PeerJ Computer Science

KECO: toward efficient task offloading in mobile edge computing with federated knowledge distillation

Zhuang Liu[1], Dong Li[2] and Panfei Yang[2]

[1] Guangzhou Human Resources and Social Security Data Service Center, Guangzhou, Guangdong, China
[2] China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou, Guangdong, China

ABSTRACT

With the diversification and personalization of mobile user demands, as well as the rapid development of distributed computing technology, leveraging mobile edge computing task offloading to provide users with convenient services has become a research focus. However, existing methods still face issues such as time consumption, high resource consumption, and data silos. Based on this perspective, we propose the federated Knowledge distillation based mobile Edge Computing task Offloading (KECO) method to achieve accurate task offloading and system energy saving. Through federated distillation learning, a teacher-student model is established, and complex and parameter-rich models are used as teachers to assist in the training of the student model. The teacher model transfers the knowledge and information it had learned to the student to enhance the generalization ability of the student. Finally, using the lightweight and flexible nature of the student model, it is deployed in a distributed system to implement the task offloading strategy in mobile edge computing. Experiments show that KECO performs well in large-scale task classification and allocation, and has efficient, reliable, and wide application potential.

Subjects Algorithms and Analysis of Algorithms, Artificial Intelligence, Data Mining and Machine Learning, Data Science, Mobile and Ubiquitous Computing

Keywords Mobile edge computing, Task offloading, System energy saving, Federated learning, Distributed systems

Submitted 26 April 2025
Accepted 19 June 2025
Published 13 October 2025

Corresponding author
Panfei Yang, yangpanfei@ceprei.com

Academic editor
Syed Hassan Shah

Additional Information and Declarations can be found on page 18

DOI **10.7717/peerj-cs.3021**

INTRODUCTION

In recent years, 5G based wireless communication technology has been widely used, and mobile edge computing (MEC) (*Dong et al., 2024*; *Yadav & Nanivadekar, 2023*; *Liu et al., 2024*) technology has also been rapidly developed. Diversified task demand processing based on mobile edge environment (*Hu et al., 2024*; *Liu, Liu & Li, 2024*; *Chen et al., 2024*) has become one of the research hotspots. In order to realize distributed task offloading of MEC, we can use the method based on multi-edge device collaborative computing to realize combinatorial optimization and diversified task processing. An efficient distributed task processing method can meet the parallel processing requirements including a single user and multiple mobile computing devices. On the contrary, when the MEC task offloading method based on the traditional framework and optimization model faces the large-scale and multi-class tasks in the mobile communication environment, the system

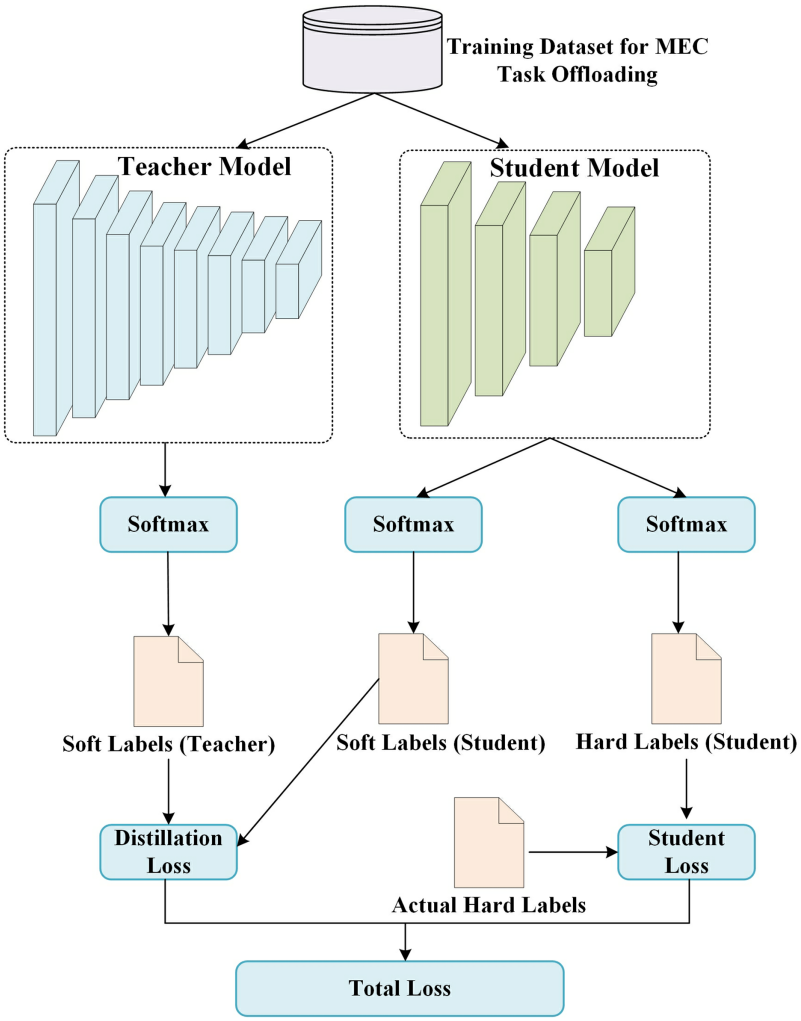

Figure 1 **The MEC task off loading under federated knowledge distillation.**
Full-size DOI: 10.7717/peerj-cs.3021/fig-1

may encounter the problems of increased computing cost, sudden increase of communication cost and load imbalance. Therefore, it is necessary to explore an efficient task processing strategy in the current process of MEC.

At present, many new computing paradigms are developing rapidly and have attracted wide attention. In the face of increasing user task requests and diversified task types, combining different domains has become a trend to solve this problem. As shown in Fig. 1, the purpose of this work is to combine the distributed computing idea and the frontier artificial intelligence, and realize the integration of multi-domain technologies, so as to achieve the reasonable allocation of collaborative computing tasks in the MEC environment. The goal is to design and implement efficient collaborative task processing methods, achieve reasonable task allocation and system energy saving, and reduce task processing costs.

Federated learning framework (*Lee et al., 2024*; *Li et al., 2024*) is adopted to realize parameter interaction between multiple mobile computing devices and parameter update

of calculation model without exposing the local private data of each mobile device, and multi-scenario task processing and global energy consumption minimization are realized from the whole system level. At the same time, based on the big model idea (*Xu et al., 2024*; *Yuan et al., 2024*), the larger deep neural network model (*Darban et al., 2024*; *Chauhan et al., 2024*) obtained by learning is transferred to the lightweight network model. The student model learns the parameters of the model by imitating the output information of the teacher model on the agent data set. Under the condition that the task data is constantly updated, a federation knowledge distillation process under the incremental learning mode for task collaborative processing is realized through the interactive learning between the large model and the mobile lightweight model, so as to realize the offloading strategy of the collaborative computing task.

The main contributions of this work are as follows:

(1) Aiming at the rapid increase in the number of task requests of mobile users and the diversification of task categories, a task offloading method of MEC based on federated knowledge distillation learning is proposed;

(2) A joint task offloading model based on knowledge distillation is proposed. By federating the data of participating aggregation nodes, every mobile device can learn the latest task offloading strategy. Meanwhile, transfer learning between teacher model and student model is introduced to improve the universality of the algorithm.

(3) Experimental verification shows that this method can achieve reasonable task offloading strategy and system energy saving. Compared with many advanced methods, it convincingly demonstrates its advantages in task offloading in MEC environments.

RELATED WORK

In this section, we will cover the related work to MEC, federated machine learning, and federated knowledge distillation.

In terms of MEC, *Loutfi et al. (2024)* recently aimed to outline the intersection between mobile awareness and MEC on 6G networks. The general concept of MEC in 6G mobile network is introduced. This will highlight the integration between MEC and 6G, bringing more efficient network and service migration to the edge, reducing latency and enhancing the user experience. At the same time, the survey discusses augmented reality and MEC applications. The survey discusses the integration of mobile awareness and MEC in upcoming mobile applications and highlights the need for 6G networks. The result of this integration is seamless communication during the handover between the service base station and the target base station. This study helps to understand the future trends of mobile perception and MEC operation in 6G mobile communications. Finally, it provides a comprehensive overview of the challenges and future research directions of MEC mobility management in 6G mobile networks, highlighting the complexity and potential of integrating mobile awareness and mobile edge. *Shah et al. (2023)* studied the utility function maximization problem by jointly optimizing the computing power, user association, performance, duration and location of user equipment (UE) in unmanned aerial vehicle (UAV)-assisted MEC networks under disaster scenarios. Several constraints are considered, including latency, quality of service, and coverage. The optimization

problem is a mixed integer nonlinear programming problem. Finally, a multi-stage offloading algorithm based on learning algorithm and interior point method is proposed to obtain a feasible solution. *Alzubi et al. (2023)* address this issue by proposing a blockchain-enabled security management framework for MEC environments. This approach provides another level of security and includes blockchain security features such as tamper-resistant, immutable, transparent, traceable, and distributed ledgers in MEC environments. The framework ensures the secure storage of data in the MEC environment. *Wu et al. (2023)* proposed a MEC system based on Software Defined Network (SDN). In MEC systems, SDs can offload compute tasks to edge servers to reduce processing latency and avoid energy waste. At the same time, with the programmability, scalability and isolation of the control plane and data plane of SDN, the SDN controller can manage edge devices within the MEC system. Secondly, based on random game, the problem of computing offload and resource allocation in MEC system is studied, and a model of computing offload based on random game is established. It is further proved that the multi-user stochastic game in this system can reach Nash equilibrium. At the same time, each SD is further treated as an independent agent, and a random-game based resource allocation algorithm with preferential experience playback (SGRA-PERs) is designed to minimize energy consumption and processing latency of multi-agent reinforcement learning. *Xiao et al. (2022)* proposed a MEC service migration method based on alliance game and location awareness for the redistribution of mobile users in crowded scenarios. The proposed methods include: using the improved K-means clustering method, MEC servers are divided into several alliances according to their Euclidean distance; discovering hot spots in each alliance region and dispatch services according to their corresponding cooperation; migrating services to appropriate edge servers to achieve high utilization and load equity among consortium members. To address the performance degradation due to interference between radar sensing and MEC, *Huang et al. (2022)* utilized advanced intelligent reflectors (IRS) to improve the performance of radar sensing and MEC. The interaction between computational offload transmission and radar perception in MEC is further characterized by using radar estimation information rate as the performance index of radar perception and considering the assistance of IRS. Under the premise of meeting the requirement of radar estimated information rate and calculating offloading delay constraint, an optimization problem is proposed to minimize the total energy consumption of all devices. Finally, an effective algorithm is proposed to optimize the computing and communication resources.

The above work represents the latest research trends in MEC, covering topics such as 6G networks, blockchain, dynamic resource allocation, security and privacy, joint learning, software-defined networking, and service migration. These studies contribute to the advancement of MEC technology and provide valuable insights for future research and development in the field.

On the federal machine learning front (*Chen, Zhou & Zhou, 2023*), recently *Nguyen et al. (2022)* conducted a comprehensive investigation into the use of federated learning (FL) in smart healthcare. It starts with the latest developments in FL, motivation and requirements for using FL in smart healthcare. It then discusses the latest FL designs in

smart health, from resource aware FL, security and privacy aware FL to incentive FL and personalization FL. This was followed by an updated review of FL's emerging applications in key healthcare areas, including health data management, remote health monitoring, medical imaging and COVID-19 detection. It concludes with an analysis of several recent FL-based smart healthcare projects and highlights key lessons learned from the survey. *Zhu et al. (2023)* conducted a comprehensive survey of the challenges, solutions, and future directions of BlockFed (BlockFed). First, it identifies key problems in federated learning and explains why blockchain offers a potential solution to these problems. Second, based on how federated learning and blockchain functions are integrated, the existing system models are divided into three categories: decoupling, coupling, and overlapping. Then, the advantages and disadvantages of these three system models are compared, the disadvantages are considered as the challenges BlockFed faces, and the corresponding solutions are studied. *Sun et al. (2023)* propose a fine-grained training strategy for federated learning to accelerate its convergence in MECs with dynamic communities. The scheme is based on multi-agent reinforcement learning, which enables each edge node to adaptively adjust its training strategy (aggregation time and frequency) according to network dynamics, while compromising each other to improve the convergence of federation learning. To further adapt to the dynamic community in MEC, they propose a meta-learning-based scheme where new nodes can learn from other nodes and rapidly migrate scenes, thereby further accelerating the convergence of federated learning. *He et al. (2023)* proposed an architecture that combines digital twin (DT) and MEC technologies with FL framework, in which DT networks can virtually mimic the state and network topology of physical entities (pe) for real-time data analysis and network resource optimization. Using MEC's computing offload technology, the resource constraint of MDs and the congestion of core network are alleviated. FL is further used to construct DT model based on pe running data. Then, based on this, the computational offloading and resource allocation problems are co-optimized to reduce discrete effects in FL. Since the objective function is a stochastic programming problem, a Markov decision process (MDP) model is established and the deep deterministic strategy gradient (DDPG) algorithm is used to solve the objective function. *Sharma et al. (2022)* aim to minimize the total energy consumption of the underlying Unmanned Aerial System (UAS) with MEC and non-orthogonal multiple access (NOMA) systems. Markov decision process (MDP) is used to transform the optimization problem into multi-agent reinforcement learning (MARL) problem. At the same time, in order to achieve the optimal strategy and reduce the total energy consumption of the system, a multi-agent joint reinforcement learning (MAFRL) scheme is proposed.

From the above work, we can observe that federated machine learning has the ability to overcome the problem of data silos through privacy-preserving model training.

In terms of federated knowledge distillation, *Wu et al. (2022)* proposed a federated learning method Federated Knowledge Distillation (FedKD), based on adaptive mutual knowledge distillation and dynamic gradient compression technology. FedKD was validated in three different scenarios requiring privacy protection, and the results showed that through centralized model learning, FedKD minimized communication costs by

Table 1 Comparison of different types of federal knowledge distillation methods. The main learning methods of federated knowledge distillation which integrates the technical features, applicable scenarios and performance.

Type	Technology	Advantages	Limitations
Parameter aggregation	The client uploads logits or soft labels, and the server aggregates them for distillation.	This reduces communication overhead and mitigates the impact of non-independent and identically distributed (Non-IID) data.	Generate soft targets by relying on public datasets, with relatively weak privacy protection.
Local distillation	The teacher-student model distillation is completed locally on the client side, and only the student model is uploaded.	It protects the privacy of the original data and supports model heterogeneity.	The local computing resources are highly consumed, thus an efficient distillation strategy needs to be designed.
Differential privacy enhanced version	Adding noise (such as the Laplace mechanism) during the logits transmission or aggregation stage.	Complies with strict privacy protection requirements (ε-DP), with strong compliance.	Noise introduction leads to a decrease in model performance (approximately 3–5% accuracy loss).
Dynamic block-level distillation	Divides the model into blocks for transmission, and the client selectively receives and distills the key modules.	Reduces 31% of communication bandwidth and adapts to heterogeneous computing capabilities.	The inter-block dependencies are complex, and the convergence speed is relatively slow.
Single-round distillation	The client uploads the local model only once, and the server aggregates the distillation in one go (One-Shot).	The communication cost is extremely low, suitable for high-latency networks.	The model performance is limited by the single-round knowledge fusion, and the effect is poor in Non-IID scenarios.
Multimodal distillation	Cross-modal knowledge transfer (such as from text to image), client-side sharing of modality-independent features.	Breaks through the limitations of single-modal data, enhances the generalization ability of the model.	Requires the design of cross-modal alignment loss function, with high training complexity.

94.89% and achieved competitive results. FedKD offers the potential to effectively deploy privacy-protecting intelligent systems in many scenarios, such as smart healthcare and personalization. *Han et al. (2022)* proposed FedX, an unsupervised federated learning framework. Models learn unbiased representations from decentralized and heterogeneous local data. It uses bilateral knowledge distillation with comparative learning as the core component, allowing federated systems to operate without requiring clients to share any data characteristics. In addition, its adaptive architecture can serve as an add-on module to existing unsupervised algorithms in federated environments. *Ma et al. (2022)* proposed continuous distillation Federated Learning (CFeD) to solve the problem of catastrophic forgetting under FL. CFeD performs knowledge distillation on both the client and server, with each party independently owning an unlabeled proxy dataset to mitigate forgetting. In addition, CFeD assigned different learning objectives to different clients, namely learning new tasks and reviewing old tasks, aiming to improve the learning ability of the model. *Yao et al. (2023)* proposed a new approach to heterogeneous federated learning, fedGKD, which incorporates knowledge from historical global models and guides local training to mitigate the "customer drift" problem. The fedGKD was evaluated by conducting a large number of experiments on various computer vision (CV) and natural language processing (NLP) datasets in different heterogeneous settings (*i.e.*, CIFAR-10/100, Tiny-Imagenet, AG News, SST5). The proposed method guarantees convergence under general assumptions and is superior to the most advanced baselines under non-Independent and Identically Distributed (non-IID) federated settings. *Chen et al. (2023)* proposed a federated learning framework based on transferred knowledge in a distributed system with

limited resources. A federated learning optimization problem based on knowledge distillation considering dynamic local resources is solved. This method uses knowledge distillation for federated learning, which avoids taking up expensive network bandwidth or bringing heavy burden to the network. Theoretical analysis proves the convergence of learning process. Table 1 is a comparison of the main learning methods of federated knowledge distillation, which integrates the technical features, applicable scenarios and performance.

To sum up, with the continuous development of machine learning, researchers have integrated the most cutting-edge technologies such as natural language models and federated learning to achieve multi-domain integration and create a new computing paradigm.

FORMALIZATION

For the local computing task latency, when users process some tasks locally, time delay can be set as t_L, which is determined by task allocation ratio ε, task quantity A, central processing unit (CPU) resource required by local processing each bit of data cpu and local computing resource C_L, which can be expressed as follows:

$$t_L^{com} = \frac{\varepsilon \times A \times cpu}{C_L}.$$

The computing energy consumption calculated locally by the user can be expressed by E_L, the size of which is related to the user's effective switching capacitance ρ and its chip properties, which can be expressed as follows:

$$E_L = \varepsilon \times A \times cpu \times \rho \times {C_L}^2.$$

In a multi-edge and single-user MEC system, user send s part of the task to the ith edge device, the expression for the transfer rate $V_{tr}(i)$ is as follows:

$$V_{\mathrm{tr}}(i) = B_i^{tr} \log_2\left(1 + \frac{P_i^{tr}|S_i^{tr}|^2}{B_i^{tr} \cdot \delta_i^{tr}}\right),$$

where ${B_i}^{tr}$ represents the bandwidth of the user to the ith edge device, ${P_i}^{tr}$ represents the transmitted power of the user to the ith edge device, ${S_i}^{tr}$ represents the status information between the user and the nth edge device, and δ_i^{tr} represents the power spectral density of noise.

Then we can derive the expression of the task transfer delay $t_{tr}(i)$ for the user to send the task to the ith edge device, which can be expressed as follows:

$$t_{\mathrm{tr}}(i) = \frac{A \cdot \varepsilon_i}{V_{tr}(i)},$$

where $\varepsilon_i (i = 1, 2, \ldots, \mathrm{I})$ indicates the proportion of tasks assigned to the ith edge device.

Then, we can get the expression of transmission energy consumption $E_{tr}(i)$ for the user to send the task to the ith edge device, that is:

$$E_{\mathrm{tr}}(i) = P_i^{tr} \cdot t_{\mathrm{tr}}(i).$$

While the user computes some tasks locally, sends the remaining tasks to the edge devices. The task computation delay of the ith edge device computing task can be defined as follows:

$$t_i^{com} = \frac{\varepsilon_i \times A \times cpu_i}{C_i},$$

where C_i indicates the computing resources of the ith edge device, and cpu_i indicates the number of CPU revolutions required by the ith edge device to compute 1 bit of data.

Based on the above, it can be deduced that the computing energy consumption E_i^{com} for the user to transmit the task to the ith edge device is:

$$E_i^{com} = \varepsilon_i \times A \times cpu_i \times \rho_i \times {C_i}^2,$$

where ρ_i represents the effective switching capacitance of the ith edge device.

Finally, for the modeling of energy consumption problem, the following optimization problem can be constructed from the formalized process:

$$\min_{\varepsilon,\varepsilon_i,B_i^{tr},P_i^{tr}} \left(\varepsilon \times A \times cpu \times \rho \times {C_L}^2 + \sum_{i=1}^{I} \left(P_i^{tr} \cdot t_{\mathrm{tr}}(i) + \varepsilon_i \times A \times cpu_i \times \rho_i \times {C_i}^2 \right) \right).$$

THE PROPOSED ALGORITHM

System architecture

This article mainly introduces the architecture of the MEC which consists of multiple edge devices and a user, and describes how users can execute a computing task locally and offload the task to multiple edge devices at the same time. In particular, we construct a data processing and transfer model based on the computing latency of a user executing a task locally, the computing latency of a user transferring a task to multiple edge devices, and the computing latency of a task executed on the edge devices.

As shown in the Fig. 2, the MEC assisted offloading calculation model in the scenario of single user and multiple edge devices is constructed. The system consists of multiple edge devices and a single user terminal which together act as a data processing center. The edge equipment and a single user jointly build the federated knowledge distillation model, and perform parameter updating and incremental training. To optimize system energy consumption, this work proposes a joint allocation scheme based on MEC task allocation ratio, user transmit power and bandwidth. The user computes some tasks locally, and then sends the remaining computing tasks to multiple edge devices deployed at the network edge. The edge devices calculate the tasks after receiving the tasks. The calculation delay and transmission delay of the tasks are taken into account so as to construct the system model.

Main idea of proposed method

In this work, an innovative edge federation learning algorithm is designed, which incorporates knowledge distillation technology. Compared with the traditional federated learning style, this marginal federated learning based on knowledge distillation has its unique features in each round of learning process. As shown in the Fig. 3, clients selected to

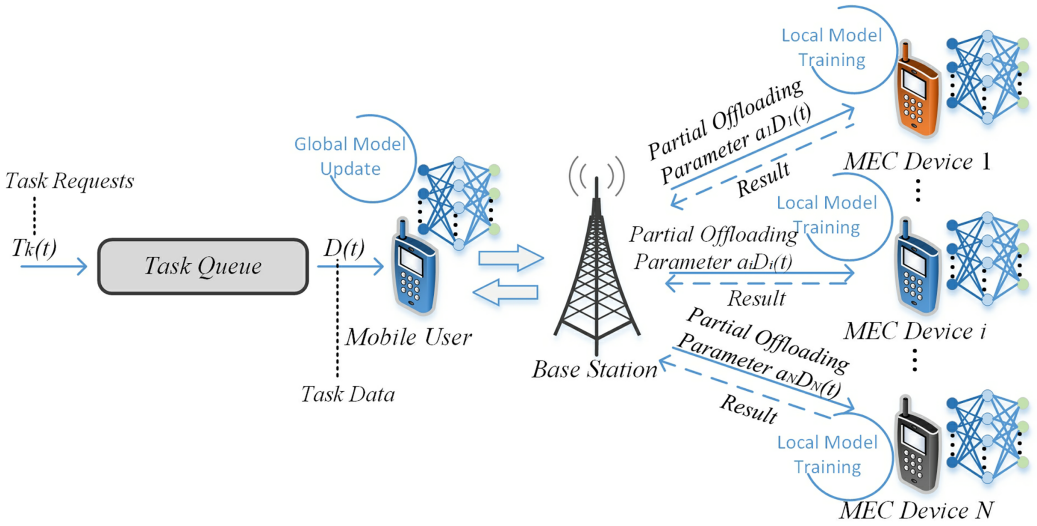

Figure 2 **The view of system architecture.** Full-size DOI: 10.7717/peerj-cs.3021/fig-2

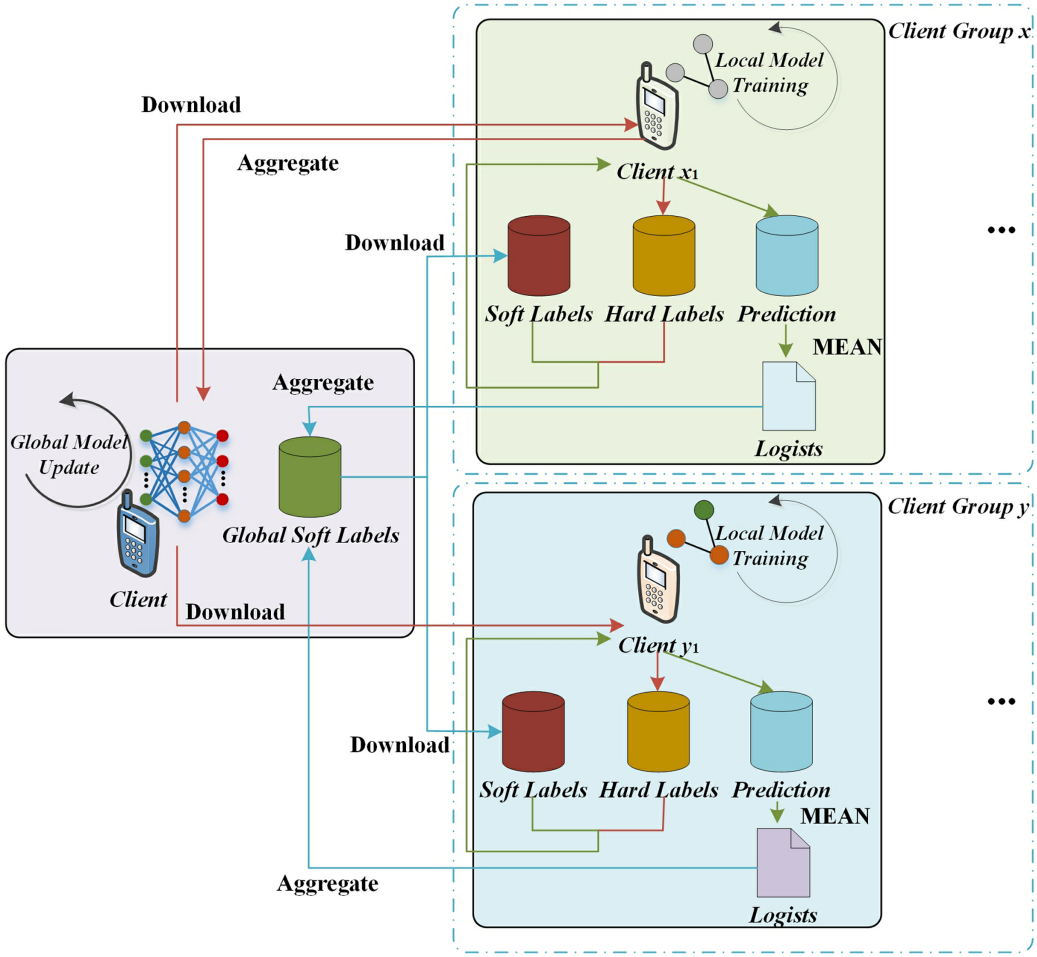

Figure 3 **The mobile edge task off loading based on federated knowledge distillation.** Full-size DOI: 10.7717/peerj-cs.3021/fig-3

participate in federated learning download not only the global model from the edge server, but also the global soft label. During the local training process, the client uses both the local hard label and the downloaded global soft label as optimization targets. In addition, it generates new sample logits while the client trains the local model. The edge server will collect the local model parameters and sample logits uploaded by the client for aggregation operation. It trains the global model and generates new global soft labels. These new global models and global soft labels will be used for the next round of federated learning. The edge federated learning architecture based on knowledge distillation proposed in this section consists of an edge server and several clients.

In the framework of the edge federated learning algorithm based on knowledge distillation technology designed in this work, we assume that the federated learning training process consists of a total of T iterations. After tth iteration, the client completes the local model update, the member of the client set x, will upload the local model parameters and the sample logits to the edge server. The member of the client set y uploads only the sample logits to the edge server. The edge server then performs an aggregation of the global model based on the received data, which can be defined as follows:

$$W_{\mathrm{t+1}} = \frac{\sum_{i=1}^{I} \left|A_i'\right| \times W_{t+1}^{i}}{\sum_{i=1}^{I} \left|A_i'\right|},$$

where W represents the parameters of the model. I is the total number of clients in client set x during federated learning. For any client i, after completing tth local training, it uploads the updated model parameters, which is referred to as W_{t+1}. Meanwhile, A_i' represents the size of the local data set for the ith client. The server aggregates global soft labels based on this information. The process is as follows:

$$G_{\mathrm{t+1}} = \frac{\sum_{i=1}^{I} \left|A_i'\right| \times G_{t+1}^{i}}{\sum_{i=1}^{I} \left|A_i'\right|} + \Delta_{G+1},$$

where G_{t+1} represents the average logits of the sample uploaded by client i in client set x after completing tth local training; And Δ_{G+1} represents the local soft label gradient generated by clients in the client set y at the end of tth local training to supplement and correct the global soft label information.

In the client set y, assume that when the time threshold T ends, it receives from the set the sample logits uploaded by u clients after $(t-1)$th training, and receives the sample logits uploaded by v clients in the set after tth local training:

$$\Delta_{G+1} = \frac{\sum_{i=1}^{u} \left|A_i'\right| \times G_{t}^{i}}{\sum_{i=1}^{u} \left|A_i'\right|} - \frac{\sum_{i=1}^{v} \left|A_i'\right| \times G_{t+1}^{i}}{\sum_{i=1}^{v} \left|A_i'\right|}.$$

In the process of local training, the client not only trains the local model, but also generates the sample logits. In this process, the local model update method of the client is similar to the local model update method in traditional federated learning, that is, each client adopts gradient descent algorithm to update the weight of the model.

$$W_{t+1}^{i} = W_{t}^{i} - \lambda r_i,$$

where λ represents the learning rate of the local model training, and the symbol r_i represents the gradient of the current model parameter W. If client I generates A' sample logits in Tth local training, then eventually it will upload the average value of these sample logits, as G^i_{t+1}, to the edge server, which is defined as follows:

$$G^i_{t+1} = \sum_{a=1}^{A'} G^i_a / A'.$$

The edge federation learning based on knowledge distillation can use both hard label and soft label knowledge in model training. The prediction model should adhere not only to the hard label of the local dataset sample, but also to the corresponding soft label. During the local training of each client in federated learning, the model loss function is computed to evaluate the model's performance in approximating hard and soft labels.

$$Loss(W) = \delta L_s(l_b, G) + (1-\delta) L_k(l'_b \| G).$$

G represents the prediction result of the model and δ is a hyperparameter in the interval (0,1), where $l_b{}'$ represents the soft label and l_b represents the hard label. $L_s()$ represents the cross entropy loss function, which measures the difference between the model prediction and the hard label. Meanwhile, $L_k()$ represents the divergence loss function, which measures the difference in distribution between the model prediction and the soft label. If in the ith client, the number of local data sets is d_i, then the calculation of the loss function can be deduced as follows:

$$L_s(l_b, G) = -\sum_{a=1}^{d_i} l_b \log \frac{\exp(G^a)}{\sum_{a'} \exp(G^{a'})}.$$

$$L_k(l'_b \| G) = -\sum_{a=1}^{d_i} l'_b \log \frac{\exp(G^a) \sum_{a'} \exp(l'^{\,a'}_b)}{\exp(l'^{\,a}_b) \sum_{a'} \exp(G^{a'})}.$$

Through derivation, the optimization function of this work can be calculated as follows:

$$\min_W \left(\Gamma(W) \triangleq \sum_{i=1}^{I} f_i (\delta L_s(W) + (1-\delta) L_k(W)) \right),$$

$$s.t. f_i = \frac{A_i}{\sum_{i=1}^{I} A_i}.$$

From the optimization function, we can observe that the hyperparameter δ plays a decisive role in the knowledge distillation process, which regulates the proportion of hard label loss and soft label loss in the total loss. Therefore, the value of δ has a profound influence on the training effect of the model.

In the initial stage and early stage of model training, the information of hard labels plays an absolutely leading role in improving model performance due to the relatively little knowledge contained in soft labels. However, in the later stage of model training, when the model has fully absorbed the knowledge of hard labels and begins to converge gradually, the introduction of soft label knowledge will further promote the improvement of model performance.

Based on the above considerations, we design a dynamic value method for hyperparameter δ. In the early stage of model training, we let the value be relatively large, so that the model can make full use of the hard label information. Subsequently, we gradually reduce the value until it reaches a minimum threshold. The purpose of this design is to make the model effectively use the hard label information in the training process, and introduce the soft label knowledge at the appropriate time, so as to improve the performance of the model.

Implementation of KECO

The implementation of knowledge distillation in the edge-federated learning method can be carried out by the following process:

Step 1: initialize the global model and the local model (the local model per client). Then select a batch of data D as the dataset of the teacher model.

Step 2: the server sends the global model to each client.

Step 3: each client is trained on the local data set using the global model to get the local model. After the local model is trained, the parameters of the local model are uploaded to the server.

Step 4: the server collects the local model parameters uploaded by all clients. Use integration techniques such as weighted averaging to aggregate all local models into a new global model.

Step 5: soft labels are generated using the teacher model (the original global model) and the integrated model. Through teacher model and integration model, soft labels are obtained respectively. The knowledge distillation (KD) divergence loss between soft labels is calculated, and the global model is optimized.

Step 6: the global model is optimized according to the loss function and updated to a new global model. The updated global model is sent to each client to begin the next round of training.

By distilling multiple local models into one global model, the overhead of model transmission can be significantly reduced. Ensemble learning technology can integrate multiple local models with poor performance into a global model with good performance, thereby improving the overall efficiency. By distilling knowledge using unlabeled data or generated data, models can be trained without exposing client data, protecting user privacy. Through the above steps and strategies, knowledge distillation can effectively improve the performance of the model in the edge-federated learning method, while reducing transmission overhead and protecting user privacy.

Additive homomorphic encryption

To ensure the security of private data, additive homomorphic encryption technology is adopted in this study. This technique can effectively prevent the leakage of task information, and at the same time, it can provide corresponding algorithm support for elements in plain text space. In this article, a method is proposed, which can not only encrypt two numbers, but also calculate the encryption results of these numbers. For illustration purposes, we set the real number a to be encrypted in the form $<a>$. Based on

this setting, for any two unencrypted numbers a and b, their encryption sum satisfies the relationship $<a> + <b> = <a + b>$. In addition, we also implement the multiplication between ciphertext and plaintext, *i.e.*, $b \cdot <a> = <ba>$, where b remains unencrypted. In this way, we can calculate the sum product of plaintext and ciphertext without leaving the encrypted digital environment.

Further, we extend this operation mechanism to the operation of vectors and matrices. Specifically, we can use $b^{\mathrm{T}}<a> = <b^{\mathrm{T}}a>$ to compute the inner product of vectors of plaintexts a and b, and $b \circ <a> = <b \circ a>$ to compute their component product. As for the specific details of matrix operation, because of its similarity with vector operation, we will not go into details here. It is worth noting that our approach enables the sharing of parameters without exposing information from the edge computing center. Relying on our federated learning model, we are able to efficiently aggregate task request data while ensuring privacy.

During the initialization stage of the federated learning system, a trusted third party (or coordination server) generates a homomorphic encryption public-private key pair. The public key is distributed to all participating clients, while the private key is kept by the designated party. Subsequently, the server issues the initial teacher model (or the locally pre-trained teacher model on the client side), and the architecture of the student model is customized by each client. In the local knowledge extraction stage, the client uses local data to generate probabilistic outputs (logits) through the teacher model, and softens the distribution through a temperature parameter. Subsequently, the client encrypts the soft labels using the homomorphic public key, and may also choose to encrypt intermediate features. In the secure knowledge aggregation stage, the client uploads the encrypted knowledge representation to the aggregation server. The server performs the aggregation operation on the ciphertext, utilizing the homomorphic addition property. In the distributed distillation stage, the authorized party decrypts the aggregated result using the private key, or directly distributes the ciphertext for the client to decrypt. The client trains the student model based on the decrypted global knowledge using the KL divergence loss. Finally, in the model iteration optimization stage, the parameters of the student model are updated to the global model through federated averaging (FedAvg) or other aggregation algorithms. Then, the temperature parameter and the homomorphic encryption level are adaptively adjusted according to the convergence situation.

DISCUSSION

In the MEC scenario, the teacher-student model based on federated knowledge distillation achieves efficient knowledge transfer through hierarchical collaboration: the teacher model deployed in the cloud or high-performance nodes generates soft labels or feature mappings, which are encrypted and aggregated to guide the training of the student model at the edge device end, forming a privacy-protected heterogeneous model collaboration framework. Its advantages are reflected in three aspects: (1) by distilling and compressing, the computing load at the edge end is reduced, such as a significant reduction in parameter quantity; (2) the federated mechanism ensures that data does not leave the domain, meeting the privacy requirements of users; (3) dynamic temperature parameters and

block-level distillation techniques are adapted to network fluctuations. This method has theoretical scalability that can support horizontal expansion of thousands of nodes and cross-modal knowledge transfer, and its adaptability is manifested in robust processing of non-IID data (such as through attention weighting) and elastic deployment in resource-constrained scenarios (single-round/incremental distillation is optional).

RESULTS

In this section, in order to validate the capabilities of the proposed MEC task offloading algorithm based on federated knowledge distillation in task processing and system energy saving, as with the validation process of our previous work (*Li & Yang, 2024*), we combine the proposed KECO algorithm with two other task offloading algorithms (including decision-based offloading strategies such as MDP and DDPG (*Lillicrap et al., 2015*)), and predict-based offloading strategies such as convolutional neural network (CNN)-based or time series (*Krizhevsky, Sutskever & Hinton, 2012*), hereinafter referred to as DDPG-based, CNN-based is compared in the following aspects:

(1) energy consumption
(2) system external service performance
(3) task offloading failure rate.

Experimental setup

In the model training session, the first convolution layer is configured with 32 convolution cores, the size of each convolution kernel is set to 6 × 6, and the step size is set to 1. The matrix size of the maximum pooling layer is set to 2 × 2, and the step size is also 2. As for the second convolution layer, it is equipped with 64 convolution cores, and the remaining parameters are consistent with the first layer. The training cycle of the designed model is usually between 1.5 and 2 h. We set the learning rate of the model to 0.001 and introduced Adam optimization algorithm to achieve dynamic updating of step size. In the early training phase, we set the weight loss ratio at 1:3 to achieve knowledge transfer. While in the later training stage, we adjusted the ratio to 3:1 to accomplish the learning of the specific task objectives. To simulate the scenario where multiple users jointly participate in the training of a federated model, it is set that this deep model has five participants. The dataset is randomly divided into five parts and sent to the five parties. Each party then trains its local model locally and conducts 50 rounds of iterative training simultaneously. The dataset is divided into a training dataset and a test dataset, so as to achieve the cross-validation during training. As the number of training sessions increases, the method gradually improves its performance in terms of accuracy and other metrics, and its results stabilize in the final few rounds of training. The overall training time remains generally between 30 and 58 min, ranging from 25 to 50 rounds, while ensuring the security of the data.

In terms of experimental environment construction, we create a virtualization scenario using OpenStack and deployed Spark, a big data processing framework. All experiments are performed on the same Linux workstation with an Intel Xeon processor, three 3.4 GHz

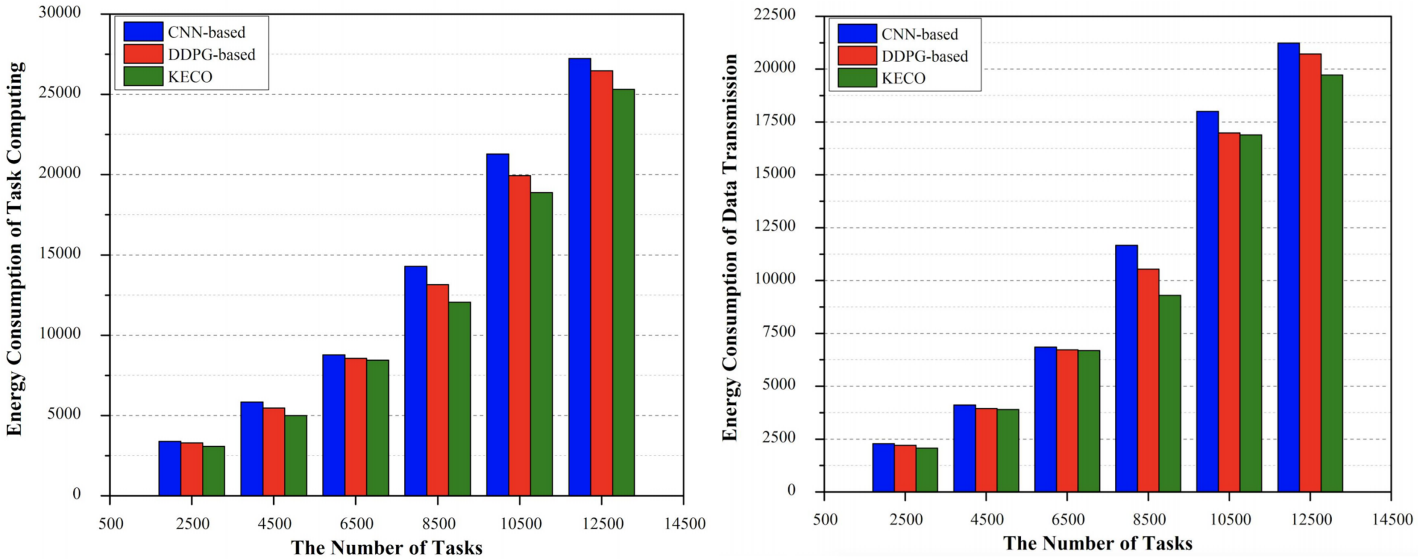

Figure 4 The performance comparison of three methods in terms of energy consumption. Energy consumption for local computing & energy consumption for data transmission. Full-size DOI: 10.7717/peerj-cs.3021/fig-4

CPU cores and 512 GB of RAM. We create 32 processing nodes with different configurations to simulate MEC nodes and resource pools.

Comparison of energy consumption

In this section, we compare the performance of two task offloading methods based on DDPG and CNN with the method proposed in this work in terms of system energy consumption. In this work, the local computing power consumption (*Shi et al., 2023*) and data transmission power consumption during the offloading of MEC tasks are used to measure the energy saving advantages. As can be seen from the Fig. 4A, in terms of local calculation, the energy consumption of each method gradually increases with the increase of data scale. On the whole, the energy consumption of the method proposed in this work is always lower than the other two methods. The energy saving effect of the system proposed in this work is basically equal to the other two methods in the initial stage of task offloading, and its performance is gradually superior to the other two methods in the later stage. In the data transmission process, as shown in the Fig. 4B, we can observe that the three methods produce less system energy consumption in the case of smaller task scale. However, communication costs for all three are increasing as missions scale up. However, the communication energy consumption of CNN method in the later stage is gradually higher than that of the other two methods. For DDPG method, the communication energy consumption in mobile edge environment is higher when the task scale is small, but its value in the later stage is lower than CNN, and it is always greater than the cost of the method proposed in this work. For the method proposed in this work, the energy consumption value is not much different from the other two methods in the early stage, and gradually lower than the other two methods in the later stage.

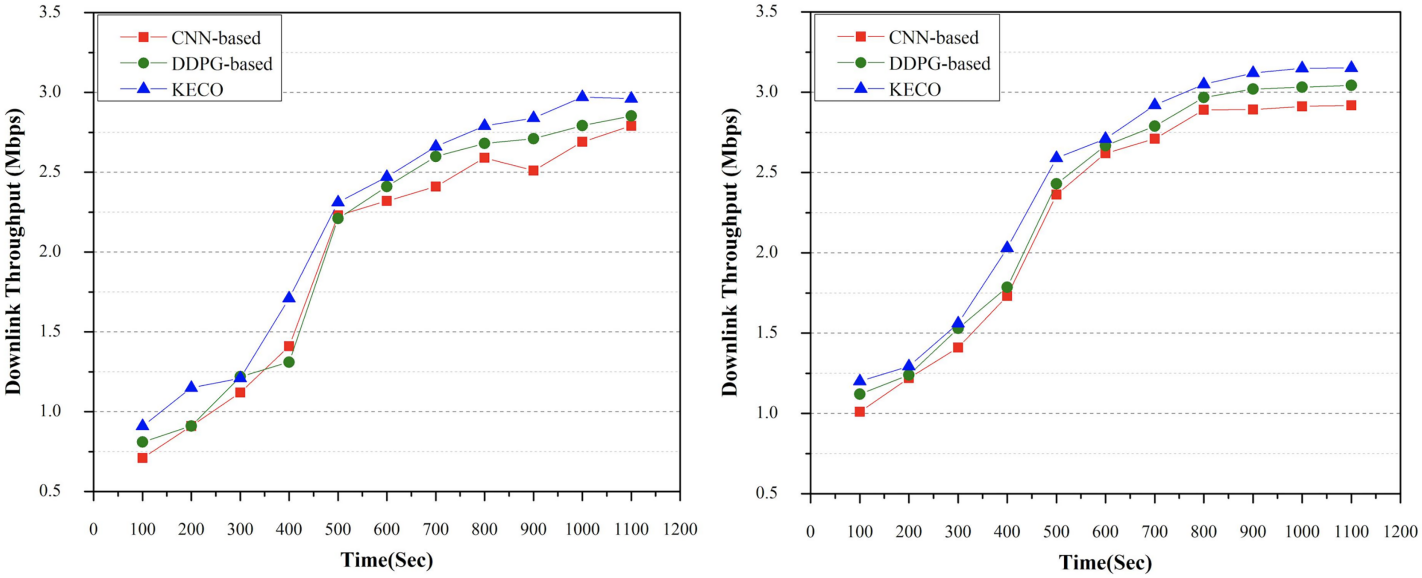

Figure 5 The performance comparison of three methods in terms of system external service performance. Task volume = 8,500 & task volume = 10,500.
Full-size DOI: 10.7717/peerj-cs.3021/fig-5

This is mainly because the method proposed in this work can realize the guidance based on the global soft label knowledge according to the parameters and sample logic learned during model training, so as to calculate the effect and total energy consumption of each feasible scheme. In the process of searching for the optimal scheme, the current optimal scheme obtained by each iteration is taken as the basis. Gradually search and try to find a solution that consumes less energy and costs less computing resources, so as to find a suitable task processing node for the corresponding computing task. This effectively ensures the resource utilization of the whole field to a great extent. In general, we can observe that the method proposed in this work has a good performance in system energy saving. It reduces the extra consumption of valuable computing resources, realizes the reasonable offloading of distributed computing tasks, and improves the overall benefits of the system in the long run.

Comparison of external service performance

In this section, the method proposed in this work is compared with the other two methods in terms of system performance to external services. This work takes system throughput as the main evaluation criterion of external service performance. As shown in Fig. 5A, when the task volume is 8500, the external service performance effects of each of the three methods used for comparison are different. For the CNN method, the external service performance of the system seems to be relatively good in the early stage, but the performance is not very stable in the later stage. For DDPG, the throughput value is low in the initial phase. With the passage of time, its system performance gradually becomes stable, and gradually exceeds the CNN method. The algorithm proposed in this work can reasonably analyze the computing task requirements and make a reasonable resource

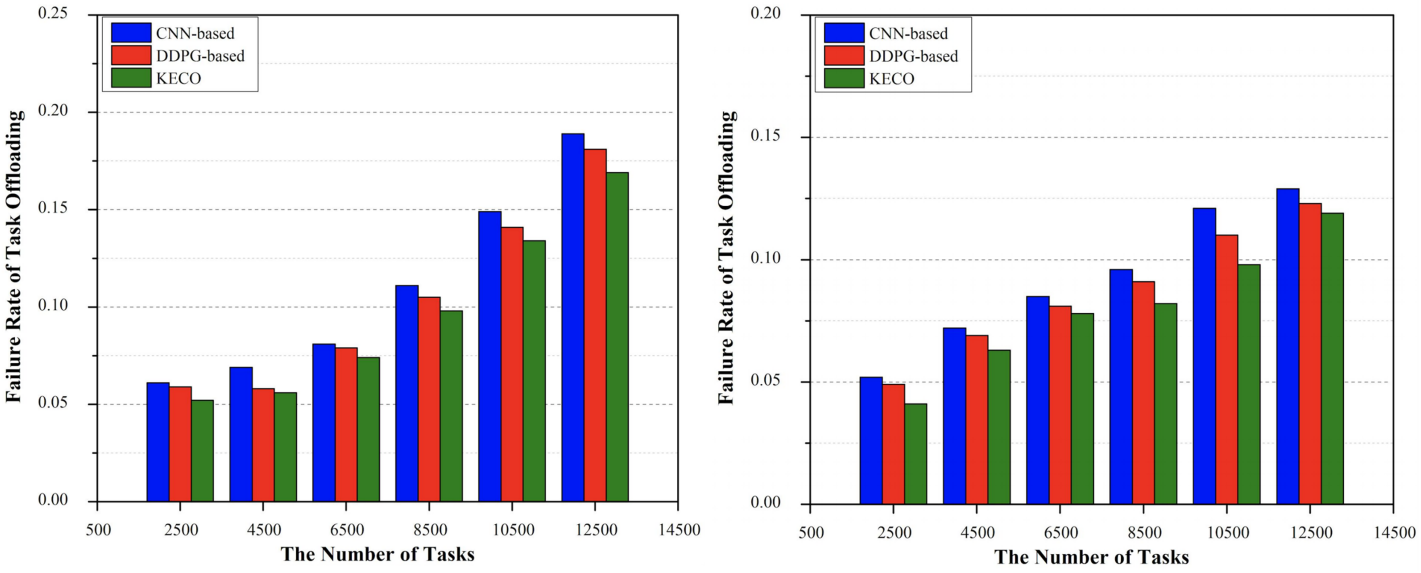

Figure 6 The performance comparison of three methods in terms of the number of task offloading failures. Performance under CPU pressure & performance under memory pressure.
Full-size DOI: 10.7717/peerj-cs.3021/fig-6

allocation strategy, and its external service performance is generally better than the other two methods. With the passage of external service time, its throughput value gradually tends to be stable, and it is slightly higher than the throughput value of CNN and DDPG in the later stage. In another group of experiments, when the task volume is 10,500, as shown in Fig. 5B, in general, the throughput value of the method proposed in this work is gradually higher than that of the other two methods in the later period. For CNN method, in the MEC environment, its external service performance is obvious in the initial stage, but its value is always lower than DDPG. When t = 500 s, the throughput growth rate of these two methods begins to decrease gradually.

As for the method proposed in this work, its performance of external service is not much different from the other two methods in the early stage, and gradually exceeds the other two methods in the later stage. On the whole, we can conclude that the method proposed in this work is relatively efficient and stable in terms of external service effects, which helps to efficiently perform large-scale distributed tasks.

Comparison of number of task offloading failures

In the actual scenario of MEC tasks offloading, some distributed computing tasks often fail to be uninstalled because some selected computing task processing nodes cannot meet the corresponding task requests. In this section, we use simulation tools to simulate some dynamic node failures, take CPU and memory pressure (*Li, Chen & Song, 2023*) as examples to simulate the impact of real pressure environment on task offloading, and compare with the other two methods in terms of the number of task offloading failures. As shown in Figs. 6A and 6B, with the expansion of task scale, the number of task offloading failures of CNN and DDPG increases rapidly during task deployment. For the algorithm

proposed in this work, with the expansion of task scale, the number of task offloading failures increases significantly slower than that of the other two methods, and the number of task deployment failures remains at a minimum.

This is mainly because the other two methods are difficult to carry out a reasonable and thorough analysis of the current remaining resource information in each computing task processing node and the MEC task requirements in the MEC network. If the resource requirement of a computing task is greater than or equal to the remaining resources of the node that processes the computing task, the computing task may fail to be uninstalled. On the contrary, the method proposed in this work uses the federated knowledge distillation network model to reasonably guide the selection of the current task offloading strategy according to labels and logical values, which can ensure that the remaining resources of each selected node are greater than the resource demand of the corresponding task and obtain certain system benefits under the condition of ensuring effective calculation and communication. Therefore, to a large extent, it can dynamically and adaptively find the appropriate task processing node from the cluster for most mobile computing task requests, so as to complete the efficient execution of collaborative computing tasks.

CONCLUSIONS

This work investigates the combination of federated knowledge distillation learning and distributed task offloading strategies in MEC environments. In this article, (1) an optimized task offloading method based on federated knowledge distillation in MEC environment is proposed; (2) a federation model of MEC task offloading based on knowledge distillation framework is designed. A federation model and a deep convolutional network model trained by updating parameters and sample logic values between edge computing models are used to achieve reasonable offloading of computing tasks and system energy saving; (3) compared with the most advanced methods, the superiority of the proposed method in collaborative task offloading is evaluated. The experimental results show that it can unload the task reasonably in the direction of minimizing the system energy consumption and reduce the task execution consumption.

Because the choice of the number of layers of the neural network in the deep network model is an open problem, the number of layers of the deep neural network should be adjusted in the process of exploring the optimal task offloading strategy. The method proposed in this article opens a new door for future research on cooperative processing of mobile tasks. We plan to extend it to large-scale task processing in larger experimental settings, taking into account the total cost of federal learning.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by “Artificial Intelligence Software and Hardware to Adapt Migration Technology and Tools”, grant number NIVY227201160. The funders had no

role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
Artificial Intelligence Software and Hardware to Adapt Migration Technology and Tools: NIVY227201160.

Competing Interests

Zhuang Liu is employed by Guangzhou Human Resources and Social Security Data Service Center.

Author Contributions

- Zhuang Liu conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Dong Li performed the experiments, prepared figures and/or tables, and approved the final draft.
- Panfei Yang conceived and designed the experiments, performed the experiments, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available in the Supplemental File.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.3021#supplemental-information.

REFERENCES

Alzubi JA, Alzubi OA, Singh A, Alzubi TM. 2023. A blockchain-enabled security management framework for mobile edge computing. *International Journal of Network Management* **33(5)**:e2240 DOI 10.1002/nem.2240.

Chauhan VK, Zhou J, Lu X, Molaei S, Clifton DA. 2024. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review* **57(9)**:250 DOI 10.1007/s10462-024-10862-8.

Chen Z, Tian Pu, Liao W, Chen X, Xu G, Yu W. 2023. Resource-aware knowledge distillation for federated learning. *IEEE Transactions on Emerging Topics in Computing* **11(3)**:706–719 DOI 10.1109/tetc.2023.3252600.

Chen Y, Zhao J, Hu X, Wan S, Huang J. 2024. Distributed task offloading and resource purchasing in NOMA-enabled mobile edge computing: hierarchical game theoretical approaches. *ACM Transactions on Embedded Computing Systems* **23(1)**:1–28 DOI 10.1145/3597023.

Chen Z, Zhou C, Zhou Y. 2023. A hierarchical federated learning model with adaptive model parameter aggregation. *Computer Science and Information Systems* **20(3)**:1037–1060 DOI 10.2298/CSIS220930026C.

Darban Z, Zahra GIW, Pan S, Aggarwal C, Salehi M. 2024. Deep learning for time series anomaly detection: a survey. *ACM Computing Surveys* **57(1)**:1–42 DOI 10.1016/j.patcog.2024.110874.

Dong S, Tang J, Abbas K, Hou R, Kamruzzaman J, Rutkowski L, Buyya R. 2024. Task offloading strategies for mobile edge computing: a survey. *Computer Networks* **254(6)**:110791 DOI 10.1016/j.comnet.2024.110791.

Han S, Park S, Wu F, Kim S, Wu C, Xie X, Cha M. 2022. FedX: unsupervised federated learning with cross knowledge distillation. In: *European Conference on Computer Vision*. Cham: Springer Nature Switzerland, 691–707.

He Y, Yang M, He Z, Guizani M. 2023. Computation offloading and resource allocation based on DT-MEC-assisted federated learning framework. *IEEE Transactions on Cognitive Communications and Networking* **9(6)**:1707–1720 DOI 10.1109/TCCN.2023.3298926.

Hu Z, Niu J, Ren T, Liu X, Guizani M. 2024. SITOff: enabling size-insensitive task offloading in D2D-assisted mobile edge computing. *IEEE Transactions on Mobile Computing* **24(3)**:1567–1584 DOI 10.1109/TMC.2024.3483951.

Huang N, Wang T, Wu Y, Wu Q, Quek TQS. 2022. Integrated sensing and communication assisted mobile edge computing: an energy-efficient design via intelligent reflecting surface. *IEEE Wireless Communications Letters* **11(10)**:2085–2089 DOI 10.1109/LWC.2022.3193706.

Krizhevsky A, Sutskever I, Hinton G. 2012. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* **25(2)**:84–90 DOI 10.1145/3065386.

Lee J, Solat F, Kim TY, Poor HV. 2024. Federated learning-empowered mobile network management for 5G and beyond networks: from access to core. *IEEE Communications Surveys & Tutorials* **26(3)**:2176–2212 DOI 10.1109/COMST.2024.3352910.

Li L, Chen X, Song C. 2023. NonPC: non-parametric clustering algorithm with adaptive noise detecting. *Intelligent Data Analysis* **27(5)**:1347–1358 DOI 10.3233/IDA-220427.

Li R, Wang C, Zheng Z, Huang H. 2024. Enhancing federated learning with self-determining mechanism in MEC. In: *2024 International Conference on Computing, Networking and Communications (ICNC)*. Piscataway: IEEE, 1006–1010 DOI 10.1109/ICNC59896.2024.10556194.

Li D, Yang P. 2024. Distributed unmanned aerial vehicle cluster testing method based on deep reinforcement learning. *Applied Sciences* **14(23)**:11282 DOI 10.3390/app142311282.

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. 2015. Continuous control with deep reinforcement learning. ArXiv DOI 10.48550/arXiv.1509.02971.

Liu X, Liu J, Li W. 2024. Truthful mechanism for joint resource allocation and task offloading in mobile edge computing. *Computer Networks* **254**:110796 DOI 10.1016/j.comnet.2024.110796.

Liu Q, Mo R, Xu X, Ma X. 2024. Multi-objective resource allocation in mobile edge computing using PAES for internet of things. *Wireless Networks* **30(5)**:3533–3545 DOI 10.1007/s11276-020-02409-w.

Loutfi SI, Shayea I, Tureli U, El-Saleh AA, Tashan W. 2024. An overview of mobility awareness with mobile edge computing over 6G network: challenges and future research directions. *Results in Engineering* **23(2)**:102601 DOI 10.1016/j.rineng.2024.102601.

Ma Y, Xie Z, Wang J, Chen K, Shou L. 2022. Continual federated learning based on knowledge distillation. In: *IJCAI*, 2182–2188 DOI 10.24963/ijcai.2022/303.

Nguyen DC, Pham Q-V, Pathirana PN, Ding M, Seneviratne A, Lin Z, Dobre O, Hwang W-J. 2022. Federated learning for smart healthcare: a survey. *ACM Computing Surveys* **55(3)**:1–37 DOI 10.1145/3501296.

Shah Z, Javed U, Naeem M, Zeadally S, Ejaz W. 2023. Mobile edge computing (MEC)-enabled UAV placement and computation efficiency maximization in disaster scenario. *IEEE Transactions on Vehicular Technology* **72(10)**:13406–13416 DOI 10.1109/tvt.2023.3274107.

Sharma H, Budhiraja I, Consul P, Kumar N, Garg D, Zhao L, Liu L. 2022. Federated learning based energy efficient scheme for MEC with NOMA underlaying UAV. In: *Proceedings of the 5th International ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*. New York: ACM, 73–78.

Shi M, Liang F, Chen Y, He Y. 2023. A local cost simulation-based algorithm to solve distributed constraint optimization problems. *PeerJ Computer Science* **9**:e1296 DOI 10.7717/peerj-cs.1296.

Sun W, Zhao Y, Ma W, Guo B, Xu L, Duong TQ. 2023. Accelerating convergence of federated learning in MEC with dynamic community. *IEEE Transactions on Mobile Computing* **23(2)**:1769–1784 DOI 10.1109/tmc.2023.3241770.

Wu G, Wang H, Zhang H, Zhao Y, Yu S, Shen S. 2023. Computation offloading method using stochastic games for software-defined-network-based multiagent mobile edge computing. *IEEE Internet of Things Journal* **10(20)**:17620–17634 DOI 10.1109/JIOT.2023.3277541.

Wu C, Wu F, Lyu L, Huang Y, Xie X. 2022. FedKD: communication efficient federated learning via knowledge distillation. *Nature Communications* **13**:2032 DOI 10.1038/s41467-022-29763-x.

Xiao X, Ma Y, Xia Y, Zhou MC, Luo X, Wang X, Fu X, Wei W, Jiang N. 2022. Novel workload-aware approach to mobile user reallocation in crowded mobile edge computing environment. *IEEE Transactions on Intelligent Transportation Systems* **23(7)**:8846–8856 DOI 10.1109/tits.2021.3086827.

Xu X, Li M, Tao C, Shen T, Cheng R, Li X, Xu C, Tao D, Zhou T. 2024. A survey on knowledge distillation of large language models. ArXiv DOI 10.1109/COMST.2024.3352910.

Yadav S, Nanivadekar S. 2023. Hybrid optimization assisted green power allocation model for QoS-driven energy-efficiency in 5G networks. *Cybernetics and Systems* **56(5)**:1–16 DOI 10.1080/01969722.2023.2175147.

Yao D, Pan W, Dai Y, Wan Y, Ding X, Yu C, Xi H, Xu Z, Sun L. 2023. FedGKD: towards heterogeneous federated learning via global knowledge distillation. *IEEE Transactions on Computers* **73(1)**:3–17 DOI 10.1109/tc.2023.3315066.

Yuan Y, Shi J, Zhang Z, Chen K, Zhang X, Stoico V, Malavolta I. 2024. The impact of knowledge distillation on the energy consumption and runtime efficiency of NLP models. In: *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. Piscataway: IEEE, 129–133.

Zhu J, Cao J, Saxena D, Jiang S, Ferradi H. 2023. Blockchain-empowered federated learning: challenges, solutions, and future directions. *ACM Computing Surveys* **55(11)**:1–31 DOI 10.1145/3570953.