SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2026, Vol. 69, Iss. 3, 132304:1–132304:18
https://doi.org/10.1007/s11432-025-4664-y

A knowledge-driven model selection and resource management method with information entropy

Dan WANG[1,2], Zhenshen LIANG[2] & Bin SONG[1*]

[1]*State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China*
[2]*Hangzhou Institute of Technology, Xidian University, Hangzhou 311231, China*

Received 14 April 2025/Revised 22 August 2025/Accepted 4 November 2025/Published online 29 January 2026

Abstract The rapid advancement of the Internet of Things (IoT), mobile edge computing (MEC), and artificial intelligence (AI) technologies is accelerating the emergence of novel services and applications with substantial computational demands. This phenomenon has resulted in an exponential growth in the complexity of devices and networks, which has engendered the demand for the evolution of network resource management paradigms. Traditional resource management strategies focus mainly on reducing device delay and energy consumption, but relatively lack attention to the selection and sharing of intelligent models among multiple devices. To address these challenges, this paper proposes an information plane (IP) framework for evaluating the representation capability of models to support complex network resource management services. Furthermore, we design a deep reinforcement learning (DRL)-based resource management scheme, which considers delay and energy consumption and takes the model representation capability as a new indicator, thereby achieving a knowledge-driven intelligent model selection and resource management. Specifically, we first use the IP framework and blockchain technology to introduce model evaluation metrics and identify the best-performing model in the network. Then, we design the resource management problem as a Markov decision problem (MDP) to achieve the optimal decision and resource allocation of the system. Experimental results show that compared to traditional resource management schemes in MEC-assisted IoT scenarios, the proposed scheme can effectively select a suitable intelligent model for IoT devices and optimize the cost of the system, and its performance is better than other schemes. Specifically, compared with traditional schemes, the proposed scheme achieves up to a 7% improvement in model representation capability, while reducing delay by 25% and energy consumption by as much as 65%.

Keywords information plane, resource management, edge computing, deep reinforcement learning, blockchain

Citation Wang D, Liang Z S, Song B. A knowledge-driven model selection and resource management method with information entropy. Sci China Inf Sci, 2026, 69(3): 132304, https://doi.org/10.1007/s11432-025-4664-y

1 Introduction

Today, high-performance and latency-sensitive Internet of Things (IoT) applications, such as facial recognition, virtual reality, autonomous driving, and smart home appliances, are gaining increasing attention, and their successful deployment is pivotal to improving people's quality of life. However, the development of these applications is based on low-latency real-time data transmission, efficient and accurate task processing, and effective resource management. Fortunately, the advent of mobile edge computing (MEC) has become one of the key technologies that can effectively implement these capabilities in the IoT. It can improve the performance and user experience of these applications through characteristics such as low latency, bandwidth optimization, distributed computing, and flexible deployment [1].

However, MEC is achieved by deploying computing and storage resources at the edge nodes of IoT devices, which are typically limited. In practical scenarios, the large-scale interconnection of numerous IoT devices inevitably leads to data interaction and energy consumption, leading to multiple mobile devices sharing limited communication, computing, including delay, energy consuand caching resources [2]. Therefore, within the purview of MEC-facilitated IoT environments, resource management has consistently emerged as a pivotal area of research.

In the current research field, traditional studies primarily focus on optimizing several core metrics to improve system performance. First, minimizing system delay is crucial, as it is essential for real-time applications and to improve the user experience [3–6]. Second, reducing energy consumption is also a key research focus, which not only contributes to sustainable development but also extends the useful life of devices limited in energy [7–11]. Additionally, researchers strive to reduce the weighted sum of delay and energy consumption [12–15], recognizing

* Corresponding author (email: bsong@mail.xidian.edu.cn)

that optimizing one performance indicator often requires a compromise on another. Therefore, it is particularly important to seek a balanced solution. Beyond these primary goals, researchers have introduced various auxiliary metrics to gain a more comprehensive understanding of system performance. These include maximizing the quality of service (QoS) for all users, ensuring high data transmission rates, low latency, and strong reliability [16, 17]. In fields requiring high-precision predictions and outcomes, such as autonomous driving systems and decision support systems, system accuracy is also critical [18]. These studies optimize these key performance indicators through well-designed resource allocation schemes, aiming to achieve the overall performance of the network by optimizing single or multiple indicators [16–21].

Although various metrics have been used to optimize the network performance of MEC-assisted IoT systems, new network performance optimization problems have emerged with the development of deep learning and artificial intelligence technologies. For example, there are various intelligent models in the network that can make data processing more efficient, decision-making smarter, and data security enhanced. However, different tasks exhibit significant performance differences when using different intelligent models, so it is essential to consider which model to choose based on the task requirements. Although various studies have investigated the performance of different models in multiple tasks [22–24], there has not yet been a suitable way to evaluate intelligent models, nor to enable intelligent model selection in networks. Additionally, current resource management strategies relatively lack focus on the selection and sharing of intelligent models among multiple devices. The question of how to make reasonable model selection has become a key to improving the intelligence and decision-making quality of network resource management.

In this paper, we investigate the problem of resource management in a multi-device MEC-assisted IoT scenario, considering how to select an optimal intelligent model according to specific requirements, and optimize system delay, energy consumption, and model representation capability. Motivated by the studies in [25, 26], we introduce a model selection framework based on the information plane (IP), which quantifies the representation capability of intelligent models by evaluating the information entropy of different task models to obtain the model accuracy index. Leveraging this framework, we evaluate the model representation capabilities and employ blockchain technology to aggregate and share the capability metrics of all models across the network. This facilitates the identification of the optimal model for executing the current task with the highest accuracy. Subsequently, we propose a deep reinforcement learning (DRL)-based model selection and resource management strategy. Specifically, we formulate this problem as a Markov decision process (MDP) and harness the DRL method to collaboratively optimize the model representation capability, delay, and energy consumption of devices, thereby achieving a knowledge-driven intelligent model selection and resource decision-making process. The main contributions of this paper are as follows.

- We propose a knowledge-driven innovative model selection and resource management scheme that utilizes an IP framework and DRL algorithm, takes into account multi-objective collaborative optimization, including delay, energy consumption, and model representation capability to achieve accurate intelligent model selection and resource management optimization, and provides an effective offloading strategy for MEC-assisted IoT devices.
- We first construct an IP-based model selection framework and integrate a novel knowledge-driven evaluation mechanism into this framework, which utilizes information entropy to evaluate model representation capability. Through this framework, we introduce the model representation capability as an optimization objective, and collect and share the capability indicators of all models in the network through blockchain technology, thereby helping IoT devices make the best model choices.
- We then define the model selection and resource management problem as the MDP and aim to balance the model representation capability, delay, and energy consumption of all devices. To address this problem, we adopted the proximal policy optimization (PPO) algorithm to train the network and learn the optimal model selection and resource allocation strategies.
- The simulation results show that compared to existing traditional methods, our proposed solution exhibits better convergence and stability in MEC-assisted IoT scenarios. In addition, the scheme effectively selects suitable intelligent models for IoT devices and demonstrates excellent performance in the comprehensive evaluation of system model representation capabilities, delay, and energy consumption.

2 Related work

This section reviews related work on MEC in recent years and surveys the related research on exploring neural networks using information entropy.

2.1 Mobile edge computing for resource management

Current research on MEC for resource management focuses mainly on optimizing computation offloading and resource allocation. Computation offloading strategies can be roughly divided into full offloading and partial offloading. Full offloading refers to offloading the entire task on the device to the edge server for processing. Zhang et al. [27] considered a task full offloading model and proposed a fine-grained algorithm to minimize resource consumption. The main advantage of full offloading is that it can significantly reduce the computing burden and energy consumption of the device, but it requires high bandwidth and stability of the network, so partial offloading has been widely studied. Partial offloading allows a device to break down a task into multiple subtasks and select some of them to be processed locally on the device and some to be offloaded to an edge server. Zhao et al. [28] considered partial offloading to minimize execution delay and energy consumption by jointly optimizing unmanned aerial vehicle (UAV) trajectory, task allocation, and communication resources.

At present, the main optimization objectives in resource allocation can be summarized into three categories, the first category being task completion time. Chen et al. [29] introduced a novel in-flight connectivity (IFC)-oriented space-air-ground integrated networks (SAGIN) framework leveraging satellite caching and inter-satellite links to optimize content delivery, significantly enhancing IFC with reduced delays and improved efficiency. Xia et al. [30] proposed a three-tier MEC network that minimizes task processing delay by jointly optimizing UAV deployment and offloading decisions. The second category is energy consumption. Mao et al. [31] considered the computational task offloading problem to minimize the energy consumption of total users in a multiple-input multiple-output system with multiple moving users. Zhou et al. [32] proposed the joint optimizing of offloading decision, task allocation, computing frequency and UAV deployment (JODFQ) algorithm to optimize offloading decisions, task allocation, computing frequency, and UAV deployment in SAGIN, significantly reducing system energy consumption. The third category is the weighted sum of delay and energy consumption. Yu et al. [33] proposed an intelligent ultradense edge computing (I-UDEC) framework, utilizing a two-timescale deep reinforcement learning (2Ts-DRL) approach to minimize offloading delay and network resource usage. An et al. [34] leveraged the deep deterministic policy gradient (DDPG) algorithm to optimize UAV trajectories, power, task offloading ratios, and destinations in SAGIN, effectively minimizing energy consumption and delay. Zhao et al. [35] proposed a spatio-temporal attention-based PPO algorithm for LEO satellite-assisted 6G IoT networks, effectively optimizing task offloading and resource allocation to reduce delay and improve throughput. Zhao et al. [36] proposed a graph deep deterministic policy gradient algorithm leveraging relational priors to accelerate convergence and reduce complexity in predictive power allocation for wireless networks. Dai et al. [37] investigated secure resource allocation in integrated sensing and semantic communication systems, proposing optimization algorithms to maximize secure semantic efficiency under multiple eavesdroppers and imperfect channel state information. In addition, recent advances in edge learning (EL) have addressed the critical challenge of synchronizing models across geographically distributed edge servers, thereby minimizing communication overhead and improving learning convergence stability. Studies on distributed EL techniques emphasize joint optimization of learning and communication processes, highlighting the interplay between model synchronization and communication efficiency [38].

The performance goal of these studies is to optimize the transmission time or energy consumption of the mobile terminal through the user selection offloading scheme, without considering the representation capability of the task execution model, but the above studies provide the basic theoretical support for our paper.

2.2 Information entropy for evaluating the representation capability of the model

Unlike traditional research that focuses on optimizing transmission time or energy consumption, recent studies have started to focus on the representation capability of the model, particularly by evaluating the task execution models performance through information entropy. Information entropy, as a tool for measuring the amount of information and uncertainty within a model, provides crucial theoretical support for assessing the model's performance across different task execution scenarios. If the model with a better representation capability is selected to handle the task, it can capture the characteristics of the data more accurately, thus improving the accuracy of the task and helping to improve the reliability of the whole system. However, different models have different representation capabilities in different tasks, so we need to find a suitable model according to a specific task. Due to the unexplainability of the model [39], there are limited studies on model selection at present. Inspired by the information bottleneck theory in [25, 40], the authors proposed analyzing a deep neural network (DNN) on the IP. Through the analysis in [25, 26], the authors designed an IP-based framework to evaluate DNN performance in image classification tasks, facilitating the selection of neural networks that achieve higher classification accuracy. In addition, the IP theory has been applied in various areas. For example, in [41], the authors used the IP to select the optimal network structure

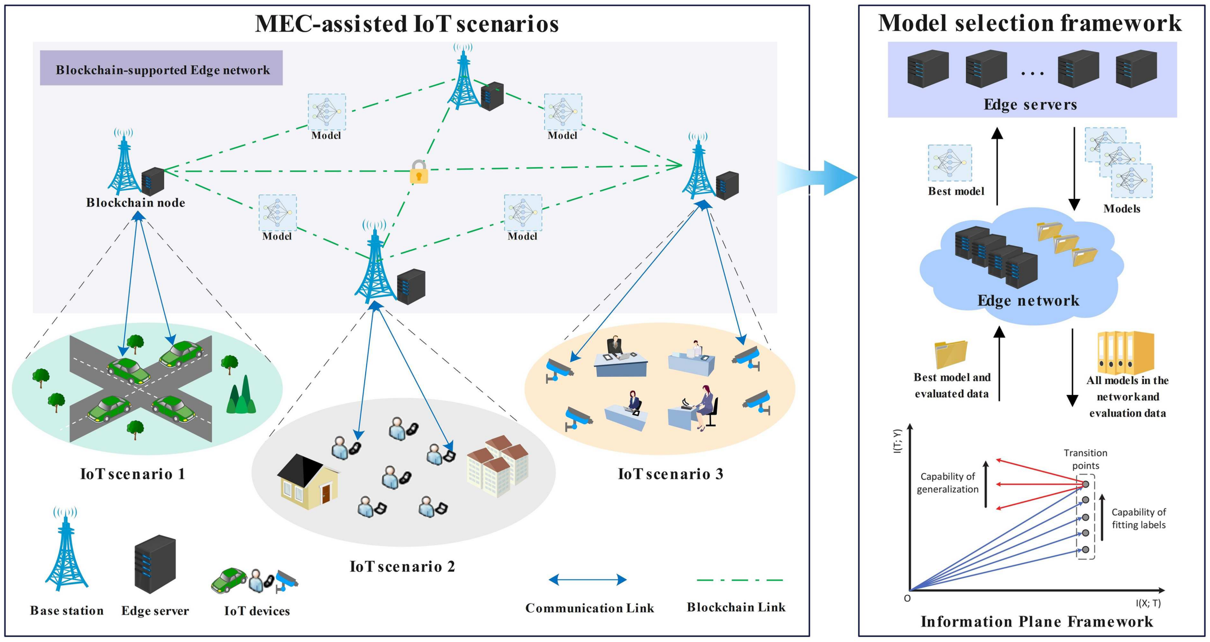
MEC-assisted IoT scenarios
Blockchain-supported Edge network
Model
Model
Model
Model
Blockchain node
IoT scenario 1
IoT scenario 2
IoT scenario 3
Base station
Edge server
IoT devices
Communication Link
Blockchain Link
Model selection framework
Edge servers
Best model
Models
Edge network
Best model and evaluated data
All models in the network and evaluation data
I(T; Y)
Transition points
Capability of generalization
Capability of fitting labels
O
I(X; T)
Information Plane Framework

Figure 1 (Color online) Illustration of the MEC-assisted IoT scenarios and the model selection framework.

for a soil image classification task. By comparing the performance of different network structures in the IP, they were able to guide model selection and optimization. In [42], a new surrogate gradient learning method is proposed based on the information bottleneck, eliminating redundant information that is not related to the target signal while retaining relevant information. However, the application of information bottleneck theory to the problem of resource management to evaluate the representation capability of the model has not been studied.

3 System model and problem formulation

In this section, the system model for the MEC-assisted IoT scenarios is presented. First, we propose the overall network architecture. Second, the system delay and energy consumption model is described in detail. Then, a framework to evaluate the representation capability of a model by an IP is described. Finally, we formulate a model selection and resource management problem.

3.1 Network architecture

As shown in Figure 1, the MEC-assisted IoT scenarios considered in this paper consist of two parts: the device side and the edge side. The edge side consists of several base stations, each with an MEC server deployed to serve tasks in the covered area, and blockchain technology is introduced between the base stations to share the information of the model in the network, facilitating the efficient scheduling of edge server resources. In this framework, the blockchain is defined as a consensus layer deployed among edge servers, used to share model information across the entire network in a decentralized and tamper-evident manner. Specifically, MEC servers on the edge side act as blockchain nodes, updating and sharing models between blocks. In terms of workflow, after IoT devices finish local training, they submit their models to the edge server. The edge server packages the model together with its IP metrics, and performs cross-verification with other edge servers. Once consensus is reached, the model and its IP metrics are shared across all edge servers in the network. In addition, the blockchain technology enables all edge servers to operate under a consistent global knowledge view, allowing them to quickly select and distribute the optimal model. At the same time, only models and capability metrics are shared on the blockchain, without uploading raw data, thus protecting the privacy of device data. Within base station coverage, the IoT devices can handle tasks using models offloaded from MEC servers or train their models to handle tasks using local data sets. The device side consists of several user devices, each in its corresponding base station area, and there are some computationally intensive applications in IoT devices. However, the energy and computational capabilities of IoT devices are limited, so it is also possible to directly use the existing trained model on the MEC server.

In MEC-assisted IoT scenarios, some user devices usually store data sets to train local models, and in this paper, we define the set of IoT devices on the user side as $\mathcal{N} = \{1, 2, 3, \ldots, N\}$, the data set stored by the nth device is D_n, and the data size is $|D_n|$. Meanwhile, we define the edge server device as $\mathcal{B} = \{1, 2, 3, \ldots, B\}$, and consider using orthogonal frequency division multiple access (OFDMA) technology to link IoT devices and edge servers. In this scenario, the IoT device uploads the model to the edge server, the edge server implements the blockchain consensus mechanism, so that the edge server has all the trained models, and then compares the representation capabilities of these trained models according to the methods provided by the IP, and saves the best models. The specific model selection framework is depicted on the right in Figure 1. In the above process, to achieve real-time interaction between the IoT device and the edge server, the delay generated in the interaction process should be reduced, and the limited computing and communication resources of the IoT device and the edge server also need to reduce the energy consumption generated in the interaction process. To sum up, the overall optimization goal is to minimize system delay and energy consumption and improve system reliability by ensuring that IoT devices can adopt a model with good presentation capability.

3.2 System delay model

In the paper, we analyze the image classification task, the system delay includes the local computation delay of IoT devices caused by the IoT device training the model itself, the transmission delay of uploading training models from IoT devices to edge servers, the transmission delay of edge servers delivering models to IoT devices, and the delay generated by the edge blockchain mechanism. Computation delay of IoT devices: the computation delay of the nth IoT device for the image classification task is defined as

$$T_n^{\text{devcomp}} = \frac{|D_n| C_n}{f_n^c}, \tag{1}$$

where C_n represents the number of CPU cycles of the nth IoT device training unit data samples, and f_n^c represents the CPU cycle of the nth IoT device.

Transmission delay. When the nth device chooses to upload the training model to the bth edge server, the uplink transmission rate from nth IoT to the bth edge server is

$$r_{n,b}^{\text{edge}}(t) = \sum_{c=1}^{C} \tau_{n,c} B_{\text{up}}^{\text{edge}} \log_2 \left(1 + \frac{p_{n,c}^{\text{edge}}(t) h_{n,c}^{\text{edge}}(t)}{N_0}\right), \tag{2}$$

where IoT devices share C sub-channels to transmit model, $\tau_{n,c}$ is number of sub-channels allocated to IoT device n in sub-channel c, $B_{\text{up}}^{\text{edge}}$ is uplink transmission bandwidth, N_0 is noise power, $p_{n,c}^{\text{edge}}(t)$ and $h_{n,c}^{\text{edge}}(t)$ is transmission power and the channel gain of IoT device n in sub-channel c, respectively. The transmission delay when nth IoT device uploads model to the bth edge server is

$$T_{n,b}^{\text{upcomm}} = \frac{|D_n^{\text{model}}|}{r_{n,b}^{\text{edge}}(t)}, \tag{3}$$

where $|D_n^{\text{model}}|$ represents the model size of IoT device n, the transmission delay of the bth edge servers delivering models to the nth IoT devices is

$$T_{b,n}^{\text{downcomm}} = \frac{|D^{\text{model}}|}{r_b^{\text{edge}}}, \tag{4}$$

where $|D^{\text{model}}|$ represents the size of the model delivered by edge servers. $r_b^{\text{edge}} = B_{\text{down}}^{\text{edge}} \log_2 (1 + \gamma_{\text{dn}})$ is the transmission rate from edge servers to IoT devices, where $B_{\text{down}}^{\text{edge}}$ is the downlink transmission bandwidth and γ_{dn} is the signal-to-noise ratio (SNR) of the downlink.

Blockchain-generated delay. The blockchain delay T^{blk} includes the model propagation delay T^{tr} between edge servers, the blockchain cross-validation delay T^{cv}, and the block generation delay T^{gen}. The block generation delay T^{gen} follows an exponential distribution with a mean of $\frac{1}{\lambda}$, and λ is the block generation rate. The model propagation delay T^{tr} between edge servers is defined as

$$T^{\text{tr}} = \sum_{b=1}^{B} \log_2 B \frac{\sum_{n=1}^{N_b} |w_{n,b}|}{r_{\text{edge}}}, \tag{5}$$

where $|w_{n,b}|$ represents the model size of the nth IoT device on the bth edge servers, the $r_{\text{edge}} = B_{\text{edge}} \log_2 (1 + \gamma_{\text{edge}})$ is the transmission rate between edge servers, B and N_b represent the number of edge servers and the number of IoT device training models included in the bth edge server, respectively. The delay of block cross-validation is calculated as

$$T^{\mathrm{cv}} = \log_2 M_p \frac{S_B}{r_{\mathrm{edge}}} + \max_b \frac{S_B f^{\mathrm{bl}}}{f^c_{b,s}}, \tag{6}$$

where f^{bl} indicates the number of CPU cycles required to process the block unit, $f^c_{b,s}$ is the number of computation resources allocated to the block unit by the bth edge servers, S_B and M_p represent the size of each block and the number of block producers, respectively. The blockchain delay is $T^{\mathrm{blk}} = T^{\mathrm{gen}} + T^{\mathrm{tr}} + T^{\mathrm{cv}}$.

3.3 System energy consumption model

In this paper, the system energy consumption includes the energy consumption generated by the computing of IoT devices, the transmission energy consumption generated by uploading models from IoT devices to edge servers, and the transmission energy consumption generated by edge servers delivering models to IoT devices.

Computational energy consumption of IoT devices. The computational energy consumption of the nth IoT device to perform the image classification task is defined as

$$E^{\mathrm{devcomp}}_n = \beta_n C_n |D_n| (f^c_n)^2, \tag{7}$$

where the size of β_n depends on the effective switching capacitance of the IoT device chip structure, the parameter is used to represent the energy efficiency per square of the CPU cycle frequency, its unit is $\mathrm{J}\cdot\mathrm{s}^2/\mathrm{cycle}^3$, f^c_n indicates the CPU cycle frequency of the nth IoT device, C_n and $|D_n|$ indicate the CPU cycle required for training data samples of the nth IoT device and the size of the data set, respectively.

Transmission energy consumption of IoT devices. The energy consumption of IoT device n transmitting model to edge server b can be expressed as

$$E^{\mathrm{iottoedge}}_{n,b} = \frac{|D^{\mathrm{model}}_n| \sum_{c=1}^{C} p^{\mathrm{edge}}_{n,c}(t)}{r^{\mathrm{edge}}_{n,b}(t) C}. \tag{8}$$

Transmission energy consumption of edge servers. The energy consumption of edge server b that delivers models to the IoT device n can be expressed as

$$E^{\mathrm{edgetoiot}}_{b,n} = \frac{|D^{\mathrm{model}}| p^{\mathrm{edge}}_b}{r^{\mathrm{edge}}_b}. \tag{9}$$

3.4 Evaluating the representation capability of the model

In this paper, we primarily consider DNNs, as DNNs have strong feature extraction and representational capabilities, making them well-suited for real-time data processing tasks on edge devices and widely used in MEC environments. However, DNNs have complex internal structures and their internal workings are often difficult to interpret, leading to a lack of user trust and challenges in error troubleshooting. Additionally, there needs to be a clear standard for evaluating the generalization capability of DNNs. Therefore, we introduce a method for evaluating DNNs. We consider the image classification task in supervised learning and use mutual information to quantify the information contained in the trained model, thereby evaluating the representation capability of the model structure. To elaborate, the image classification task feeds the dataset X with the label Y to the K-layer DNN for mapping to obtain the representation variable T and the predicted output $\hat{Y}$. Consider the K-layer DNN structure as a Markov chain of continuous internal representations of the input X, formalized as $Y \rightarrow X \rightarrow T_1 \rightarrow \cdots \rightarrow T_i \rightarrow \cdots \rightarrow T_k \rightarrow \hat{Y}$. Where T_i represents the multivariate variable of the ith hidden layer and is defined by the encoder and decoder, $P(T|X)$ and $P(Y|T)$, respectively [25]. According to two properties of mutual information, the invariance of reversible transformation and the data processing inequality (DPI) can be obtained

$$I(X;Y) \geqslant I(T_1;Y) \geqslant I(T_2;Y) \geqslant \cdots \geqslant I(T_k;Y) \geqslant I(\hat{Y};Y).$$

We aim to evaluate the representation capability of the DNN, and the last hidden layer reveals the relationship among the input X, label Y and model output T, so only the last hidden layer needs to be quantified. Further, the Markov chain formed by the DNN is simplified to $Y \rightarrow X \rightarrow T \rightarrow \hat{Y}$. Meanwhile, the number of neurons in the last hidden layer of the DNN for the image classification task is the same as the number of classes in the dataset. Therefore, we compress the C-dimensional real vector z into the C-dimensional vector $\sigma(z)$, which is compressed into the range $[0,1]$ and whose the real values sum to 1 [26]. Here, C is the number of data set classes. Finally, the C-dimensional vector $\sigma(z)$ is divided into C equal intervals between 0 and 1 to obtain the model output T.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{c=1}^{C} e^{z_c}},\ j = 1,\ldots,C. \tag{10}$$

Moreover, we use the concept of minimal sufficient statistics to explain the optimal representation features for the image classification task. The sufficient statistics $S(X)$ is a mapping of input X, capturing all the information of X on Y, formalized as $I(S(X);Y) = I(X;Y)$ [25]. The image classification task aims to learn the minimum sufficient statistic $T(X)$ of the input X on the label Y. Formulating the task as a Markov chain $Y \rightarrow X \rightarrow S(X) \rightarrow T(X)$, thus transforming it into a constrained optimization problem based on the DPI property of mutual information.

$$T(X) = \mathrm{argmin}_{S(X):I(S(X);Y)=I(X;Y)} I(S(X);X). \tag{11}$$

As just mentioned, the concept of minimum sufficient statistics considers that the goal of DNN learning is to make $I(S(X);X)$ as small as possible to obtain an efficient representation vector. At the same time, ensuring that the values of $I(S(X);Y)$ and $I(X;Y)$ are the same, which means that the information on Y will not be lost. Namely, the learning process of the DNN is a trade-off between $I(S(X);X)$ and $I(S(X);Y)$. The information bottleneck (IB) theory provides a computational framework for finding approximate minimum sufficient statistics [40]. The IB theory considers an efficient representation as a trade-off between the compression of X and the predictive capability of Y. We set $x \in X$ as the input data, and the representation of x is defined by the probability mapping $p(t|x)$. Meanwhile, $t \in T$ is the output of the model and is defined as the compressed representation of x. At this point, the Markov chain of the model is $Y \rightarrow X \rightarrow T$. The IB trade-off transforms solving the minimum sufficient statistics into the following optimization problem:

$$\min_{p(t|x),p(t),p(y|t)} \{I(X;T) - \beta I(T;Y)\},$$

where the Lagrangian multiplier β determines the level of relevant information captured by the vector t; the probability distributions $p(t|x)$, $p(t)$ and $p(y|t)$ are independent of each other, and are defined by the following equations, respectively.

$$\begin{cases} p(t|x) = \dfrac{p(t)}{G(x;\beta)} \exp(-\beta D_{KL}[p(y|x)||p(y|t)]), \\ p(t) = \sum_x p(t|x)p(x), \\ p(y|t) = \sum_x p(y|x)p(x|t), \end{cases} \tag{12}$$

where $G(x;\beta)$ is the normalization function, $[p(y|x)||p(y|t)]$ is the Kullback-Liebler divergence of the distributions $p(y|x)$ and $p(y|t)$. Given the parameters β, the joint probability distributions $p(t|x)$, $p(t)$ and $p(y|t)$, the optimal $I(X;T)$ and $I(T;Y)$ are obtained by minimizing the expressions. The mutual information values $I(X;T)$ and $I(T;Y)$ are both related to the representation capability of the DNN. Therefore, we use $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ to measure the representation capability of the DNN training process. Here, the smaller the value of $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$, the stronger the model representation capability and the higher the model accuracy [26].

Using the above conclusions, we can analyze each model with the IP and obtain the slope from the transition point to the convergence point in the IP. The details of the analysis are described in Subsection 3.5. We define the model representation capability of the nth IoT device as $L_n^{\mathrm{dev}} = |\frac{\Delta I_n(T;Y)}{\Delta I_n(X;T)}|$. In addition, when IoT devices upload models to edge servers, edge servers obtain models of all IoT devices through the blockchain consensus mechanism. From this, the representation capability of all models is calculated and the best model is saved. Thereby, the model representation capability saved by the edge server is $L_b^{\mathrm{edge}} = |\frac{\Delta I_b(T;Y)}{\Delta I_b(X;T)}|$.

3.5 Problem formulation

In this paper, we denote the model training strategy for the nth IoT device as a binary variable $o_{n,t} \in [0,1]$, where $o_{n,t} = 1$ means that the nth IoT device uses the local dataset trains model locally to process the image classification task, and $o_{n,t} = 0$ means that the nth IoT device uses the model delivered by the edge server to process the image classification task. Specifically, when the model training strategy of the nth IoT device $o_{n,t} = 1$, the IoT device uses the local dataset to train the model processing tasks and produces computation delay and energy consumption. Then, the IoT device uploads the trained model to edge servers, while producing transmission delay and energy consumption. Further, the trained models of all IoT devices are uploaded to edge servers. Meanwhile, edge servers perform the blockchain consensus mechanism, giving edge servers all the trained models. Based on this, compare the representation capability of the trained models and save the best model. When the model training strategy of the nth IoT device $o_{n,t} = 0$, the IoT device uses the model delivered by edge servers to process the task while producing transmission delay and energy consumption. However, IoT devices no longer upload models to the edge servers. After that, when $o_{n,t} = 1$, the training models from IoT devices are uploaded to edge servers. The edge servers then perform the blockchain consensus mechanism, compare the representation capability of all the models, and save the best model.

Therefore, the computation delay and transmission delay generated by the system due to selecting a model to process the image classification task of the nth IoT device is defined as

$$T_n = o_{n,t} \cdot (T_n^{\rm devcomp} + T_{n,b}^{\rm upcomm}) + (1 - o_{n,t}) \cdot T_{b,n}^{\rm downcomm}. \tag{13}$$

Based on the model training strategy $o_{n,t}$ of the nth IoT device, the total energy consumption for selecting a model to process the nth IoT device tasks is defined as

$$E_n = o_{n,t} \cdot (E_n^{\rm devcomp} + E_{n,b}^{\rm iottoedge}) + (1 - o_{n,t}) \cdot E_{b,n}^{\rm edgeioiot}. \tag{14}$$

The model representation capability obtained by processing IoT device n is defined as

$$L_n = o_{n,t} \cdot L_n^{\rm dev} + (1 - o_{n,t}) \cdot L_b^{\rm edge}. \tag{15}$$

To sum up, in this paper, the optimization objective is to maximize the model representation capability, and minimize the system delay and energy consumption. The optimization objective is defined as

$$\Phi_{\rm target}^{\rm ef} = \mu_L L^{\rm rep} + \mu_T T^{\rm cost} + \mu_E E^{\rm cost} = \mu_L \frac{\sum_{n=1}^N L_n}{N} - \mu_T \frac{\sum_{n=1}^N T_n + T^{\rm blk}}{N} - \mu_E \frac{\sum_{n=1}^N E_n}{N}, \tag{16}$$

where μ_L, μ_T and μ_E are weight coefficients. Considering the constraints in the system, we formulate the optimization problem as follows:

$$\max_{\mu,f,B,D} \Phi_{\rm target}^{\rm ef} = \mu_L \frac{\sum_{n=1}^N L_n}{N} - \mu_T \frac{\sum_{n=1}^N T_n + T^{\rm blk}}{N} - \mu_E \frac{\sum_{n=1}^N E_n}{N}, \tag{17}$$

$$\begin{aligned} \text{s.t.} \quad & C1: \mu_L + \mu_T + \mu_E = 1,\ \mu_L, \mu_T, \mu_E \in [0,1], \\ & C2: \textstyle\sum_{n=1}^N \tau_{n,c} \leqslant 1, c \in C, \\ & C3: T^{\rm cost} \leqslant T^{\rm th}, \\ & C4: E^{\rm cost} \leqslant E^{\rm th}, \\ & C5: L^{\rm rep} \geqslant L^{\rm th}, \end{aligned}$$

where constraint $C1$ means that for the scale factor μ_L, μ_T and μ_E, weight coefficients can be assigned to each performance indicator according to the IoT scenario. Constraint $C2$ guarantees that each sub-channel can only be allocated to at most one IoT device. Constraints $C3$–$C5$ ensure that system delay, energy consumption, and the model representation capability are better than the specified threshold.

Due to the limitations of various conditions, it is necessary to determine the bandwidth allocation, decision parameters and weight coefficients of each time slot. Here, the appeal problem is an NP-hard optimization problem that is difficult to solve using classical convex optimization algorithms. DRL is a powerful tool for solving optimization problems with long-term goals and constraints. The main idea behind DRL is to translate the optimization problem into an MDP formula, where the goal of MDP is to maximize long term expected rewards. In addition, DRL has the ability to achieve online optimization, which is suitable for solving automatic decision problems. Hence, we adopt a DRL algorithm to solve this problem.

4 Proposed algorithm

In this section, we design a DRL-based model selection and resource management scheme to allow users to choose an appropriate model to handle tasks in the network and minimize the delay and energy consumption of the system. In this scheme, we propose an IP framework that utilizes mutual information to effectively evaluate the representation

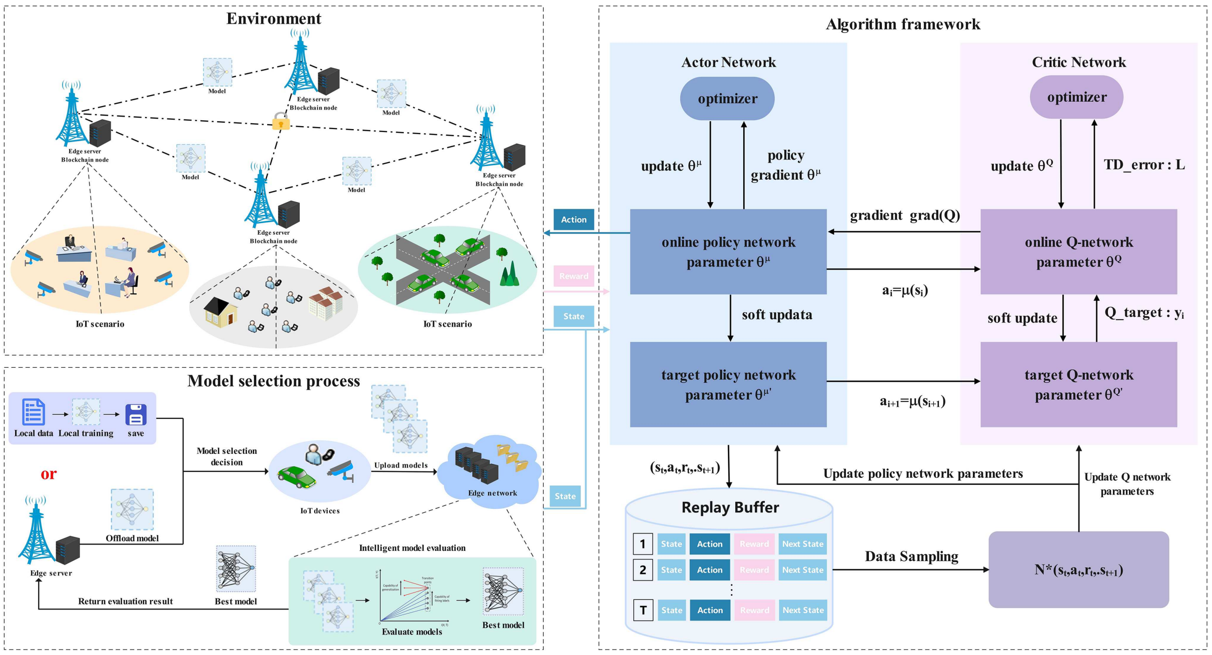

Figure 2 (Color online) Illustration of model selection and resource management scheme based on DRL.

capability of models, and the IP provides more information than the loss curve, which helps us select intelligent models that improve the classification accuracy more effectively. Specifically, we first use the IP framework and blockchain technology to introduce model evaluation metrics and identify the best-performing model in the network. Through the IP framework, the agent can better evaluate the quality of its behavior, thereby making better decisions and improving the performance of the system. This scheme is elaborated in the rest of this section, and its algorithm framework is shown in Figure 2.

4.1 Markov decision process construction

To solve the complex problems in the scenarios presented in this paper, a DRL algorithm is designed, where each MEC server acts as an agent to learn the resource management scheme and solve the corresponding problem formulated. Specifically, we first re-model the resource management problem targeting IoT devices and edge services as an MDP. The characteristic of this learning framework can be defined as a tuple (N, S, A, P, R), where N represents a set of agents, S represents the possible states, A is the set of actions, P represents the probability of transition from one state to another, and R is the reward function about $s \in S$ and $a \in A$. By modeling the computation offloading process as an MDP system, a balanced state can be found to achieve high-efficiency system performance. The states, actions, and rewards of the system are defined as follows:

State (S). For every IoT device $n \in N$, the state of the time slot t can be defined as $s_{t,n} = \{D_{t,n}, H_{t,n}, E_b, F_n, W\}$, $D_{t,n}$ is the size of the model of the IoT device n in the time slot t, $H_{t,n}$ is the gain in channels of the IoT device n. Here, we quantize H_t into L levels, denoted by $H_t = \{h_1, h_2, \ldots, h_L\}$. E_b is the number of user devices served on the edge server b, F_n are the computing resources of the IoT device n, W is the bandwidth.

Action (A). The state of time slot t can be defined as $a_{t,n} = \{B_{t,n}, o_{t,n}\}$, $B_{t,n}$ is the bandwidth allocation rate, we denote the model training strategy for the nth IoT device as a binary variable $o_{t,n} \in [0, 1]$, where $o_{t,n} = 1$ means that the nth IoT device uses the local data set to train the model locally to process the image classification task, and $o_{t,n} = 0$ means that the nth IoT device uses model delivered by the edge server to process the image classification task.

Reward (R). After executing an action a_t, the agent transitions to the next state $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ and obtains an immediate reward r_t. To maximize the efficiency of the system, the reward function is formulated as

$$r_t = \mu_L L^{\text{rep}} + \mu_T T^{\text{cost}} + \mu_E E^{\text{cost}}. \tag{18}$$

In our algorithm, each agent can make decisions to determine an optimal resource allocation strategy based on the states and rewards gained in DRL. In addition, the blockchain can record the state and knowledge learned by an agent to achieve a consensus process between other agents.

Table 1 Mutual information metrics for transition points and convergence points.

Index	Model	Transition point			Convergence point		
		$I(T;Y)$	$I(X;T)$	Epochs	$I(T;Y)$	$I(X;T)$	Epochs
NN0	DNN-2(ReLU)	2.739	3.671	3	3.116	3.474	90
NN1	DNN-3(ReLU)	2.737	3.664	3	3.113	3.445	77
NN2	DNN-4(ReLU)	2.746	3.671	3	3.117	3.446	60
NN3	DNN-5(ReLU)	2.712	3.657	3	3.121	3.425	67
NN4	DNN-4(tanh)	2.795	3.662	3	3.086	3.463	52
NN5	CNN-2	2.779	3.666	1	3.215	3.386	36
NN6	CNN-3	2.725	3.654	1	3.233	3.380	29
NN7	CNN-4	2.818	3.672	2	3.233	3.370	38

4.2 Model selection mechanism with IP

In our proposed framework, we need to use model representation capability as knowledge to optimize decision-making and complete the model selection mechanism, so we need to use the IP framework to obtain information entropy to represent model representation capability. In this section, we will introduce the model selection mechanism based on the IP framework. IP framework demonstrates that the representation capability of intelligent models can be effectively assessed through mutual information, thereby aiding in the selection of models that significantly enhance classification accuracy. According to the above, we use the IP to evaluate the model selected by the device each time, and make $I(X;T)$, $I(T;Y)$ and training epochs for each model at transition points and convergence points, where X and Y represent validation input and validation label respectively, and draw the IP of the model. Here, we plot the IP of eight neural networks on the MNIST dataset, as shown in Figure 3. The transition point in the information plane marks the change from the fitting phase, where stochastic gradient descent (SGD) rapidly increases $I(T;Y)$ and $I(X;T)$ by capturing label-relevant information, to the compression phase, where SGD noise and implicit regularization reduce $I(X;T)$ while $I(T;Y)$ plateaus, discarding input details irrelevant to the labels. The convergence point occurs when both metrics stabilize, indicating no further meaningful representation changes. Experiments on MNIST show that CNNs reach the transition earlier and converge in fewer epochs than DNNs. A smaller post-transition slope $\frac{\Delta I(T;Y)}{\Delta I(X;T)}$ indicates stronger representation capability and better generalization, and the gap between transition and convergence reflects the length of the compression process.

Table 1 records the transition points, convergence points and training epochs of each model. In the IP, the height of the transition point represents the model's capability to fit labels, and the slope after the transition point can represent the model's generalization capability. In terms of prediction accuracy, the longer the training time of the network, the more accurate the prediction judgment of the model can be, because the future mutual information path may have a greater slope change. Here, we control the training time of the network model, to more intuitively reflect the difference in the performance of different models in the IP, that is, the difference in the slope after the transition point. In summary, the IP framework provides a powerful tool to evaluate and compare the representation capability and generalization performance of different models, facilitating optimal model selection for improved accuracy.

4.3 PPO-based model selection and resource management algorithm

According to the above analysis, we introduce here the PPO-based model selection and resource management algorithm. In the communication scenario that we consider, each IoT device acts as an agent, and each agent interacts with the environment and then takes actions accordingly. During the learning process, the agent constantly updates the strategy according to the rules of the PPO algorithm until it learns the optimal strategy. The scheme proposed in this paper considers how to select the model intelligently and optimize the resource allocation problem. When the target network is updated, the IoT devices will upload the local model and the edge servers will update the edge model to preserve the model with good representational ability. As shown in the right part of Figure 2, there is a global network in our algorithm, including the actor network and the critic network, and there are multiple agents with the same network structure as the global network, that is, each agent consists of two parts. One is the actor network, which interacts with the environment to collect data and approximates the policy function with the parameter θ. The other is the critic network, which is responsible for judging the good and bad behavior of the actor network and estimating its value. Each agent interacts independently with the environment without interfering with the other, and after interacting with the environment, each agent computes the gradient of the neural network loss function and updates the shared parameters of the global network independently. Every once in

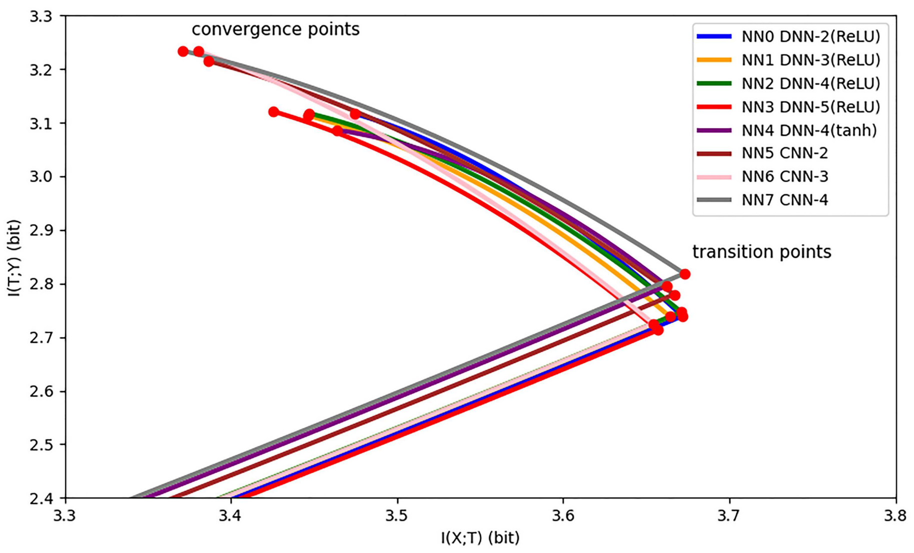

Figure 3 (Color online) The variation of the IP curves of the eight proposed neural network models.

a while, each agent updates the parameters of its own neural network using the parameters of the global network.

To improve the training speed and data utilization, in the PPO algorithm used in this paper, each agent uses the policy $\pi_{\theta_{\text{old}}}$ to obtain experience data (s_t, a_t, r_t, s_{t+1}) from the environment, including each agent's channel gain, bandwidth allocation rate, model selection strategy. To be specific, $s_t = \{s_1, \ldots, s_N\}$, $a_t = \{a_1, \ldots, a_N\}$, $r_t = \{r_1, \ldots, r_N\}$. The experience data for each moment contains data for N agents. Actor network with parameter θ uses experience data (s_t, a_t, r_t, s_{t+1}) to update parameter θ. The update of the parameter θ using data collected by $\pi_{\theta_{\text{old}}}$ is called importance sampling. We can obtain the objective function after importance sampling as

$$\begin{aligned} J(\theta) &= E_t[\hat{A}_t \log \pi_\theta(a_t|s_t)] \\ &= E_t\left[\hat{A}_t \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\right] \\ &= E_t[\hat{A}_t r_t(\theta)], \end{aligned} \tag{19}$$

where $r_t(\theta)$ is the importance weight and $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} = \mu_L L^{\text{rep}} + \mu_T T^{\text{cost}} + \mu_E E^{\text{cost}}$. $r_t(\theta)$ represents a combination of model representation capability, system delay, and energy consumption in the time slot t. The advantage function can be estimated by the generalized advantage estimation (GAE) [43]. The advantage function can estimate the return of trajectory $\Phi^{\text{ef}}_{\text{target}}$.

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{20}$$

where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, $V_\phi(s_t)$ is the state value at time slot t and $(\gamma\lambda)$ is the discount factor.

To ensure that the difference between policy π_θ and $\pi_{\theta_{\text{old}}}$ is small, the PPO algorithm adopts the clipping operation. We can obtain the objective function of the actor network after clipping as

$$J(\theta) = E_t[\hat{A}_t \min(r_t(\theta), \hat{A}_t \text{clip}(r_t(\theta), 1+\varepsilon, 1-\varepsilon))], \tag{21}$$

where ε is a hyperparameter and $\text{clip}(r_t(\theta), 1+\varepsilon, 1-\varepsilon)$ is equivalent to

$$r_t(\theta) = \begin{cases} 1-\varepsilon, & r_t(\theta) < 1-\varepsilon, \\ r_t(\theta), & 1-\varepsilon \leqslant r_t(\theta) < 1+\varepsilon, \\ 1+\varepsilon, & r_t(\theta) \geqslant 1+\varepsilon. \end{cases} \tag{22}$$

Also, we can obtain the objective function of the critic network by state value $V_\phi(s_t)$ and cumulative discounted return

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}, \tag{23}$$

where T is the total number of steps in an episode. Finally, the objective function of the critic network is denoted by

$$J(\phi) = E_t[(V_\phi(s_t) - R_t)^2]. \tag{24}$$

The reward function is designed to make an appropriate trade-off between model representation capabilities, system delay, and energy consumption.

PPO-based model selection and resource management algorithm can adaptively maximize the weighted sum of model representation capability, delay and energy consumption of the system to achieve reasonable resource allocation. The specific algorithm is described in Algorithm 1.

Algorithm 1 PPO-based model selection and resource management algorithm.

```
Require: Set of IoT devices on the user side $\mathcal{N} = \{1, 2, 3, \ldots, N\}$, stored dataset $D = \{D_1, \ldots, D_n\}$, set of edge server $\mathcal{B} = \{1, 2, 3, \ldots, B\}$.
 1: Initialize actor, critic network parameters, initial model on edge server, blockchain model;
 2: for episode = 1 to MAX_EPISODE do
 3:     for step = 1 to MAX_STEP do
 4:         Edge server $b$ delivers the model to user $n$;
 5:         System calculates transmission delay, according to (1)–(6), and calculates energy consumption, according to (7)–(9), and calculates
            the model representation capability according to (10)–(12);
 6:         User transmits the model to the edge server, and each agent learning knowledge;
 7:         if step > 1 then
 8:             Network extracts the aggregated knowledge from the previous STEP;
 9:             Edge server $b$ begins a transaction to blockchain;
10:             Edge server $b$ collects all transactions and broadcasts knowledge of extracted to other BSs for verification;
11:             Edge server $b$ aggregate all knowledge, and save the model with the best representation;
12:             Edge servers distribute knowledge to users;
13:             Users combine the knowledge and make decisions based on knowledge;
14:         end if
15:     end for
16: end for
```

5 Performance evaluation

In this section, the simulation results are shown to validate the proposed PPO-based model selection and resource management scheme.

5.1 Simulation setup

The experiments were conducted in an Ubuntu operating system (CPU AMD Ryzen 5 4600H, memory 16 GB, GPU NVIDIA GeForce GTX 1650 Ti, which contains 8-GB graphics memory). Here, we built the MEC-assisted IoT scenarios consisting of 4 edge servers and 16 IoT devices evenly distributed across the coverage of 4 edge servers. At the same time, we quantified and divided the channel gain h_i into 10 values as $[0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3.0]$ [44]. The key experimental parameters are shown in Table 2. To evaluate the effectiveness of the proposed PPO-based model selection and resource management scheme, we also provide six other comparison schemes. The parameters of these six comparison schemes are consistent with those of the proposed scheme, for example, the total bandwidth is set to 25 MHz and the number of training epochs is set to 1200. In addition, the channel gain transfer probability in the experiment is set according to [45]. In this paper, the data set used in the experiment is the MNIST data set, which is a relatively simple data set for handwritten digit image recognition. It has 10 labels and contains 60000 training sets and 10000 test sets. In the simulation, we used Tensorflow to build the network architecture, and extract knowledge to train the neural network through the data uploaded by users, such as the bandwidth assigned by users and the decisions made by users. At the same time, each IoT device is randomly assigned a certain amount of data set to train the local neural network model.

5.2 Simulation results and discussion

We propose six schemes to evaluate with the proposed PPO-based model selection and resource management scheme (Joint_IPOR_PPO scheme), which are as follows.

(1) PPO-based offloading and resource allocation scheme (OR_PPO_joint scheme). In this scheme, the user makes offloading decisions and allocates resources for transmission and computation with the PPO algorithm. This scheme does not take into account the model representation capabilities.

Table 2   Key simulation parameters.

Parameter	Value
Total bandwidth for communication	25 MHz
The data size of each user	[2–147] MB
Transmit power	2 W
Required CPU cycles of training data sample	8.2 G cycle/s
The AWGN power	−114 dBm
SINR threshold	20 dB
Block generation rate	5
Learning rate of actor	0.001
Learning rate of critic	0.002
Discount factor γ	0.99
Probability of random exploration	0.1
Target updating weight	0.8
Batch size of experience replay	128
Number of episodes	1200

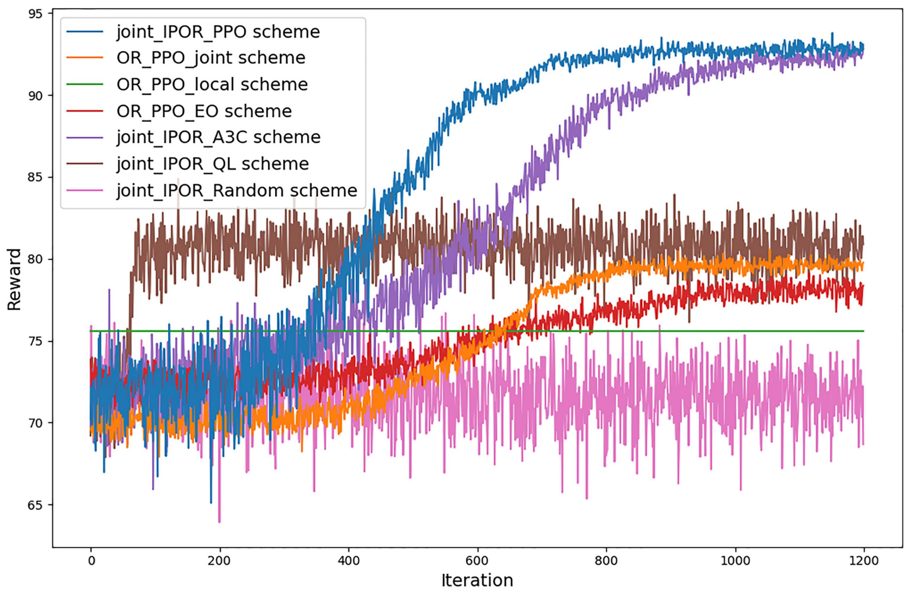

Figure 4   (Color online) System average reward of different algorithms.

(2) PPO-based offloading and resource allocation scheme only process locally (OR_PPO_local scheme). In this scheme, edge offloading is not considered. The tasks are just executed with local devices. This scheme does not take into account the model representation capabilities.

(3) PPO-based offloading and resource allocation scheme only with edge offloading (OR_PPO_EO scheme). In this scheme, the process locally is not considered. The tasks are just executed with edge servers. This scheme does not take into account the model representation capabilities.

(4) A3C-based model selection and resource management scheme (Joint_IPOR_A3C scheme). To show that the algorithm we used is more advantageous, this scheme considers the A3C algorithm to solve the resource allocation problem in model selection.

(5) QL-based model selection and resource management scheme (Joint_IPOR_QL scheme). This scheme uses a Q-learning algorithm to solve the resource allocation problem in model selection.

(6) Random-based model selection and resource management scheme (Joint_IPOR_Random scheme). The learning agent adopts a random way to update the policy.

Figure 4 compares the convergence of the proposed Joint_IPOR_PPO scheme with several baselines in terms of reward over iterations. The Joint_IPOR_PPO scheme achieves both the fastest convergence and the highest final reward. Its reward rises sharply after about 200 iterations, reaches about 90 by iteration 400, and converges around 94 at iteration 600 with minimal fluctuations, showing high efficiency and robustness. In contrast, the Joint_IPOR_A3C scheme converges more slowly, reaching about 90 only after 700–800 iterations and stabilizing

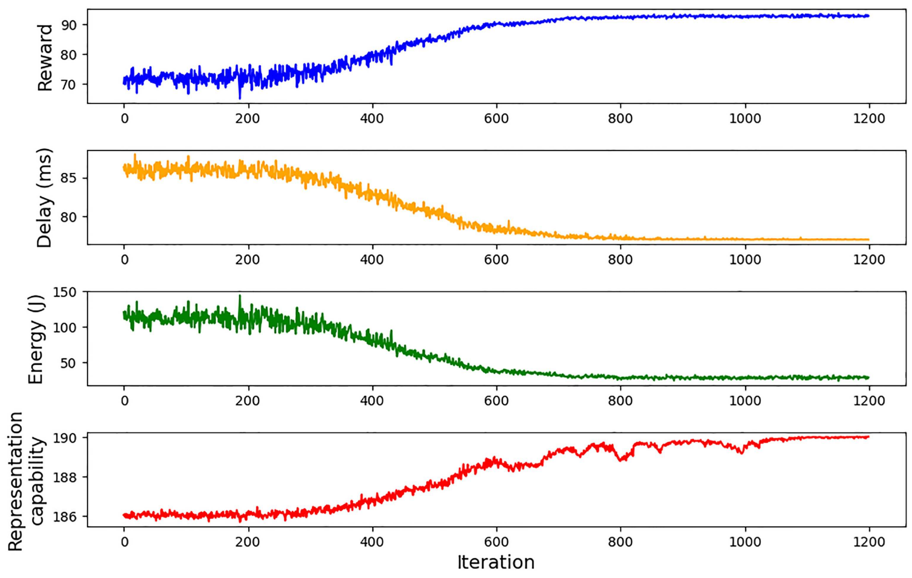

Figure 5 (Color online) The experimental results of PPO-based model selection and resource management scheme.

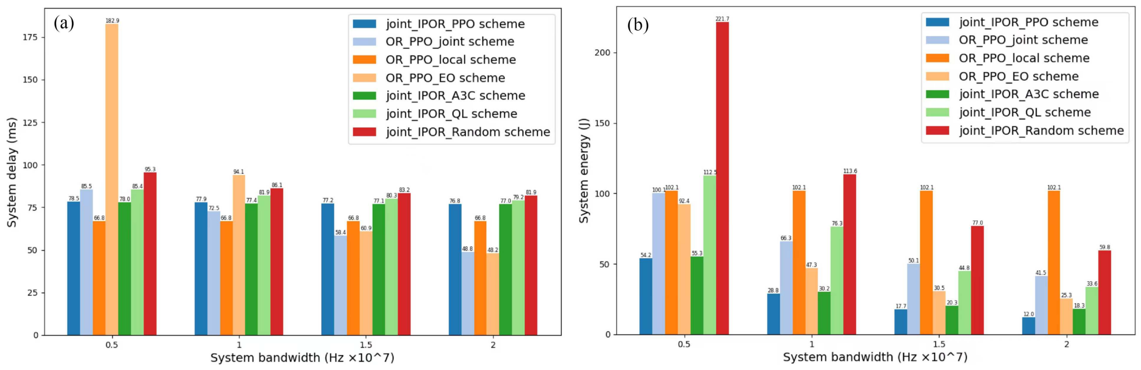

Figure 6 (Color online) Performance at different system bandwidths, including (a) system delay and (b) system energy.

around iteration 900, requiring about 50% more training steps despite a similar final reward. Other PPO-based baselines improve gradually after iteration 400, plateauing only at 900–1000 iterations, highlighting the benefit of the proposed Joint_IPOR_PPO scheme. The OR_PPO_local scheme shows almost no gain, remaining near 76, while the Joint_IPOR_Random scheme performs worst, confirming the ineffectiveness of random actions. According to the above analysis, it can be proved that the Joint_IPOR_PPO scheme demonstrates superior decision-making capabilities in comparison to other approaches.

As illustrated in Figure 5, the proposed Joint_IPOR_PPO scheme exhibits a positive correlation between the number of iterations and both the system reward and model representation capabilities, demonstrating an inverse relationship with system delay and energy consumption. In addition, the convergence analysis reveals that the proposed scheme achieves stable optimization, effectively balancing the multiple objectives of maximizing model representation capacity while minimizing system delay and energy consumption.

Subsequently, we investigate the impact of varying bandwidth conditions on system delay and energy consumption, as depicted in Figure 6. The experimental results demonstrate that delay and energy consumption decrease as the system bandwidth increases across all evaluated schemes. Notably, the OR_PPO_EO scheme exhibits a significant reduction in delay and energy consumption, primarily due to its complete reliance on uploading tasks to the edge server for processing. This design makes the scheme particularly sensitive to uplink bandwidth variations, so it has low delay and energy consumption under high bandwidth conditions. In contrast, the OR_PPO_local scheme maintains consistent performance metrics regardless of bandwidth fluctuations, as it processes tasks locally without requiring uplink transmission. Comparative analysis of the four IP framework-based schemes indicates that the proposed Joint_IPOR_PPO scheme achieves optimal performance, maintaining the lowest delay and energy consumption across all bandwidth conditions. In addition, comparative analysis reveals that while our scheme exhibits marginally higher delay than offloading approaches (OR_PPO_joint scheme, OR_PPO_local scheme, and

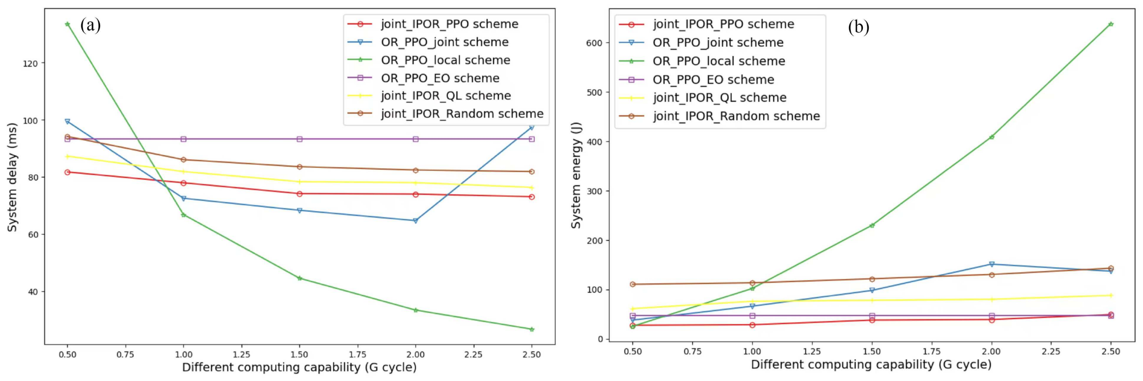

Figure 7 (Color online) The performance of PPO-based model selection and resource management scheme under the different computing capabilities, which include (a) system delay and (b) system energy.

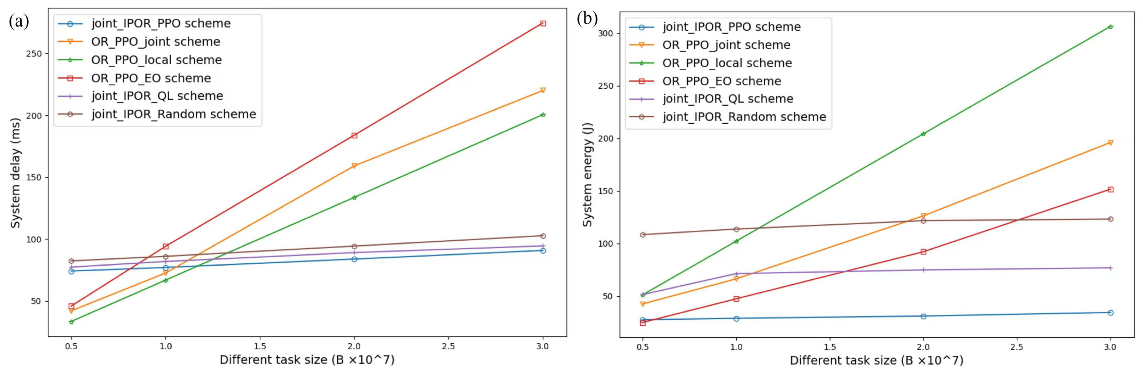

Figure 8 (Color online) The performance of the PPO-based model selection and resource management scheme under the different task sizes, which include (a) system delay and (b) system energy.

OR_PPO_EO scheme) with increasing bandwidth, it achieves significantly higher system rewards and lower energy consumption, particularly under limited bandwidth conditions. This performance advantage underscores the effectiveness of our proposed solution in balancing multiple objectives of the system.

We also evaluate the impact of varying computing capabilities on system delay and energy consumption, as shown in Figure 7. The experimental results reveal that, as a general trend, delay is negatively correlated with computing capability, while energy consumption is positively correlated with computing capability across all evaluated schemes. Notably, the OR_PPO_local scheme is particularly sensitive to changes in computing capability. Although this method achieves low delay under high computing capability conditions, it simultaneously incurs high energy consumption, leading to suboptimal system reward. On the other hand, the OR_PPO_EO scheme, which relies entirely on offloading tasks to edge servers for processing, shows a delay and energy consumption that remain unaffected by computing capability, resulting in constant values. Specifically, it can be observed that the curve of the OR_PPO_joint scheme exhibits a sudden change, which is attributed to the system's objective of identifying actions with higher reward, rather than solely focusing on minimizing delay or energy consumption. Comparative analysis with the remaining schemes reveals that the Joint_IPOR_PPO scheme reduces delay by 2%–25% and energy consumption by 1%–65%, consistently maintaining stable performance across all computing capability conditions.

Furthermore, we conduct a comparative analysis of system delay and energy consumption under varying task sizes, as illustrated in Figure 8. The experimental results demonstrate that the Joint_IPOR_PPO scheme maintains superior performance metrics compared to traditional approaches, with a notable 54% to 67% reduction in system delay and a 77% to 89% reduction in energy consumption in large-scale task scenarios. This performance advantage is due to the Joint_IPOR_PPO scheme's local task processing capability and efficient model transmission mechanism, which effectively mitigates the impact of task size variations. In contrast, traditional schemes exhibit degraded performance with increasing task sizes due to their requirement for complete task uploads to edge servers. This design limitation results in suboptimal system rewards, especially when handling large transmission tasks. Therefore, the proposed scheme's ability to maintain stable performance across different task scales highlights its robustness and efficiency in MEC-assisted IoT scenarios.

As shown in Figure 9, we examine energy consumption and delay as the number of IoT devices increases. The

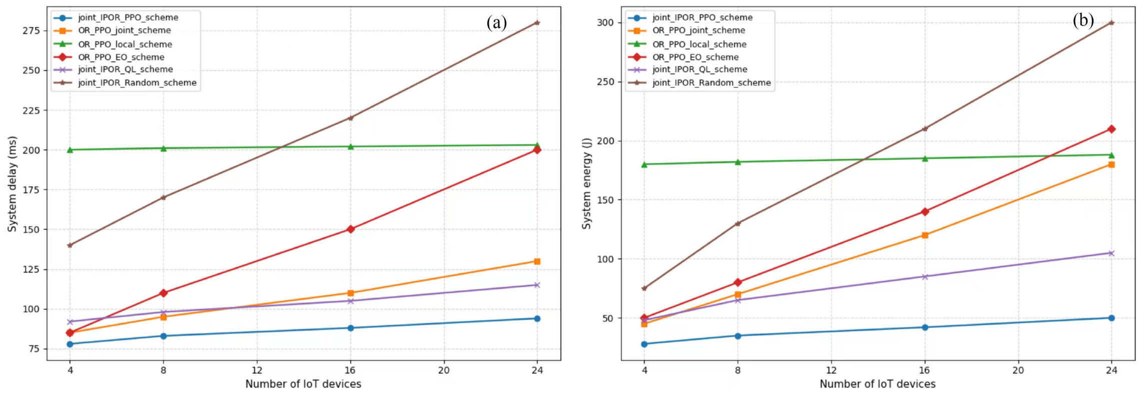

Figure 9 (Color online) The performance of PPO-based model selection and resource management scheme under the different numbers of IoT devices, which include (a) system delay and (b) system energy.

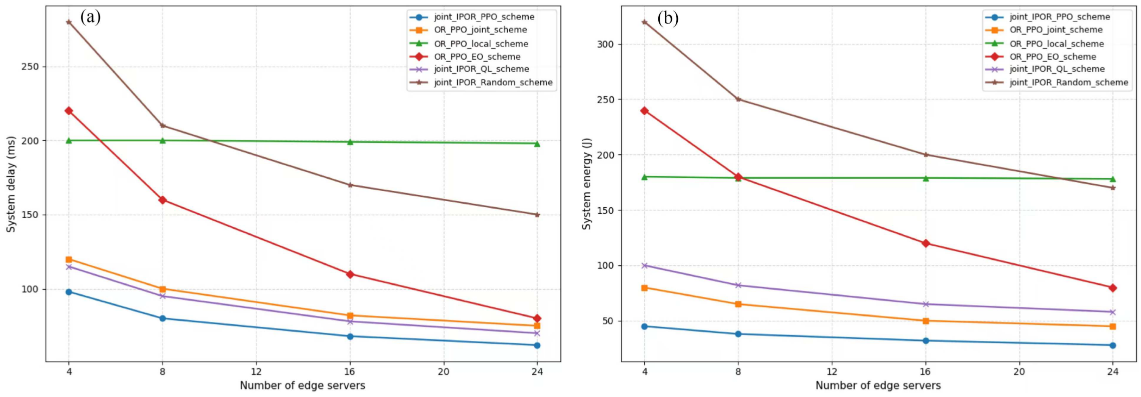

Figure 10 (Color online) The performance of PPO-based model selection and resource management scheme under the different numbers of edge servers, which include (a) system delay and (b) system energy.

proposed Joint_IPOR_PPO scheme maintains the lowest levels and smallest growth rates throughout, achieving about 50 J energy and 95 ms delay at 24 devices, demonstrating strong scalability. The OR_PPO_joint scheme, without IP knowledge, shows roughly linear increases, revealing suboptimal allocation. The OR_PPO_EO scheme grows more sharply due to aggravated uplink contention and queuing, which amplify transmission delay. The OR_PPO_local scheme is largely unaffected by scale but keeps consistently high energy and delay, limited by local processing capability. The Joint_IPOR_Random scheme performs worst, with almost linear degradation, indicating that lacking effective learning and knowledge sharing makes it unable to cope with congestion. These results confirm that the proposed Joint_IPOR_PPO scheme can effectively suppress energy and delay growth in dense networks.

As shown in Figure 10, system energy and delay generally decrease as more edge servers are deployed. The proposed Joint_IPOR_PPO scheme consistently achieves the lowest values and the smoothest decline. At 24 servers, it reaches about 32 J energy and 65 ms delay, corresponding to reductions of roughly 30% and 34% from the initial values. This advantage comes from IP-based knowledge-driven model selection with resource allocation implemented by the PPO algorithm. The OR_PPO_joint scheme drops at a moderate rate but remains inferior due to not introducing the IP framework. The OR_PPO_EO scheme shows the steepest improvement, with energy decreasing from about 240 to 85 J and delay from about 220 to 80 ms as the server count grows. This is because the scheme relies entirely on offloading tasks to edge servers for processing. The OR_PPO_local scheme is almost unaffected by server scale, indicating that performance is limited by local processing bottlenecks. The Joint_IPOR_QL scheme outperforms non-IP baselines but remains inferior to Joint_IPOR_PPO scheme, highlighting the superiority of the proposed Joint_IPOR_PPO scheme.

As shown in Figure 11, the test-set classification accuracy of seven schemes is compared. The results indicate that the Joint_IPOR_PPO scheme performs best, achieving an accuracy of 98.0%. The scheme that also incorporates the IP framework but uses the A3C algorithm ranks second with 96.6%, demonstrating that introducing IP knowledge can stably improve accuracy across different DRL algorithms. The scheme combining the IP framework with Q-learning achieves an accuracy of 86.7%, which is 11.3 percentage points lower than PPO, reflecting that PPO is more advantageous. The three PPO baseline schemes without IP indicators, which only optimize offloading and resource

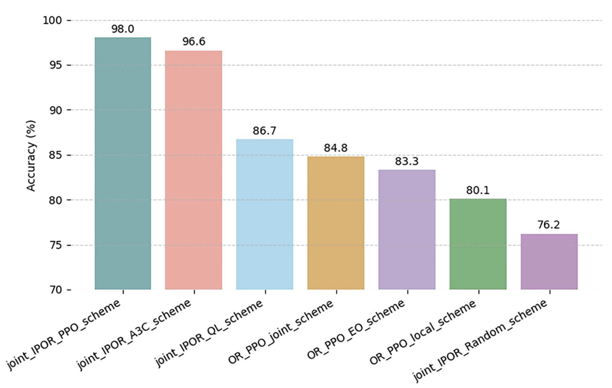

Figure 11 (Color online) A comparison of the accuracy rates between the proposed scheme and the other six baseline schemes.

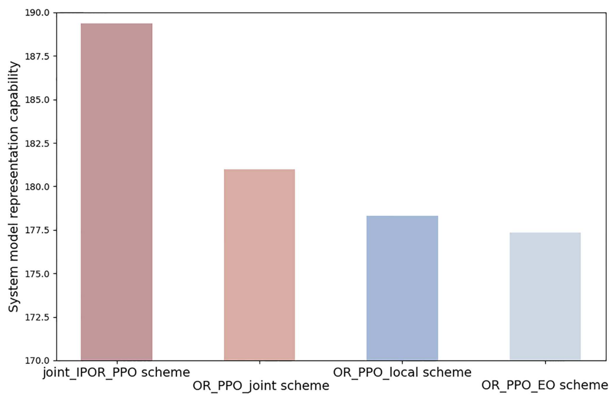

Figure 12 (Color online) The performance of the system model representation capability of the scheme of the joint IP framework.

allocation, exhibit overall significantly lower accuracy, with the scheme using a random policy performing the worst at only 76.2%, further confirming that the lack of effective learning and knowledge aggregation leads to unstable model selection. Overall, knowledge-driven model selection based on IP and blockchain sharing, combined with PPO's policy optimization, can significantly improve task accuracy while keeping system overhead under control.

Figure 12 presents a comparative evaluation of system model representation capabilities between the Joint_IPOR_PPO scheme and traditional approaches. The proposed scheme demonstrates the best representation performance, primarily attributed to its dynamic model update mechanism on edge servers, which enables saving optimal model representations. As can be seen in the figure, the quantitative analysis conducted through the IP framework reveals that the Joint_IPOR_PPO scheme achieves a significant improvement in model representation capability, with a 4%–7% increase compared to traditional approaches. This performance improvement can be attributed to our scheme's adaptive learning architecture and efficient model selection mechanism.

6 Conclusion

In this paper, we propose an intelligent resource management framework that integrates model representational capability evaluation into network resource allocation strategies for MEC-assisted IoT environments. By leveraging the IP framework and blockchain technology, we introduce a systematic approach to assessing the effectiveness of intelligent models, thereby facilitating optimal model selection within dynamic and heterogeneous network conditions. Furthermore, we formulate the resource management problem as an MDP and develop a DRL-based optimization scheme that jointly considers delay, energy consumption, and model representation capability. Experimental results demonstrate that compared with other approaches, the proposed PPO-based model selection and resource management scheme exhibits superior convergence characteristics and enhances stability in MEC-enabled IoT environments, thereby significantly improving overall network performance.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2023YFB2904700) and National Natural Science Foundation of China (Grant Nos. 62201419, 62372357).

References

1 Wang K, Yang K, Chen H H, et al. Computation diversity in emerging networking paradigms. IEEE Wireless Commun, 2017, 24: 88–94
2 Yang L, Cao J, Yuan Y, et al. A framework for partitioning and execution of data stream applications in mobile cloud computing. SIGMETRICS Perform Eval Rev, 2013, 40: 23–32
3 Munoz O, Pascual-Iserte A, Vidal J. Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Trans Veh Technol, 2015, 64: 4738–4755
4 Gao Y, Ye Z, Yu H. Cost-efficient computation offloading in sagin: a deep reinforcement learning and perception-aided approach. ArXiv:2407.05571
5 Zhang S, Cui G, Long Y, et al. Joint computing and communication resource allocation for satellite communication networks with edge computing. China Commun, 2021, 18: 236–252
6 Zhu G, Wang Y, Huang K. Broadband analog aggregation for low-latency federated edge learning. IEEE Trans Wireless Commun, 2020, 19: 491–506
7 Le H Q, Al-Shatri H, Klein A. Efficient resource allocation in mobile-edge computation offloading: completion time minimization. In: Proceedings of IEEE International Symposium on Information Theory (ISIT), 2017. 2513–2517
8 Wang Y, Wu S, Wang Y, et al. Goal-oriented transmission scheduling for energy-efficient wireless networked control in SAGIN: an AoI-thresholding mechanism. IEEE Trans Veh Technol, 2024, 73: 11503–11517
9 Cao X, Wang F, Xu J, et al. Joint computation and communication cooperation for energy-efficient mobile edge computing. IEEE Int Things J, 2019, 6: 4188–4200

10 Tang Q, Fei Z, Li B, et al. Computation offloading in LEO satellite networks with hybrid cloud and edge computing. IEEE Int Things J, 2021, 8: 9164–9176
11 Song Z, Hao Y, Liu Y, et al. Energy-efficient multiaccess edge computing for terrestrial-satellite Internet of Things. IEEE Int Things J, 2021, 8: 14202–14218
12 You C, Huang K, Chae H, et al. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans Wireless Commun, 2017, 16: 1397–1411
13 Nguyen M D, Ajib W, Zhu W P, et al. Integrated computation offloading, UAV trajectory control, and resource allocation against jamming in SAGIN. In: Proceedings of the 99th Vehicular Technology Conference, 2024. 1–5
14 Jia M, Zhang L, Wu J, et al. Joint computing and communication resource allocation for edge computing towards huge LEO networks. China Commun, 2022, 19: 73–84
15 Chen Q, Meng W, Quek T Q S, et al. Multi-tier hybrid offloading for computation-aware IoT applications in civil aircraft-augmented SAGIN. IEEE J Sel Areas Commun, 2023, 41: 399–417
16 Zhang L, Jabbari B, Ansari N. Deep reinforcement learning driven UAV-assisted edge computing. IEEE Int Things J, 2022, 9: 25449–25459
17 Zhang L, Chakareski J. UAV-assisted edge computing and streaming for wireless virtual reality: analysis, algorithm design, and performance guarantees. IEEE Trans Veh Technol, 2022, 71: 3267–3275
18 Han X, Tian D, Sheng Z, et al. Reliability-aware joint optimization for cooperative vehicular communication and computing. IEEE Trans Intell Transp Syst, 2021, 22: 5437–5446
19 Tang J, Tang F, Long S, et al. Utilizing large language models for advanced optimization and intelligent management in space-air-ground integrated networks. IEEE Network, 2025, 39: 173–181
20 Qin Y, Yang Y, Tang F, et al. Differentiated federated reinforcement learning based traffic offloading on space-air-ground integrated networks. IEEE Trans Mobile Comput, 2024, 23: 11000–11013
21 Qin Z, Wei Z, Qu Y, et al. AoI-aware scheduling for air-ground collaborative mobile edge computing. IEEE Trans Wireless Commun, 2023, 22: 2989–3005
22 Shiri F M, Perumal T, Mustapha N, et al. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. ArXiv:2305.17473
23 Hussain H, Tamizharasan P S, Rahul C S. Design possibilities and challenges of DNN models: a review on the perspective of end devices. Artif Intell Rev, 2022, 55: 5109–5167
24 Liu Y, Wang Y, Yang X, et al. Short-term travel time prediction by deep learning: a comparison of different LSTM-DNN models. In: Proceedings of the 20th International Conference on Intelligent Transportation Systems (ITSC), 2017. 1–8
25 Shwartz-Ziv R, Tishby N. Opening the black box of deep neural networks via information. ArXiv:1703.00810
26 Cheng H, Lian D, Gao S, et al. Evaluating capability of deep neural networks for image classification via information plane. In: Proceedings of European Conference on Computer Vision, 2018. 168–182
27 Zhang W, Wen Y, Wu D O. Collaborative task execution in mobile cloud computing under a stochastic wireless channel. IEEE Trans Wireless Commun, 2015, 14: 81–93
28 Zhao N, Ye Z, Pei Y, et al. Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing. IEEE Trans Wireless Commun, 2022, 21: 6949–6960
29 Chen Q, Wu C, Han S, et al. Intersatellite-link-enhanced transmission scheme toward aviation IoT in SAGIN. IEEE Int Things J, 2025, 12: 11812–11826
30 Xia J, Wang P, Li B, et al. Intelligent task offloading and collaborative computation in multi-UAV-enabled mobile edge computing. China Commun, 2022, 19: 244–256
31 Mao Y, Zhang J, Song S H, et al. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans Wireless Commun, 2017, 16: 5994–6009
32 Zhou Q, Yao Y, Liu Q, et al. Energy consumption minimization of multi-UAV assisted mobile edge computing in Sagin. In: Proceedings of IEEE/CIC International Conference on Communications in China, 2024. 499–504
33 Yu S, Chen X, Zhou Z, et al. When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. IEEE Int Things J, 2021, 8: 2238–2251
34 An P, Du L, Chen Y. Learning-based task offloading and UAV trajectory optimization in sagin. In: Proceedings of the 33rd Wireless and Optical Communications Conference, 2024. 12–16
35 Zhao D, Ding R, Song B. Satellite-assisted 6G wide-area edge intelligence: dynamics-aware task offloading and resource allocation for remote IoT services. Sci China Inf Sci, 2025, 68: 122303
36 Zhao J Y, Yang C Y. Graph reinforcement learning with relational priors for predictive power allocation. Sci China Inf Sci, 2025, 68: 122302
37 Dai J X, Fan H, Zhao Z X, et al. Joint communication and computation design for secure integrated sensing and semantic communication system. Sci China Inf Sci, 2025, 68: 132301
38 Xu W, Yang Z, Ng D W K, et al. Edge learning for B5G networks with distributed signal processing: semantic communication, edge computing, and wireless sensing. IEEE J Sel Top Signal Process, 2023, 17: 9–39
39 Guidotti R, Monreale A, Ruggieri S, et al. A survey of methods for explaining black box models. ACM Comput Surv, 2019, 51: 1–42
40 Tishby N, Pereira F C, Bialek W. The information bottleneck method. ArXiv:physics/0004057
41 Al-Najjar H A H, Kalantar B, Pradhan B, et al. Land cover classification from fused DSM and UAV images using convolutional neural networks. Remote Sens, 2019, 11: 1461
42 Yang S, Chen B. SNIB: improving spike-based machine learning using nonlinear information bottleneck. IEEE Trans Syst Man Cybern Syst, 2023, 53: 7852–7863
43 Schulman J, Moritz P, Levine S, et al. High-dimensional continuous control using generalized advantage estimation. ArXiv:1506.02438
44 Wang D, Zhao N, Song B, et al. Resource management for secure computation offloading in softwarized cyber-physical systems. IEEE Int Things J, 2021, 8: 9294–9304
45 He Y, Zhang Z, Yu F R, et al. Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. IEEE Trans Veh Technol, 2017, 66: 10433–10445