Energy-saving service management technology of internet of things using edge computing and deep learning

Defeng Li¹,² · Mingming Lan¹ · Yuan Hu¹

Received: 11 September 2021 / Accepted: 28 January 2022 / Published online: 18 March 2022 © The Author(s) 2022

Abstract
The purpose is to solve the problems of high transmission rate and low delay in the deployment of mobile edge computing network, ensure the security and effectiveness of the Internet of things (IoT), and save resources. Dynamic power management is adopted to control the working state transition of Edge Data Center (EDC) servers. A load prediction model based on long-short term memory (LSTM) is creatively proposed. The innovation of the model is to shut down the server in idle state or low utilization in EDC, consider user mobility and EDC location information, learn the global optimal dynamic timeout threshold strategy and N-policy through trial and error reinforcement learning method, reasonably control the working state switching of the server, and realize load prediction and analysis. The results show that the performance of AdaGrad optimization solver is the best when the feature dimension is 3, the number of LSTM network layers is 6, the time series length is 30–45, the batch size is 128, the training time is 788 s, the number of units is 250, and the number of times is 350. Compared with the traditional methods, the proposed load prediction model and power management mechanism improve the prediction accuracy by 4.21%. Compared with autoregressive integrated moving average (ARIMA) load prediction, the dynamic power management method of LSTM load prediction can reduce energy consumption by 12.5% and realize the balance between EDC system performance and energy consumption. The system can effectively meet the requirements of multi-access edge computing (MEC) for low delay, high bandwidth and high reliability, reduce unnecessary energy consumption and waste, and reduce the cost of MEC service providers in actual operation. This exploration has important reference value for promoting the energy-saving development of Internet-related industries.

Keywords Internet of things devices · Mobile edge computing · Energy-saving service management · Long short-term memory model · Reinforcement learning · Dynamic power management

Introduction
Due to the rapid development of the mobile Internet, business application scenarios and business diversification have brought great challenges to wireless communication technology. New 5G business scenarios (such as autopilot and augmented reality) will lead to the explosive growth of mobile network data and put forward higher requirements for hardware performance, network delay, bandwidth, and reliability [1]. It is difficult for cloud computing with a centralized data center to meet the development needs of new business scenarios [2]. For example, some studies have combined Internet of things (IoT) technology with the smart home to building an automatic recommendation system for smart homes [3, 4]. Cloud computing has two major drawbacks in dealing with related scenarios. First, a long delay. Cloud computing adopts centralized data processing, which takes a long time. Mobile games, real-time voice translation, and automatic driving require extremely fast response speed. Otherwise, it will affect the user experience and even cause danger. Second, the high energy consumption of edge equipment. These two defects hinder the development of cloud computing and cause huge energy waste.
When the equipment at the edge of the network needs to be transmitted to the remote data center for centralized processing, its energy consumption may increase by 1.5 times, because they need network remote interaction in a high-power state [5]. To solve the current problems of cloud computing, in 2014, the industry proposed the concept of multi-access edge computing (MEC) to make up for the shortcomings of long cloud computing delay and difficulty in meeting the traffic density requirements of 5G new business scenarios [6]. In the follow-up research, some tasks of the cloud computing data center are set in the edge cloud for processing to realize the requirements of low delay and high bandwidth of the network and effectively reduce the program delay [7]. MEC server has a strong processing capacity because of its connection with a wireless network during operation. It reveals that it is very important to study MEC and reduce network computing delay in multiple scenarios.

Network edge devices will increase continuously in future mobile communication, and base stations, wireless access points, and other devices will have strong computing power. The energy consumption of the equipment remains at 70% in an idle state, which greatly increases the network operation cost. Most devices must rely on cloud service devices or edge devices to enhance processing capacity due to their limited computing and storage resources. Therefore, it is crucial to use MEC to improve the capability of mobile cloud computing (MCC). Pushing traffic, data, and network control to the edge of the network, generating more local data, and processing core data in the cloud computing data center can effectively reduce the energy consumption of mobile devices.

To solve the energy waste caused by cloud computing in the new business scenario, this exploration will express it in two parts: method and result. In the method part, the problems faced by IoT devices in the cloud computing environment are introduced, and MEC and long-short term memory (LSTM) models are proposed, based on which the load prediction model and power management strategy are designed. Then, these data are used for actual simulation training. In the result part, the optimal parameters of the model are determined. Moreover, the test and energy consumption of different load prediction models and different power management schemes are compared and evaluated, and the main research conclusions are drawn. This exploration will provide a reference for the energy-saving development of IoT devices.

At present, the application research of other algorithms in the energy-saving of IoT equipment in complex scenarios is less, and most of them are theoretical research. Therefore, this exploration is to make algorithm innovation based on previous research. Specific innovations are as follows. (1) The LSTM model is introduced to realize Edge Data Center (EDC) load prediction. LSTM can learn the periodic changes of time series, and model the random changes of time series to achieve more accurate prediction. (2) The load data of EDC and its surrounding EDC are used as model input. This method fully considers the geographic location information of EDC and realizes accurate load prediction. (3) Q-learning method is adopted. When EDC is idle, dynamic timeout threshold method is adopted. When EDC sleeps, N-policy is adopted. The global optimal effect can be achieved by learning the dynamic timeout threshold and N value through trial and error. The EDC server can consume the least energy while ensuring the system performance.

**Literature review**

**The research status of load prediction**

The results of load prediction will be provided to the dynamic power management model to more accurately understand the characteristics of the future load changes and make better power management decisions [8]. Since MEC has deployed server clusters with computing and storage capabilities in all access nodes of the mobile access network to form dense EDC, it leads to a certain energy consumption problem of the system. Accurate load prediction is crucial for resource allocation and utilization, based on which the correct decision can improve resource utilization and reduce energy consumption.

The load change has the above characteristics, through which the load prediction is realized through current methods.

There are four common methods for load prediction: (1) the autoregressive integrated moving average (ARIMA) model is the combination of the autoregressive (AR) model, moving average (MA) model, and ARMA. However, the three models require the time series to meet the stationary characteristics. ARIMA can be used for non-stationary time series [9]; (2) the seasonal autoregressive integrated moving average (SARIMA) model is similar to the ARIMA model, but it is mainly to solve time series with periodic changes. Unlike the ARIMA model, it can perform a long-term prediction with higher reliability [10]; (3) support vector machine (SVM) has achieved good results in linear and nonlinear time series data, and can achieve global optimization [11]; (4) recurrent neural network (RNN) is the preferred network for time series prediction because of the advantage of short-term memory compared with traditional machine learning methods. The method of time series prediction using RNN is similar to SVM.

The results of using RNN for the prediction of long-time series data are poor because the gradient is easy to disappear [12]. Unlike RNN, three gating structures are added in LSTM to solve the problem of gradient disappearance and capture the long-term dependence between time series. It has achieved good results in time series prediction and is widely used in text generation, stock prediction, and load predic-
Hence, LSTM is selected as the load prediction algorithm after the comparative analysis of common load prediction algorithms.

**Research status of energy-saving management**

The research content is the energy-saving management mechanism of EDC, which aims to save the huge energy consumption caused by the dense deployment of EDC and the idle time of the EDC server. At present, there are few related studies, but the research on energy-saving management of traditional cloud computing data centers starts early. EDC and cloud computing data centers are quite similar. Nashaat et al. [13] proposed virtual machine migration while ensuring the performance and efficiency of the cloud data center. According to the utilization of the Central Processing Unit (CPU), whether the host was in an overload state was detected by setting the threshold of CPU utilization. These virtual machines were placed in a migration list L in this process and then selected according to the shortest cycle time to complete the virtual machine migration. Finally, the whole virtual machine migration process was completed [14].

However, it is pointed out that too many virtual machines need to be migrated in this method. To solve this problem, some researchers proposed a single intelligent reinforcement learning method to dynamically select the optimal virtual machine and transfer the task from an overloaded host to the virtual machine. It is proved that this method can save energy consumption and reduce the number of virtual machine migrations [15]. Beloglazov et al. used the same method [16]. They used the Q-learning method in reinforcement learning during virtual machine selection. The state-space was the current CPU utilization of the host, and all virtual machines were mapped into actions based on the CPU utilization of the host. The overloaded host was placed in a list, and the reinforcement learning agent selected a virtual machine according to the method. At this time, the agent observed the utilization level of the new host and received a reward related to energy consumption [17].

**EMC load and energy saving**

People concern more about the MEC, and there are increasing studies on its theory, application, and production. Ismail et al. proposed the weakness of traditional cloud computing and pointed out that edge computing can decentralize the centralized network. Moreover, it eliminates bottlenecks and potential failure points by eliminating or reducing the importance of the centralized environment to make it better adapt to failure [18]. Flavio et al. pointed out that edge cloud can provide some potential new services, including location-based, IoT, data caching, big data, and sensor monitoring activities [19]. The Docker experiment proves that the network delay and delay jitter of edge computing is much lower than that of cloud computing. Meanwhile, it has the location awareness that cloud computing does not have, and its mobility and real-time connectivity are also better [20]. An edge computing device for data processing and information generation in the mobile environment based on wireless communication is proposed. It arranges MEC between the data collection unit and server and integrates MEC and DCU into ECD.

Unlike DCU, ECD has the function of processing and calculation [21]. Wen et al. focused on energy optimal application execution on cloud-assisted mobile platforms, intending to minimize the total energy consumed by mobile devices. When a mobile device runs an application, the computing energy can be minimized by optimizing the scheduling of its clock frequency [22]. KoK et al. proposed a distributed concurrent offloading mobile edge cloud computing framework, optimizing the computing-intensive and data-intensive tasks of IoT devices in the environment of IoT [23].

Today, dynamic power management technology is the main method to save the energy consumption of servers in idle time. Its basic principle is to turn the idle or low utilization server into the dormant state and wake up the server when a task arrives [24]. Related research shows that this method can reduce the energy consumption of the system. The random strategy, predictive strategy, and timeout strategy are often used in dynamic power management. The dynamic timeout threshold combined with the N-policy method can achieve a better energy-saving effect [25]. The N-policy means that when a system device is in a dormant state, it does not wake up immediately when a task arrives, but waits for the number of tasks to exceed N. It can save energy consumption during the state transition and reduce the loss caused by frequent switching equipment.

At present, user mobility and EDC location information are not considered in most studies of EDC workload prediction. These two reasons will lead to the correlation between adjacent EDCs. Thereby, the accuracy of using deep learning to predict the workload of an EDC can be improved if the historical data of its adjacent EDCs is added [26].

This idea is adopted to improve the "perception" ability of dynamic power management to the load change trend to accurately control the working state of the server and maximize energy savings.

**Research methods**

**IoT devices and algorithms**

The core of IoT is MCC, which is a model to realize sharing and convenient access to computer resources, including servers, storage, applications, and user services. This method can effectively reduce the interaction between work and ser-
vice providers, and provide and allocate resources quickly and accurately [27]. Figure 1 is the architecture of IoT cloud computing. All mobile devices are connected to the mobile Internet through the base station on the left. The mobile Internet sends instructions to the mobile operator service provider through the user’s information and request, and then sends a plurality of information to the regional agent on the right side of Fig. 1 through the central processor. The agent provides user authentication, authorization, and billing services. At present, almost all network mobile applications use the same data transmission processing method [28]. However, the increasing popularity of IoT cloud computing has brought multiple problems, including unreasonable resource allocation, non-standard effective allocation, and threats to security and privacy, which are the core problems in this field [29].

At present, a crucial related problem is that it still consumes more than 69% energy when the server is completely idle, while most data centers only provide 10–15% of the workload, leading to the data center losing 60–80% of the energy consumption [30]. The work mode switching of the server between the working state and the deep sleep state will increase the task delay by 1000 times. Therefore, studying the energy-saving management mode is crucial for the cloud computing of IoT devices [31].

MEC can use the wireless access network to provide the services needed by IT of telecom users and cloud computing functions nearby and create a carrier-class service environment with high performance, low latency, and high bandwidth. It can accelerate the rapid download of various contents, services, and applications in the network, and enable consumers to enjoy an uninterrupted high-quality network experience [32]. MEC can improve the user experience and save bandwidth resources. It also provides the third-party application integration by sinking the computing power to the mobile edge node, providing unlimited possibilities for the service innovation of mobile edge entrance. The deployment scheme can be divided into centralized control and distributed control according to whether there is a control center in the deployment scheme. Figure 2 shows its specific structure.

1. Centralized deployment scheme: it is a control center among base stations. When a task is unloaded to the base station group, the task decomposition, and processing in the base station group are scheduled by the control center. Centralized task scheduling can ensure that the task is balanced offloaded to the base station, and a load of each base station is balanced. However, centralized processing requires constant information interaction between base stations, leading to energy consumption and a decision delay. In the related existing literature, the energy consumption and delay caused by the information interaction are ignored, which focuses on the formulation of the task unloading strategy.

2. Distributed deployment scheme: there is no controlling entity in the base station group. When a task is unloaded to the base station group, the status information of all the base station interaction task queues in the whole network is needed to make the global optimal unloading strategy, and the information interaction cost of the global optimal unloading strategy is very large. The distributed task unloading strategy is more suitable for the distributed deployment scheme. However, in the existing literature, there is little related research or the cost of information interaction is ignored in the use of this strategy.

MEC provides real-time local area network information (such as network load and user location) for application developers and content developers. The real-time network information is adopted to provide context-perceived service to mobile users to improve user satisfaction and experience.
The forget gate reads the input \( x_t \) and determines which information is forgotten from the unit state. The process of the model is as follows. First, the forget gate is used in each structure [34]. LSTM has two structures: input gate, output gate, and forget gate. The sigmoid transmission states: appearance [33]. As shown in Fig. 3, it consists of three gating language recognition. It is often adopted in time series preprocessing and has achieved good results in natural language processing and with relatively long intervals and delays in time series. It is suitable for processing and predicting important events which needs energy saving.

LSTM is a kind of time recurrent neural network, and it is suitable for processing and predicting important events with relatively long intervals and delays in time series. It has achieved good results in natural language processing and language recognition. It is often adopted in time series prediction research because its structure can effectively solve the problem of long-term data dependence and gradient disappearance [33]. As shown in Fig. 3, it consists of three gating structures: input gate, output gate, and forget gate. The sigmoid function is used in each structure [34]. LSTM has two transmission states: \( c_t \) and \( h_t \). The transmitted \( c_t \) changes very slowly. Usually, the output \( c_t \) is the result of \( c_{t-1} \) transmitted from the previous state plus some numerical values, while \( h_t \) often differs greatly under different nodes.

This structure is the most appropriate for MEC problems. The process of the model is as follows. First, the forget gate determines which information is forgotten from the unit state. The forget gate reads the input \( x_t \) and the state \( h_{t-1} \) of the previously hidden layer. After the activation function, \( f_t = 1 \) means complete reservation; \( f_t = 0 \) means complete forgetting and the obtained new state \( i_t \). The specific equation reads:

\[
f_t = \sigma_1(W_f \cdot [h_{t-1}, x_t] + b_f),
\]

(1)

\[
C_{t-1}.
\]

(2)

In the input gate, new information is stored in the unit state will be determined, and the state \( x_t \) and state \( h_{t-1} \) of the previous hidden layer will be input. After the activation function is output, \( C_t \) is obtained as follows:

\[
C_t = \phi_1(W_c \cdot [h_{t-1}, x_t] + b_c).
\]

(3)

Then, the previous unit state \( C_{t-1} \) is updated to the current cell state \( C_t \). The equation reads:

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot C_t.
\]

(4)

Finally, the current unit state of the output part is determined through the activation function \( \phi_3 \) of the output gate. The unit state is activated by the activation function \( \phi_2 \) in output gate, as shown in Eq. (6). By multiplying the results of these two steps, the final output part \( h_t \) can be determined:

\[
o_t = \sigma_2(W_o \cdot [h_{t-1}, x_t] + b_o),
\]

(5)

\[
h_t = o_t \cdot \phi_2(C_t),
\]

(6)

\[
\phi_2(x) = \max(0, x).
\]

(7)

LSTM has been applied to more and more fields with the rapid development of artificial intelligence. It is difficult for single-layer LSTM to meet complex practical requirements. For example, part of the time series depends on the state of future time and past time. Meeting these two parts will get accurate prediction results. Therefore, bidirectional LSTM comes into being. Figure 4 shows the network structure of bidirectional LSTM, showing that the outputs and inputs of the forward propagation layer and backward propagation layer of bidirectional LSTM are connected, and their weights are shared. There are four connection layers.

To obtain the final output result, the output of the forward hidden layer at each time in the forward propagation layer is obtained first. Then, the output of the backpropagation layer to the hidden layer at each time is calculated and saved. Finally, the final output result is obtained by combining the output of the corresponding time of the forward propagation layer and the backpropagation layer at each time.

Besides, the single-layer LSTM may have the problem of underfitting in dealing with practical problems, so the deeper LSTM appears. The multi-layer LSTM takes the output of
Output layer
Backward layer
Forward layer
Input layer
W1
W2
W3
W4
W5
W6

W2 W2 W1 W1
W3 W3
W4 W4
W5
W6
W5 W5
W6
W6

Third layer
Second floor
First floor
W1
W2
W2
W1
W1
W1
W2
W2
W2
W3
W6
W6
W6
W6
W5
W7
W7
W7
W7

Fig. 4  Structure diagram of bidirectional LSTM model

Fig. 5  Structural diagram of multilayer LSTM model

the previous layer as the input of the next layer. Figure 5 is the structure diagram of its three-layer LSTM. However, the more layers in the deep LSTM does not mean the better effect, because the more layers will increase the time complexity of the model and overfitting will occur. Generally, the number of layers of LSTM shall not exceed 6.

Design of load prediction model

The MEC prediction model is designed according to the structure of LSTM. Figure 6 shows that the prediction model also includes the input layer, output layer, and hidden layer. There is a potential correlation between MEC and neighbor MEC due to the mobility of users. To further mine this correlation, in the load prediction of each MEC, the input data of the model includes not only the load data of the current MEC but also that of the neighbor mobile edge devices. The original time series data are mapped to a fixed quantity after passing through the input layer. ReLU is used as the activation function between the input layer and the hidden layer. Unlike the Sigmoid, the calculation amount is smaller, so the training speed will be faster, which will cause the sparsity of the network and effectively alleviate the problem of data over-fitting. The prediction results are output in the output layer after the data passes through the LSTM layer.

Input data
Hidden layer
Output data
X1
LSTM
LSTM
Y1
X2
LSTM
LSTM
Y2
X3
LSTM
LSTM
Y3

Fig. 6  Structure chart of a load prediction model based on LSTM

Fig. 7  The process of generating time series from historical load data

When the network is overfitted, the methods that can be used are regularization, dropout, and batch normalization layer (BN). Regularization includes L1 regularization and L2 regularization, and L2 regularization is used in LSTM. Moreover, in the use of dropout and BN layers, it is essential to focus on the different settings on the train set and the test set. For example, dropout is set to 0.5 on a train set; dropout is removed on the validation set and test set.

Figure 7 shows that the load of num_steps time interval is used in the model to predict the load of the \( n + 1 \) interval because the MEC load of the \( t \)-th period has the strongest correlation with the load of the nearest \( t + 1 \)-th time interval. The LSTM model is used and the dropout layer \([34]\) is added to this layer. The fixed value is 0.5. During model training, it is removed from the random LSTM unit in the neural network. After the output of the model is obtained through the above steps, the difference between the predicted output value and the actual load value is evaluated, and the mean square error is selected as the loss function of the model. \( \text{Predict} \) represents the predicted value, \( \text{true} \) is the real load value, and \( \theta \) is the parameter of the model.

Based on the above contents, the number of neural units in the hidden layer of the training network is determined as follows. The increase of the number of hidden layer neural units will increase the network width and reduce the network error. It is easier to reduce the error by increasing the number of hidden layer nodes than by increasing the number of hidden layers. The number of hidden layers is determined in the result part. The number of neurons in the hidden layer is affected by the number of hidden layers. In the results part, the optimal parameters are given by the length of training time. At present, the best parameters are that the number of hidden layers and neurons is 4 and 128, respectively. The experiment
will be analyzed in combination with the research content. A small-scale data set extracted from the original dataset will be used to verify the accuracy of the network before parameter adjustment. The original dataset of power supply for this exploration is Shanghai 6 × 24 data from March 20, 2020, to March 25, 2020, from the "Shanghai public data open platform".

**Design energy-saving scheme of IoT**

Dynamic power management framework must learn the input of the environment and constantly adjust the power management strategy to adapt to the changing load of the EDC server. The goal of dynamic power management in the EDC is to minimize energy consumption while ensuring system performance. This goal is achieved through the dynamic power management model based on Q-learning. Figure 8 shows the dynamic power management model. The Power Manager in the figure is an Agent in Q-learning, which will select different actions according to the different states of service queue (SQ) and server (SP) in EDC. Next, the state space, action space and reward function of the dynamic power management framework will be introduced to explain how the system achieves the balance of performance and energy consumption through the dynamic power management framework based on Q-learning.

State-space: during research, the working state of the system consists of three parts, namely, service request generator (SR), SQ, and SP. The working state includes active, idle, and dormant. SR is adopted to generate job requests. The job request is the load request sent by taxi to EDC. The workload predictor transmits the predicted load information to the Power Manager. The job request generated by SR will be stored in SQ, and the job in SQ will be processed by SP in FIFO order. Action space: the system will adopt the timeout threshold strategy when the SP is idle and the number of jobs in SQ is 0. At this time, the action space is a series of timeout thresholds. Power Manager uses the greedy method to dynamically select the timeout threshold timeout from the action space, so that the system can achieve the tradeoff between performance and energy consumption. The system will adopt the N-policy strategy when SP is in a sleep state. The action space is the list of all possible N, and the range of N value is from 0 to the average job of each training episode. The SP side is awakened from the sleep state to the active state when the selected N value is 0. Otherwise, it will continue to sleep and store the arriving jobs in SQ. It will become active when the number of jobs in SQ is greater than N.

The dynamic power management strategy includes SR, SQ, and server edge (SE) in MEC. SR is used to generate task requests, and job requests adopt load requests sent to EDC. Workload Predictor is a process of transferring the information of load prediction to a dynamic power manager. Job requests generated by R are stored in SQ. The job in SQ is used by the state space of the SE processing system in the FIFO order. The specific equation reads:

\[ S_t = [S_{qt}, S_{p}] \]  

(8)

\[ S_{qt} \] is the number of jobs in SQ, and its range is from 0 to the maximum queue length max(\_ queue size). The equation of \( S_{qt} \) reads:

\[ S_{qt} = [0, 1, 2... \text{max(queue\_size)}]. \]  

(9)

SE is idle when the number of jobs in SQ is 0. When it is less than max() queue size, it indicates that there are resources allocated to jobs in SQ in SE; otherwise, it indicates that there are insufficient resources in the current system, and the job is in the state of queuing for resource release. Figure 9 presents the process. \( S_{pt} \) represents the working state of SE, including active, idle, and sleep. The specific equation reads:

\[ S_{pt} = \{\text{active, idle, sleep}\}. \]  

(10)

SE will change its working state according to the state of SQ and the strategy adopted by the power manager. Figure 9 is the flow chart of the system processing job, showing the specific transformation.

In the IoT data, the characteristics of mobile users are considered. The data come from the China traffic information network platform, and 925 IoT taxis are calculated for real moving trajectory. First, the original task request data are processed into mobile edge load data, and the original data are from October 2018 to November 2018. The content is specific geographic location information and specific task request of the taxi. The mobile EDC is simulated to determine the processing range of each moving edge. 39 small hexagons are used to form a large polygon. Each hexagon is a specific data processed by moving edge, and each data interval is 1 km, as shown in Fig. 9.
Waiting for the system to allocate resources, the time delay increases

Fig. 9 Optimization flow chart of power management strategy

SE is converted to Idle state

SE processes data in FIFO state

Waiting for the system to allocate resources, the time delay increases

End

Then, a load of each mobile edge center at different times is counted according to the distance between the service request issued by vehicles at different time points and the mobile EDC. Then, each request is located. Data processing tools such as pandas and numpy in Python are used. The total load of 39 mobile edge data is calculated based on the above steps. Then, the matplotlib tool is used for visualization. The simple processing of the data shows that the load distribution is highly concentrated in some areas. Among them, the ratio of the training dataset to test data is 8:2 (Fig. 10).

Experimental environment and performance evaluation

1. Experimental simulation environment: the computer system version is Ubuntu 14.04 OS. The TensorFlow framework is used for simulation and its version is TensorFlow 1.4.0. Python 3.6.3 is used for programming. The computer hardware includes a GPU graphics card, NVIDIA 1080Ti, with 11G video memory and 16G random-access memory (RAM). Table 1 presents the details.

| Simulation Environment | Parameter            |
|------------------------|----------------------|
| Computer system version| Ubuntu 14.04 OS      |
| GPU graphics card model | NVIDIA 1080Ti       |
| Video memory           | 11G                  |
| RAM                    | 16G                  |
| Programming language   | Python 3.6.3         |

2. Model performance evaluation: common evaluation indicators for time series prediction tasks include the mean absolute percentage error (MAPE) and root mean square error (RMSE). \( R = (r_1, r_2, \ldots, r_n) \) and \( P = (p_1, p_2, \ldots, p_n) \) represent the real load and the load predicted by the model respectively. The calculation equations of the two indicators read:

\[
MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{r_i - p_i}{r_i} \right|, \quad (11)
\]

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (r_i - p_i)^2}. \quad (12)
\]

The above equations show that MAPE cannot be calculated when it is 0, it is asymmetric, and its penalty for negative errors is greater than that for positive errors. Therefore, on this basis, symmetric mean absolute percentage error (SMAPE) is introduced. SMAPE is used as the evaluation indicator of experimental results. The calculation equation reads:

\[
SMAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|r_i - p_i|}{|r_i| + |p_i|}. \quad (13)
\]

As an indicator to measure the distance between the predicted value and the real value, the smaller the SMAPE is, the higher the accuracy of the model prediction is.

3. Performance evaluation of power management strategy: the purpose of the exploration is to save energy consumption of the MEC system. However, it is essential to ensure the overall performance of the system to meet the needs of users while saving energy. Thereby, time delay and energy consumption are used to evaluate the effect of five different power management strategies. Finally, the trade-off curve of time delay and energy consumption is used to comprehensively evaluate the model. The application scenario is a large geographical range of 39 different MEC, and each MEC load, geographical location, and the number of users covered are different. To comprehensively evaluate the MEC energy-saving management mechanism, the average delay of each job in 39
MEC is calculated after the average time delay of each job in each MEC is determined. Finally, the relationship between the number of jobs and the total delay of jobs is obtained. That is, if the delay of each job in each MEC is \( t_i \), the final average delay \( t_{avg} \) is expressed by the following equation:

\[
t_{avg} = \frac{1}{n} \sum_{i=1}^{n} t_i, \quad n = 37. \tag{14}
\]

The energy consumption in MEC comes from many aspects, such as from the process of job execution, from the dynamic operation, and refrigeration, lighting, and ventilation equipment. In this experiment, the energy consumption in the process of the job execution and the dynamic operation is mainly considered. If the total energy consumption is represented by \( E_{total} \), the energy consumption generated during the job execution is represented by \( E_{exe} \), and the dynamic operation energy consumption is mainly generated by the state transition of the server in MEC, which is represented by \( E_{dynOp} \). The equation of total energy consumption reads:

\[
E_{total} = E_{exe} + E_{dynOp}. \tag{15}
\]

\( E_{exe} \) is determined by the time the server stays in the current working state and can be expressed as Eq. (16), active \( E \), idle \( E \), and sleep \( E \) are the energy consumption of servers in the active, idle, and sleep states, respectively. \( a_{duration} t \), \( i_{duration} \), and \( s_{duration} \) are the length of time that the server is in the active, idle, and sleep state, respectively. Once the execution energy consumption is determined, the total energy consumption of the server can be determined:

\[
E_{exe} = \begin{cases} 
E_{active} * t_{a_{duration}}, \text{ state } = \text{ active} \\
E_{idle} * t_{i_{duration}}, \text{ state } = \text{ idle} \\
E_{sleep} * t_{s_{duration}}, \text{ state } = \text{ sleep} 
\end{cases}. \tag{16}
\]

The average energy consumption of each job in MEC can be obtained after the total energy consumption and the number of jobs are determined. If the energy consumed by each job in each MEC is \( P \), the equation of the final average energy consumption \( P_{avg} \) reads:

\[
P_{avg} = \frac{1}{n} \sum_{i=1}^{n} P_j, \quad n = 37. \tag{17}
\]

It is essential to evaluate the experimental effect comprehensively. After the average delay and average energy consumption are calculated, the trade-off curve of delay and energy consumption is needed to evaluate the proposed model more reasonably. The curve is obtained from the average delay and energy consumption of each job. The energy consumption of several different power management strategies under the same time delay can be obtained to understand the comprehensive effect of different strategies more intuitively.

The time optimization of the algorithm proposed adopts AdaGrad optimization solver, because the parameters involved have low-frequency characteristics, and the optimizer has a large amount of updating for low-frequency parameters. Matlab software is used to compare the time-saving proportion of common optimization solver algorithms, including Nadam, adam, RMSprop, and AdaDelta.

**Results and discussion**

**Determination of model parameters**

The algorithm verification accuracy of the small-scale train set in this experiment is 1. The optimization time ratio of the selected AdaGrad optimization solver is 33%, while the optimization time ratio of Nadam, adam, RMSprop, and AdaDelta is 28, 21, 18, and 29%, respectively, which are less than that of the AdaGrad optimization solver. It reveals that the time optimization solver selected in this experiment is more appropriate.

Figure 11 shows that as the dimension of input features increases, the smaller the SMAPE value is, the higher the accuracy of the model is, because the status range of each EDC service is smaller. The accuracy of the model increases with the increase of LSTM layers, while the training time also increases. The value of SMAPE decreases with the increase of window length, but the best performance is between 30 and 45, and the result is relatively stable.

In a reasonable range, the memory utilization of the system increases with the increase of the parameter batch size, and the error fluctuation of model training is relatively less. When the batch size is 128, the training time is 788 s, and the value of SMAPE is relatively small, 13.02%. After the model has trained 350 epochs, the prediction performance of the model is stable at about 115, suggesting that the number of data can make the model close to the convergence state. Based on the above results, it is found that the model has the best performance when the feature dimension is 3, the number of LSTM network layers is 6, the length of time series is 30–45, the value of batch-size is 128, the training time is 788 s, the number of units is 250, and the number of times is 350. The selected AdaGrad optimization solver shows excellent optimization performance and model accuracy, and the model prediction performance is stable, which lays a foundation for the prediction accuracy of the overall model in the following sections.
Fig. 11 Determination of the optimal parameters of the model

Model performance comparison

The best parameters in 4.1 are used. Here, the prediction errors of the LSTM model and the currently widely used ARIMA model [35] are compared. Figure 12 shows the comparison results. It shows that compared with the traditional ARIMA method, in most mobile edge load prediction, the prediction accuracy of the model based on LSTM is higher, with an average of 4.21%. This shows that the selected LSTM model has good prediction performance and the prediction feasibility of the overall scheme is high.

Analysis of load performance of IoT

Figure 13 shows the comparison results of delay differences of different power management strategies under different loads and the same load. It shows that with the increase of load, the delay increase of the static threshold method is increasingly larger, and the delay increase of power management strategy with load prediction is smaller. The data show that the proposed strategy can reasonably control the growth rate of task response delay according to the information provided by load prediction, and can minimize the overall energy consumption of the system, followed by ARIMA + dynamic power management strategy with load prediction. The proposed dynamic power management strategy with load prediction can save more energy consumption. The energy-saving scheme proposed is feasible.

Figure 14 is the energy consumption comparison results of different power management methods under different loads and the same load. It shows that the energy-saving effect of the proposed method is not obvious when the total load is
small, but the method can show advantages over other methods when the load increases. When the load reaches 100,000, the dynamic power management method with LSTM load prediction can reduce 12.5% energy consumption compared with the dynamic power management method with ARIMA load prediction, and can reduce 44.75% energy consumption compared with the static threshold method. The more the energy consumed by the system is, the smaller the average delay of task response is.

Moreover, due to the increase of load prediction, the minimum average energy consumption of the proposed method and the ARIMA + RL method (the minimum value in the range of horizontal axis variation) is smaller than that of other methods. It suggests that the proposed energy-saving mechanism with load prediction can still work normally under the condition of small energy consumption, while other methods need to consume more energy to meet the needs of edge devices. Furthermore, the proposed energy-saving mechanism achieves good results in the time delay under the same energy consumption. Compared with the other four energy management methods, the proposed method can achieve a better balance between time delay and energy consumption.

Fig. 13 The influence of data size on recognition effect. A1 is fixed timeout model. Note: A2 is dynamic timeout model, A3 is dynamic power management strategy, A4 is ARIMA + dynamic power management, A5 is LSTM + dynamic power management, A6 is Q-learning model.

Conclusion

Aiming at the problem of high energy consumption of MEC, a multi-layer LSTM load prediction model is innovatively established after considering the correlation between MEC geographical location information and adjacent MEC load. The advantage of the proposed load prediction model based on LSTM is that it can achieve accurate load prediction effect. The number of neighbors of MEC and the sparsity of load data will affect the prediction results. The more the neighbors are, the denser the load data are, and the more accurate the prediction results are. The disadvantage is that when there are few load data, the prediction accuracy of the model is not high. A dynamic power management energy-saving mechanism based on reinforcement learning method is proposed. This mechanism can significantly reduce the energy consumption and delay of MEC. This exploration provides a theoretical basis for the application of MEC in the IoT. The research results are expected to be applied in IoT. Although the energy-saving management of edge data center can learn from cloud data center, the following improvements are still needed. First, the input dimension of the model and the sparsity of load data will affect the accuracy of load prediction model. Therefore, in future research, the accuracy of the load prediction in edge data centers can be improved to save more energy. Then, in the research on energy-saving manage-
ment of edge data center, only dynamic power management strategy is adopted, and virtual machine migration can be considered. The two can be combined to realize complementary, energy-saving and green MEC. Finally, when the load prediction increases, the minimum average energy consumption of this method is compared with ARIMA + RL method under the minimum value in the variation range of horizontal axis without considering all the data in the variation range of horizontal axis. Therefore, in the future, the accuracy of load prediction of edge data centers and virtual machine migration in energy-saving management of edge data centers will be further discussed and evaluated to continuously reduce the energy consumption of mobile Internet applications. In the comparison when the load prediction increases, it is necessary to add all the data within the variation range of the horizontal axis and the model state when the attack increases and decreases sharply, so as to put forward an energy-saving management scheme more in line with the actual requirements.

Acknowledgements This work was supported by the Key Technology Research Program of the Higher Education Institutes of Henan Province, China (Grant nos. 20B430008 and 18B580002); the Key Science and Technology Program of Henan Province (Grant no. 162102210272). The authors acknowledge the help from the university colleagues.

Author contributions All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

Declarations

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Chekired DA, Togou MA, Khoukhi L, Ksentini A (2019) 5G-slicing-enabled scalable SDN core network: toward an ultra-low latency of autonomous driving service. IEEE J Sel Areas Commun 37(8):1769–1782
2. Kim HW, Jeong YS (2018) Secure authentication-management human-centric scheme for trusting personal resource information on mobile cloud computing with blockchain. Human-cent Comput Inf Sci 8(1):11–25
3. Gladence LM, Anu VM, Rathna R, Brumancia E (2020) Recommender system for home automation using IoT and artificial intelligence. J Ambient Intell Hum Comput 12:1–9
4. Gladence LM, Anu VM, Revathy S, Jeyanthi P (2021) Security management in smart home environment. Soft Comput 25:1–9
5. Hu L, Miao Y, Wu G, Hassan MM, Humar I (2019) iRobot-Factory: an intelligent robot factory based on cognitive manufacturing and edge computing. Future Gener Comput Syst 90:569–577
6. Rahman A, Srikumar V, Smith AD (2018) Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks. Appl Energy 15(212):372–385
7. Taheri S, Ahmadi A, Mohammadi-Ivatlo B, Asadi S (2021) Fault detection diagnostic for HVAC systems via deep learning algorithms. J Build Eng 35:102111
8. Taheri S, Jooshaki M, Moeini-Aghtaie M (2021) Long-term planning of integrated local energy systems using deep learning algorithms. Int J Electr Power Energy Syst 129:106855
9. Zhou Z, Liao H, Zhao X, Ai B, Guizani M (2019) Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty. IEEE Trans Veh Technol 68(9):8322–8335
10. Petukhova T, Ojkic D, McEwen B, Deardon R, Poljak Z (2018) Assessment of autoregressive integrated moving average (ARIMA), generalized linear autoregressive moving average (GLARMA), and random forest (RF) time series regression models for predicting influenza A virus frequency in swine in Ontario, Canada. PLoS ONE 13(6):e0198313–e0198324
11. Huang Y, Zhao L (2018) Review on landslide susceptibility mapping using support vector machines. CATENA 165:520–529
12. Ding G, Qin L (2020) Study on the prediction of stock price based on the associated network model of LSTM. Int J Mach Learn Cybern 11(6):1307–1317
13. Nashaat H, Ashry N, Rizk R (2019) Smart elastic scheduling algorithm for virtual machine migration in cloud computing. J Supercomput 75(7):3842–3865
14. Burd TD, Brodersen RW (2019) Processor design for portable systems. J VLSI Signal Process Syst Signal Image Video Technol 13(2):203–221
15. Lee Y, Kim W, Moo K et al (2019) A mobile edge computing device to support data collecting and processing from IoT. In: 2019 international conference on electronics, information, and communication (ICEIC). IEEE, pp 1–3
16. Wen Y, Zhang W, Luo H (2018) Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones. In: 2012 proceedings IEEE Infocom. IEEE, pp 2716–2720
17. Ko K, Son Y, Kim S et al (2017) DisCO: a distributed and concurrent offloading framework for mobile edge cloud computing. In: 2017 ninth international conference on ubiquitous and future networks (ICUFN). IEEE, pp 763–766
18. Haghhighi MA, Maen M, Haghparast M (2019) An energy-efficient dynamic resource management approach based on clustering and meta-heuristic algorithms in cloud computing IaaS platforms. Wirel Pers Commun 104(4):1367–1391
19. Khattar N, Sidhu J, Singh J (2019) Toward energy-efficient cloud computing: a survey of dynamic power management and heuristics-based optimization techniques. J Supercomput 75(8):4750–4810
20. Li L, Jin S, Huo Z (2018) A novel energy saving strategy with N-policy sleep mode in cognitive radio networks. J Internet Technol 19(5):1371–1382
21. Akherfi K, Gerndt M, Harroud H (2018) Mobile cloud computing for computation offloading: issues and challenges. Appl Comput Inform 14(1):1–16
22. Noor TH, Zeadally S, Alfazi A, Sheng QZ (2018) Mobile cloud computing: challenges and future research directions. J Netw Comput Appl 1157:70–85. https://doi.org/10.1016/j.jnca.2018.04.018
23. Mohammed MA, Ğaçpuz N (2017) A novel approach of reducing energy consumption by utilizing enthalpy in mobile cloud computing. Stud Inform Control 26(4):425–434
24. Peng H, Wen W-S, Tseng M-L, Li L-L (2019) Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment. App Soft Comput 80:534–545. https://doi.org/10.1016/j.asoc.2019.04.027
25. Tran TX, Hajisami A, Pandey P, Pompili D (2017) Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges. IEEE Commun Mag 55(4):54–61
26. Fischer T, Krauss C (2018) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669
27. Zhou Y, Huang Y, Pang J, Wang K (2019) Remaining useful life prediction for supercapacitor based on long short-term memory neural network. J Power Sources 44(2):27149–27153
28. Liu J, Gong X (2019) Attention mechanism enhanced LSTM with residual architecture and its application for protein-protein interaction residue pairs prediction. BMC Bioinform 20(1):609–616
29. Zhang Q, Lin M, Yang LT, Chen Z, Khan SU, Li P (2018) A double deep Q-learning model for energy-efficient edge scheduling. IEEE Trans Serv Comput 12(5):739–749
30. Li T, Wu J, Dang A, Liao L, Xu M (2019) Emission pattern mining based on taxi trajectory data in Beijing. J Clean Prod 206:688–700
31. Schubert AL, Hagemann D, Voss A, Bergmann K (2017) Evaluating the model fit of diffusion models with the root mean square error of approximation. J Math Psychol 77:29–45
32. Maiseli BJ (2019) Optimum design of chamfer masks using symmetric mean absolute percentage error. EURASIP J Image Video Process 2019(1):74–83
33. Ali Z, Jiao L, Baker T, Abbas G, Abbas ZH, Khaf S (2019) A deep learning approach for energy efficient computational offloading in mobile edge computing. IEEE Access 7:149623–149633
34. Zhang G, Zhang W, Cao Y, Li D, Wang L (2018) Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. IEEE Trans Ind Inform 14(10):4642–4655
35. Zhao P, Tian H, Qin C, Nie G (2017) Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access 5:11255–11268

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.