, by the source node and the number of packets received, denoted as R, by the destination node within the link. The packet loss rate can be computed using the following formula

$$\text{LossRate} = \frac{S - R}{S} * 100\% \tag{23}$$

Finally, following the network delay and packet loss rate measurements, network bandwidth, another crucial network parameter, will be evaluated. An active measurement method will be employed, utilizing the variation in one-way network delay as a proxy for bandwidth estimation. The primary measurement process is outlined in Algorithm 1.

Firstly, the simulator initiates the transmission of measurement data packets into the network at a predefined rate, maintaining this rate while invoking the delay measurement function to continuously assess real-time network delays. Based on the measured network delay data, the transmission rate is dynamically adjusted in response to network conditions.

Secondly, by monitoring changes in one-way delay, the system can deduce network congestion if the delay consistently increases, indicating that the current transmission rate exceeds the available network bandwidth. Conversely, if the one-way delay remains stable, it suggests that the link bandwidth surpasses the

Algorithm 1: Network Bandwidth Test.

Input: Packet rate: r.
Output: Bandwidth: BW.

Initialize r, $delay$ maximum;
while $delay$ enlarge **do**
  Reduce packet rate r;
  calculation $delay$;
Record current packet rate r;
Get r multiple times and take the average value;
return BW.

current transmission rate, warranting a rate increase. Consequently, the sender can adapt the rate of sending measurement packets based on one-way delay fluctuations until the delay stabilizes.

Finally, the link's bandwidth is assessed by computing the average transmission rate of measurement packets over recent periods. This process enables efficient adaptation to evolving network conditions.

Computing Task Processing Time Model. Building upon the PCM evaluation detailed in Sect. 1) and the network performance analysis in Sect. 2), this section proceeds to construct models for the processing speed of unit data volume for both edge-side equipment and the cloud computing center. These models aim to quantify the processing time required for a unit of data volume when processed by these respective components.

Drawing from the insights gained in Sects. 1) and 2), the processing time for edge-side equipment, denoted as T_e, is calculated as follows for a unit data volume

$$T_e = \frac{1}{\alpha P + \eta_1 L + \eta_2 M} \tag{24}$$

where P represents the performance of the edge side equipment, L represents the load status of the equipment, and M represents the load redundancy value, that is, the remaining processing capacity of the current equipment. α, η_1, η_2 is the coefficient of the three.

Edge side processing and transmission time T_{e+s} is formulated as

$$T_{e+s} = T_e + \frac{1 + D + \beta L}{BW} \tag{25}$$

For the junction between the edge side and the cloud computing center, the processing time per unit data volume needs to be modeled by integrating the factors of equipment processing speed and network performance. The model T_c can be formulated as

$$T_c = \frac{n}{\sum_{i=1}^{n} (\alpha P_i + \eta_1 L_i + \eta_2 M_i)} \tag{26}$$

Therefore, for the unit data volume, the overall response time of the edge cloud collaboration system can be formulated as

$$T = \delta T_e + \gamma T_{e+s} + \theta T_c \tag{27}$$

This section has modeled the overall response time of the edge cloud collaboration system. The next section will utilize processing times and the deep confidence hierarchical network to construct a weighted directed graph for data processing within the edge cloud collaboration system. Computing tasks will be segmented using the BFS algorithm to minimize the overall system response time.

Adaptive Algorithm Based on BFS. Building upon prior research, this section will employ the computing task processing time model to quantify data and construct the WDG diagram, illustrated in Fig. 5.

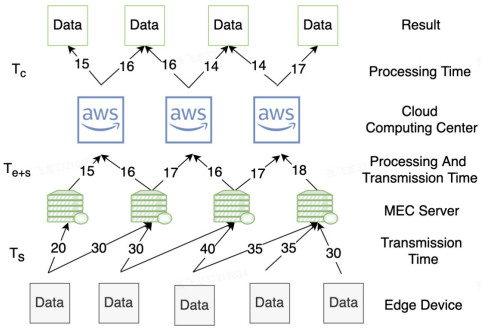


Fig. 5. Overall design drawing of time delay measurement

Upon constructing the WDG diagram, the BFS algorithm will be applied to acquire the optimal solution for the system's overall response time. The detailed algorithm flow is as follows.

Following traversal of the WDG structure diagram using the BFS algorithm, the computing task unloading scheme that minimizes the overall system response time is obtained. An illustrative example of the cut model is presented in Fig. 6.

The entire process of the adaptive algorithm for data computing tasks is depicted in Algorithm 2. This algorithm is invoked for each input data, facilitating data evaluation and the unloading of computing tasks. Subsequently, these computing tasks are dispatched to various edge-side and cloud computing center equipment for task computation.

4 Simulation Results

This section begins with a description of the experimental setup, followed by the execution of simulation experiments. The simulation experiment consists of two

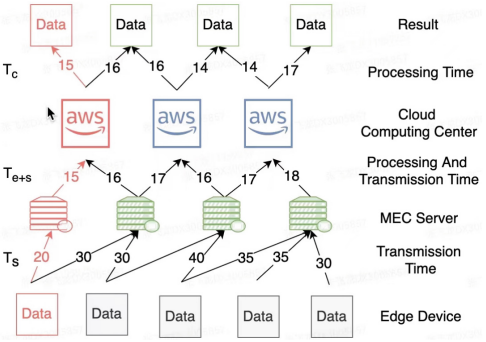


Fig. 6. Example of calculation task of each node after traversal

Algorithm 2: Data Computing Task adaptive algorithm.

Input: Data: d.
Output: Proportion: p.

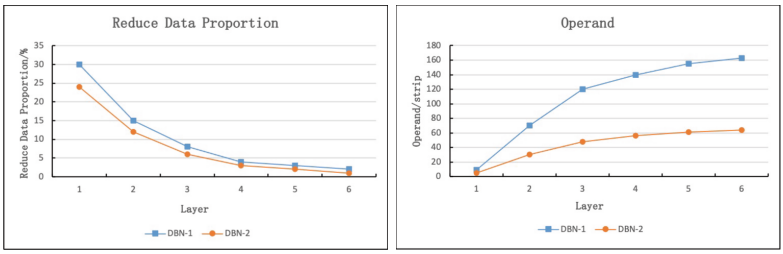
Initialize d maximum;
while d **do**
  Obtained $P(D_i)$,$L(D_i)$,$M(D_i)$ according to PCM algorithm;
  calculation $delay$,BW,$LossRate$;
  Modeling edge processing time: T_e;
  Modeling edge processing time plus network transmission time: T_{e+s};
  Modeling cloud computing center processing time: T_c;
  Build WDG diagram of overall response time of computing task;
  BFS algorithm breadth first traversal to obtain the minimum response time;
return p.

primary components. In the first part, two types of DBN are defined using the TensorFlow framework to analyze the data volume and computational savings achieved by DBN with varying layers. The second part simulates the computing task unloading ratio under different experimental scenarios.

4.1 Performance Evaluation of DBN

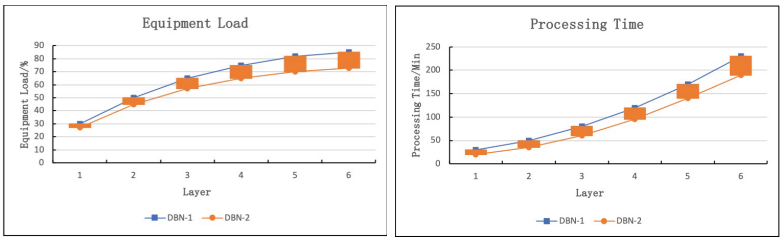
This section focuses on evaluating DBN performance. Two sets of DBNs are constructed using the TensorFlow framework and executed on hardware configurations featuring an Intel Core i7 9700 CPU and an NVIDIA GeForce GTX 2080 graphics card. During the experiment, two DBN computation tasks are conducted, involving the calculation of operands and intermediate data generated at each layer.

The variations in data reduction ratio and computational overhead across different layers for the two DBNs is illustrated in Fig. 7. Notably, these DBNs differ in their weight matrix W^2. As observed in the figure, as the number of DBN layers increases, the data reduction ratio experiences a sharp decline, accompanied



(a) Reduce data proportion (b) Operand

Fig. 7. Analysis of reducing data proportion and operation quantity of DBN



(a) Reduce data proportion (b) Operand

Fig. 8. Analysis of reducing data proportion and operation quantity of DBN

by a rapid rise in the number of operands. This substantial increase in computational operations results in the generation of a significant volume of intermediate data, consequently leading to a decrease in the data reduction ratio.

The impact of DBN on equipment load status and processing time is illustrated in Fig. 8. It is evident that as the number of DBN layers increases, the corresponding equipment load capacity also experiences a significant rise. When the number of layers reaches 4, the equipment load capacity reaches approximately 70%. Beyond this point, increasing the number of layers results in equipment load capacity becoming a critical limiting factor for system processing capability. Furthermore, the processing time diagram in Fig. 7 reveals that as the number of layers increases, equipment processing time accelerates. When the number of layers surpasses 4, the curve's slope becomes steeper, indicating a more pronounced increase in equipment processing time.

Taking into account the combined effects of reduced data proportion, intermediate operations, and the number of DBN layers on equipment load and processing time, we have opted to utilize DBN-1 for our subsequent experiments.

4.2 Performance Analysis of Adaptive Algorithm Based on BFS

In this section, we will analyze the adaptive computing task unloading algorithm's performance in various environments. The impact of different factors on the task unloading proportion is illustrated in Fig. 9.

When examining the influence of data quantity while keeping other factors constant, illustrated in Fig. 9 (a), it's evident that as the data volume grows, more computing tasks are assigned to edge devices for processing. However, there's a point where the allocation to edge devices plateaus due to their performance limitations.

The effect of network bandwidth when other factors remain fixed is shown in Fig. 9 (b). As network bandwidth increases, the proportion of computing tasks allocated to edge devices gradually decreases. The most significant drop occurs when bandwidth reaches 5000MB, after which the decline slows.

When analyzing the impact of edge device load proportion while keeping other factors constant, illustrated in Fig. 9 (c), we observe that with low edge device loads, roughly 30% of computing tasks are allocated to the edge. This allocation decreases as the edge device load increases. When edge devices reach a 60% load proportion, they are rarely assigned computing tasks.

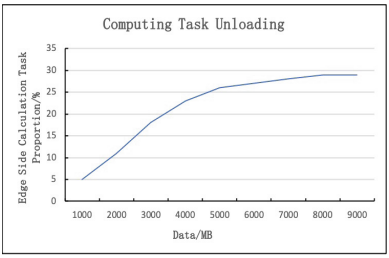
The impact of network delay when other factors are fixed is displayed in Fig. 9 (d). In scenarios with low network delays, indicating good network conditions, more computing tasks are directed to the cloud computing center. However, as network delay increases, an increasing number of tasks are offloaded to the edge. Yet, this allocation saturates when the load capacity of edge equipment is reached.

The overall response time of the system only relying on the Cloud Computing Center for computing and the edge cloud collaborative system for computing under different circumstances are showns in Fig. 10.

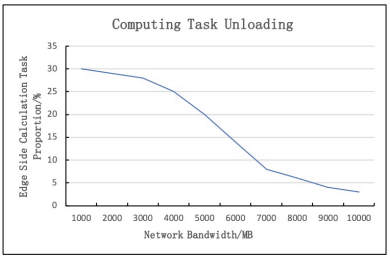
The impact of increasing data volume while keeping other factors constant is shown in Fig. 10 (a). Both processing methods exhibit an increase in response time with growing data volume. However, the difference in response time between the two methods gradually widens, highlighting the edge cloud collaboration system's superior response speed in unconstrained conditions.

As shown in Fig. 10 (b), we observe the effect of increasing network bandwidth while other factors remain unchanged. As network bandwidth increases, the overall response time decreases for both processing methods. Notably, when network bandwidth is limited, the system's response time using the Cloud Computing Center alone is longer. However, when network bandwidth reaches 7000MB, it no longer limits system performance. At this point, due to the lower processing capacity of edge equipment compared to the cloud computing center, the response time of the cloud computing center becomes shorter than that of the edge cloud collaboration system.

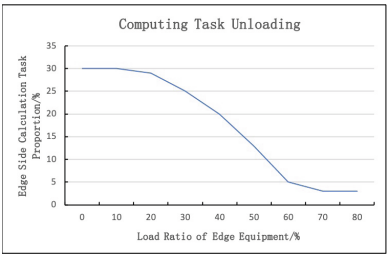
The influence of network delay while keeping other factors constant is illustrated in Fig. 10 (c). Generally, both methods show increasing response times as network delay grows. In scenarios with low network delay, the response time of the cloud computing center is shorter than that of the edge cloud collaboration system. Conversely, as network delay becomes higher, the edge cloud collaboration system gains a noticeable advantage in response time.



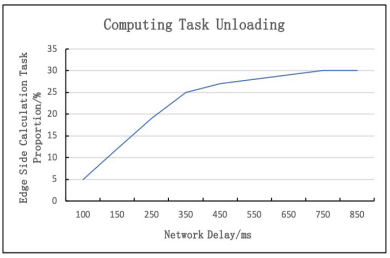
(a) Effect of data volume on task unloading ratio



(b) Influence of network bandwidth on the proportion of task unloading

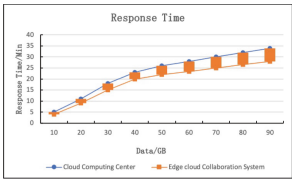


(c) Effect of edge device load on task unloading ratio

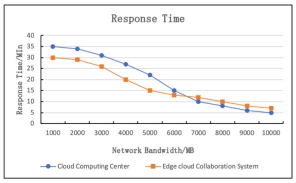


(d) Effect of network delay on the proportion of task unloading

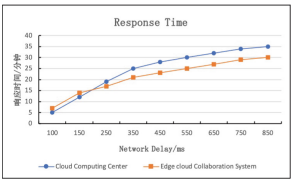
Fig. 9. Influence of different factors on computing task unloading



(a) Influence of data volume on system response time



(b) Influence of network bandwidth on system response time



(c) Effect of edge device load on task unloading ratio

Fig. 10. Influence of different factors on system response time

5 Conclusion

In this article, we present an adaptive unloading algorithm for edge cloud collaboration, consisting of the following key components:

DBN Integration: We discuss the integration of Deep Belief Networks (DBN) into the edge cloud collaboration system. This hierarchical processing of equipment-generated data is facilitated through DBN.

Adaptive Unloading Algorithm: We propose an adaptive algorithm for unloading data computing tasks. This algorithm takes into account computing

equipment performance evaluated using the PCM algorithm, analyzes network performance encompassing factors like network bandwidth and delay, and models the processing time of computing tasks. The algorithm is then realized using a Breadth-First Search (BFS) approach.

Simulation Experiments: We conduct simulation experiments to evaluate the advantages of DBN in reducing intermediate data. Additionally, we analyze how the edge cloud collaboration system impacts the unloading proportion of computing tasks under various conditions, including data volume, network bandwidth, edge device load, and network delay. These comparisons and analyses demonstrate that the edge cloud collaboration system exhibits shorter response times compared to traditional cloud computing centers under certain conditions.

This work highlights the potential benefits of DBN in reducing data overhead and showcases the advantages of edge cloud collaboration in terms of response time across different scenarios.

References

1. Paniagua, C., Delsing, J.: Industrial frameworks for internet of things: a survey. IEEE Syst. J. **15**(1), 1149–1159 (2020)
2. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutorials **19**(3), 1628–1656 (2017). https://doi.org/10.1109/COMST.2017.2682318
3. Zhang, S., He, P., Suto, K., et al.: Cooperative edge caching in user-centric clustered mobile networks. IEEE Trans. Mobile Comput. **17**(8), 1791–1805 (2017)
4. Aceto, G., Persico, V., Pescape, A.: A survey on information and communication technologies for industry 4.0: state of the art, taxonomies, perspectives, and challenges. IEEE Commun. Surv. Tutorials **21**(4), 3467–3501(2019)
5. Prathiba, S., Sankar, S.: Architecture to minimize energy consumption in cloud datacenter. Int. Conf. Intell. Comput. Control Syst. (ICCS) **2019**, 1044–1048 (2019). https://doi.org/10.1109/ICCS45141.2019.9065682
6. He, C., Yang, Y., Hong, B.: Cloud task scheduling based on policy gradient algorithm in heterogeneous cloud data center for energy consumption optimization. Int. Conf. Internet Things Intell. Appl. (ITIA) **2020**, 1–5 (2020). https://doi.org/10.1109/ITIA50152.2020.9312273
7. Chen, X., Zhang, J., Lin, B., Chen, Z., Wolter, K., Min, G.: Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments. IEEE Trans. Parallel Distrib. Syst. **33**(3), 683–697 (2022). https://doi.org/10.1109/TPDS.2021.3100298
8. Zhou, Q., et al.: Energy efficient algorithms based on VM consolidation for cloud computing: comparisons and evaluations. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), pp. 489–498 (2020). https://doi.org/10.1109/CCGrid49817.2020.00-44
9. Kosta, S., Aucinas, A., Hui, P., et al.: Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Proceedings of the 31st ACM INFOCOM, Piscataway, NJ, pp. 81–81. IEEE (2016)
10. Xing, N., Wu, P., Jin, S., Yao, J., Xu, Z.: Task classification unloading algorithm for mobile edge computing in smart grid. In: 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), pp. 1636–1640 (2021). https://doi.org/10.1109/IAEAC50856.2021.9390642

11. Tang, J., Lang, W.: Multi-user efficient computing task offloading and resource optimization. In: 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), pp. 15–19 (2021). https://doi.org/10.1109/ICSP51882.2021.9408712
12. Meng, Y., Li, J.: Task offloading and resource allocation mechanism of moving edge computing in mining environment. IEEE Access **9**, 155534–155542 (2021). https://doi.org/10.1109/ACCESS.2021.3129464
13. Si, F., Han, Y., Wang, J., Zhao, Q.: Connectivity verification indistribution systems using smart meter voltage analytics: a cloud-edgecollaboration approach. IEEE Trans Ind. Inf. **17**(6), 3929–3939 (2020)
14. Ren, J., Yu, G., He, Y., Li, G.Y.: Collaborative cloud and edgecomputing for latency minimization. IEEE Trans. Veh. Technol. **68**(5), 5031–5044 (2019)
15. Qin, M., et al.: Service-orientedenergy-latency tradeoff for IoT task partial offloading in MEC-EnhancedMulti-RAT networks. IEEE Internet Things J. **8**(3), 1896–1907 (2020)
16. He, Q., Cui, G., Zhang, X., Chen, F., Yang, Y.: A game-theoreticalapproach for user allocation in edge computing environment. IEEETrans. Parallel Distrib. Syst. **31**(3), 515–529 (2019)
17. Yang, H., Alphones, A., Zhong, W.D., Chen, C., Xie, X.: Learning-based energy-efficient resource management by heterogeneous RF/VLCfor ultra-reliable low-latency industrial IoT networks. IEEE Trans Ind. Inf. **16**(8), 5565–5576 (2019)
18. Mao, Y., Zhang, J., Song, S.H., Letaief, K.B.: Stochastic jointradio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans. Wireless Commun. **16**(9), 5994–6009 (2017)
19. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloadingfor mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. **34**(12), 3590–3605 (2016)
20. Wei, Z.: A summary of research and application of deep learning. Int. Core J. Eng. **5**(9), 167–169 (2019)
21. Tom, Y., Devamanyu, H., Soujanya, P., et al.: Recent trends in deep learning based natural language processing [Review Article]. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018)
22. Xi, X.F., Zhou, G.D.: A survey on deep learning for natural language processing. Acta Automatica Sinica **32**(2), 604–624 (2016)
23. Tomin, N., Kurbatsky, V., Negnevitsky, M.: The concept of the deep learning-based system. Artif. Dispat. Power Syst. Control Dispatch (2018)
24. Impedovo, D., Dentamaro, V., Abbattista, G., et al.: A comparative study of shallow learning and deep transfer learning techniques for accurate fingerprints vitality detection. Pattern Recogn. Lett. **151**(3), 11–18 (2021)
25. Lou, P., Liu, S., Hu, J., et al.: Intelligent machine tool based on edge-cloud collaboration. IEEE Access **8**, 139953–139965 (2021)
26. Wolf, J, Wolf, K.: Linux-Unix-Programmierung. Professionelles Website (2016)