# CoMS: Collaborative DNN Model Selection for Heterogeneous Edge Computing Systems

Aikun Xu, Zhigang Hu, Xi Li, Bolei Chen, Hui Xiao, Xinyu Zhang, Hao Zheng, *Graduate Student Member, IEEE*, Xianting Feng, Meiguang Zheng, Ping Zhong, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—The accelerated integration of edge computing and artificial intelligence has promoted the rise of edge intelligence, which is regarded as the key to solving the last-mile delivery problem of artificial intelligence technology. Although previous research has made many efforts in this area, they have struggled to serve the scenario containing multiple heterogeneous task requests. This challenge is further exacerbated when the number of terminal devices increases and multiple edge servers are required to collaborate to handle task requests. To this end, this paper proposes a fine-grained Collaborative DNN Model Selection scheme for heterogeneous edge computing systems (CoMS), aiming to promote cooperation between edge servers and achieve more effective model selection. Specifically, we first design a reinforcement learning scheme with real-time dynamic normalization strategy, aiming to accelerate model convergence and improve the efficiency of model selection. Next, we introduce a model selection strategy based on greedy algorithm and an efficient fine-grained collaborative model selection strategy respectively to promote cooperation between edge servers, thereby further achieving a balance between inference accuracy and overhead. Extensive experimental results show that compared with baselines, our CoMS reduces the average trade-off overhead by 4.2% to 12.8% and improves the success ratio by 3% to 16%.**

***Index Terms*—Deep learning, deep reinforcement learning, edge computing, edge intelligence, model selection.**

## I. Introduction

IN RECENT years, vision sensors (*e.g.*, cameras) have been widely used as a type of device for capturing images and videos in fields such as smart factories, traffic management systems, and unmanned driving [1]. The images and videos in the above fields are often processed by deep learning models to ensure production safety and driving safety [2]. Such tasks need to be processed (*i.e.*, inference or prediction) by deep learning models (*e.g.*, Deep Neural Network, DNN) are called Deep Learning Tasks (DLT) [3], and are also computation-intensive and resource-intensive tasks [4]. Effectively handling the DLT can improve air quality and employee safety as well as help reduce investment in human and material resources [5]. However, when faced with numerous DLTs, resource-constrained sensors (*i.e.*, terminal devices) may be unable to maintain normal operation (*e.g.*, not able to perform shooting operations) due to frequent calls to the DNN model. This problem can be solved by offloading technology in edge computing to transfer tasks to edge servers with rich resources and closer to users [6], [7].

As an emerging computing paradigm, edge computing can achieve response in a shorter time due to its low-latency characteristics [8], [9]. The convergence of artificial intelligence and edge computing has given rise to edge intelligence, which provides an effective solution for efficiently processing DLTs [10], [11]. Thanks to the advantages of edge computing, the deployed DNN model can perform low-latency inference at the edge of the network. Nevertheless, edge nodes are resource-constrained (*e.g.*, weak CPU or GPU), which becomes a performance bottleneck when facing state-of-the-art DNN models such as YOLO [12], SSD [13], and R-FCN [14].

To address the above issues, researchers have proposed many model compression methods to promote the matching of DNN models with resource-constrained edge nodes [15]. Specifically, by applying knowledge distillation [16], model weight pruning [17], quantization [18], *etc.*, to compress DNN models. The updated model has fewer weights, smaller model size, and consume less resource. However, model compression harms the inference accuracy of the model [19]. Therefore, we should not blindly compress the model but flexibly select appropriate inference models for edge nodes with different resources. The latter is defined as model selection, which can serve inference requests in a resource-efficient manner [20]. Specifically, when edge system resources are limited, the most appropriate model can be selected for arriving inference requests and the inference accuracy can be improved without violating resource constraints.

Given the advantages of the model selection strategy, researchers have carried out some important research work. For example, Romero et al. [21] developed a distributed inference service system that can select appropriate model variants according to queries with different service level goals (*e.g.*, latency

Received 19 May 2024; revised 30 July 2024; accepted 24 September 2024. Date of publication 27 September 2024; date of current version 14 February 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62172442 and Grant 62272489, in part by the National Science Foundation of Hunan Province under Grant 2022JJ30760, and in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 512340030. The review of this article was coordinated by Dr. Wei Quan. *(Corresponding authors: Zhigang Hu; Xi Li.)*

Aikun Xu, Zhigang Hu, Xi Li, Bolei Chen, Hui Xiao, Xinyu Zhang, Hao Zheng, Xianting Feng, Meiguang Zheng, and Ping Zhong are with the School of Computer Science and Engineering, Central South University, Changsha 410073, China (e-mail: aikunxu@csu.edu.cn; zghu@csu.edu.cn; lixi@csu.edu.cn; boleichen@csu.edu.cn; huixiao@csu.edu.cn; zhangxinyu11014@163.com; zhenghao@csu.edu.cn; xiantingfeng@csu.edu.cn; zhengmeiguang@csu.edu.cn; ping.zhong@csu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TVT.2024.3469281

requirements). Wang et al. [22] simultaneously optimized video configuration and network bandwidth allocation with Lyapunov technology to achieve a trade-off between accuracy, latency, and energy consumption. Gao et al. [23] formulated the model selection problem as an online optimization problem with long-term constraints, whose goal is to minimize the total cost. However, model selection becomes challenging for the following reasons: (1) For arrival inference requests, they can be processed by multiple DNN models. For example, YOLO, SSD, and R-FCN can be used to process inference requests for target detection tasks. To handle the same task, different DNN models require resources and provide accuracy may be different. (2) We can reduce resource overhead by sharing the same DNN model instance to handle different types of inference requests (*e.g.*, traffic analysis applications such as object counting and collision analysis). (3) As the number of terminal devices increases, the limited number of edge servers and the lack of good cooperation among edge servers lead to a decrease in inference accuracy and an increase in edge server operational overhead, which impairs the quality of service and user experience. To solve the above challenges, this paper proposes a **Co**llaborative **M**odel **S**election strategy for heterogeneous edge computing systems based on deep reinforcement learning and greedy algorithms (**CoMS**), aiming to select the most appropriate model for each DLT, determine the number of instances of each DNN model, and promote reasonable collaboration between edge servers, thereby trade-off the inference accuracy and overhead of the edge computing system. In particular, the main contributions of this paper are as follows:

- We propose CoMS, a scheme for collaborative DNN model selection in heterogeneous edge computing systems. It promotes cooperation between edge servers and enables more efficient model selection, aiming to trade-off the inference accuracy and system overhead (*i.e.*, increase the inference accuracy while reducing the overhead of edge systems as much as possible), thus improving the quality of service and user experience.
- In the independent model selection stage, a reinforcement learning scheme via Proximal Policy Optimization (PPO) based on real-time dynamic normalization strategy is proposed, which aims to alleviate the training instability problem caused by sudden increases in DLTs and accelerate the model convergence, thereby enhancing the efficiency of model selection.
- In the collaborative model selection stage, a model selection strategy based on greedy algorithm is designed to prevent discarding some DLTs due to edge server overload. Moreover, an efficient fine-grained collaborative model selection strategy is proposed to promote cooperation between edge servers, aiming to further achieve a trade-off between inference accuracy loss and overhead.
- We verified CoMS through extensive experiments on two large datasets. The results show that CoMS outperforms various baselines in terms of the average trade-off overhead and the success ratio.

The paper is organized as follows. Section II provides a review of related work. Section III describes the system model and problem formulation. Section IV introduces the CoMS. The proposed solution is evaluated in Section V. This paper is concluded in Section VI.

## II. Related Works

With the widespread application of vision sensors (*e.g.*, cameras), people can easily access large amounts of image and video data. Effective processing of the above visual information with the help of deep learning techniques can drive the development of such fields as product quality inspection [24], factory dangerous behavior detection [25], and traffic condition detection [26]. These tasks that rely on deep learning models for processing (*i.e.*, inference or prediction) are known as Deep Learning Tasks (DLTs), and they are both also computation-intensive and resource-intensive tasks [27]. In general, whether the system can effectively handle DLTs determines whether the system can guarantee the quality of service, user experience, and operator's revenue. Therefore, it is crucial to study the scheduling problems (*i.e.*, the offloading optimization problem in this paper) for DLTs in edge computing systems.

Compared with processing traditional task requests, handling DLT not only relies on hardware (*e.g.*, end devices, edge servers, or cloud servers, *etc*.), but also requires the support of various DNN models [28], *e.g.*, R-FCN, ResNet [29] and YOLO, *etc*. With the accelerated integration of artificial intelligence and edge computing, edge intelligence has achieved unprecedented development [30] and is expected to become a key solution for artificial intelligence in the 'last mile' problem [31]. Edge intelligence inherits the advantages of edge computing, allowing DNN models to process (*i.e.*, inference or prediction) DLT with lower latency at the edge of the network closer to the user. Nevertheless, the limitations of edge server resources (*e.g.*, weak CPU or GPU) may become a bottleneck in system performance when faced with some advanced DNN models (usually containing more parameters) with higher resource requirements.

To address the resource constraints of edge device, researchers have attempted to compress existing DNN models with techniques such as knowledge distillation [32], model weight pruning [33], and quantification [34] to reduce the number of parameters and the size of DNN model, thus to reduces resource consumption of systems. However, the above solution may have a negative impact on the inference accuracy of model [19]. It can be seen that blind compressing the DNN model is not desirable, but appropriate inference models should be flexibly selected based on edge servers with different resources. The latter is called model selection, which is a strategy for task inference without changing the accuracy of the DNN model [35]. However, balancing the inference accuracy and server operation overhead brought by different models, selecting the most appropriate model for each task, and determining the number of instances of each model remains a challenging problem of model selection in edge computing systems.

To obtain a reasonable model selection solution, researchers have made many efforts. For example, Lu et al. [36] employed a contextual multi-armed bandit machine framework to achieve optimal model selection decisions. Yang et al. [37]

simultaneously optimized video configuration and edge server resource allocation through deep reinforcement learning, aiming to ensure the real-time delay requirements of various video streams while maximizing long-term inference accuracy. Crankshaw et al. [38] proposed an inference service system with two-layer framework. The system combines model selection and model abstraction to achieve low latency, high throughput, and high accuracy. However, the above solutions are difficult to apply to multi-user scenarios.

To this end, Zhang et al. [39] formulated an optimization problem to maximize the average inference accuracy, and solved it with the Markov decision process. Wu et al. [40] studied collaborative DNN inference between multi-terminal devices and servers, and jointly optimized task sampling rate selection, task offloading, and resource allocation through a deep reinforcement learning scheme, which aims to minimize service latency while guaranteeing average accuracy requirements. Huang et al. [41] provided latency guarantees for inference tasks by adaptively adjusting the compression ratio of task data. Zhao et al. [4] explored how to serve multiple artificial intelligence applications and models at the edge of the network, and the proposed EdgeAdaptor achieves a trade-off between inference accuracy, latency, and resource consumption.

Nevertheless, as the number of terminals and edge servers increases, the above solutions will fail. The reason is that they do not consider the cooperation between edge servers, failing to respond to task requests on time, which damages the quality of service and user experience. To this end, we propose a collaborative DNN model selection scheme based on adaptive strategy that aims to promote cooperation among edge servers, full utilize system resources, reduce the edge server operational overhead and improve the inference accuracy of the edge computing system, thereby enhancing the quality of service and user experience.

## III. System Model and Problem Formulation

In this section, we first introduce the system model of the heterogeneous edge computing systems. Then, we give an explanation of the performance metrics. Finally, we describe the problem formulation.

### A. System Model

This paper explores using multiple edge servers to provide inference services for heterogeneous Deep Learning Tasks (DLTs). An example of heterogeneous DLT inference in Edge Computing-assisted Transportation Systems (ECTS) is shown in Fig. 1. This example contains $N$ IoT sensors (*e.g.*, cameras), $J$ edge servers, and $L$ base stations. The set of IoT sensors is denoted as $S = \{s_0, s_1, \ldots, s_{N-1}\}$, which are often used to capture image data of pedestrians, buses, cars, motorcycles, *etc*. The set of edge servers is denoted as $E = \{e_0, e_1, \ldots, e_{J-1}\}$, where each server is pre-deployed with different types of DNN models for processing image data (*i.e.*, task requests) captured by IoT sensors. The set of DNN deployed on the $j$th edge server is represented by $D_j = \{d_{j0}, d_{j1}, \ldots, d_{j(K-1)}\}$ and their capacities are $Cap_j = \{cap_{j0}, cap_{j1}, \ldots, cap_{j(K-1)}\}$, where $K$

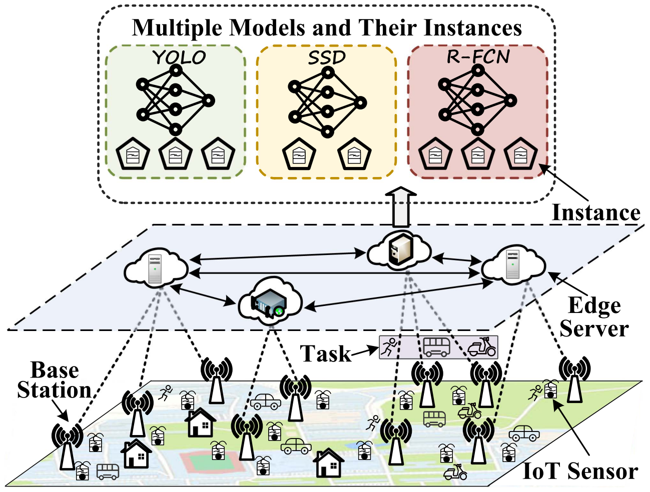


Fig. 1. An example of model selection in edge computing-assisted transportation systems. In brief, the IoT sensors in the example are used to collect various types of task requests and transmit them to appropriate edge servers through base stations for processing. However, there are different DNN instances in each edge server and the revenue obtained when processing task requests using different instances is different. To improve the overall revenue of the system, it is necessary to adapt the best DNN instance to the arrival task request.

is the number of DNN types. We define $A_i^t$ as the number of $i$th task requests in the $t$th time slot, where $i \in I$, $I$ is the number of task types. Base stations are used to receive image data (*i.e.*, task requests) captured by IoT sensors and send them to edge servers. In this scenario, it is challenging to select an appropriate DNN model for handling different tasks. As the number of IoT sensors increases and collaborative processing tasks between multiple edge servers are required, this challenge is further exacerbated.

### B. Performance Metrics

*Edge server operational overhead:* This paper considers the operational overhead incurred by running DNN model instances on edge servers, which is determined by both energy consumption and response costs [42]. Here, $c_k$ represents the unit cost of running one instance of the $k$th DNN model. The system's operational overhead is the total accumulated overhead while processing multiple tasks. Thus, this overhead depends on the number of instances of the DNN model and the corresponding unit cost. To illustrate more specifically, the total operational overhead in the $t$th time slot is defined as follows:

$$U_E^t = \sum_{j\in J}\sum_{k\in K} c_k^t y_{jk}^t, \tag{1}$$

where $y_{jk}^t$ represents the number of the $k$th DNN's instances on the $j$th edge server in the $t$th time slot.

*Inference accuracy loss:* There are usually differences in the inference accuracy obtained by processing tasks with different DNN models. To accurately describe this case, $a_{ik}$ represents the inference accuracy achieved by the $i$th task under the $k$th model. Meanwhile, $x_{ijk}$ is set as the decision variable for model selection, where $x_{ijk} \neq 0$ means that the $i$th task is processed by the $k$th model instance on the $j$th edge server. Thus, the total

TABLE I
THE SUMMARY OF NOTATIONS

| Notation | Description |
|---|---|
| $N$ | Number of IoT sensors |
| $J$ | Number of edge servers |
| $L$ | Number of base stations |
| $K$ | Number of DNN types |
| $I$ | Number of task types |
| $S$ | The set of IoT sensors |
| $E$ | The set of edge servers |
| $A_i^t$ | Number of $i$th task requests in the $t$th time slot |
| $D_j$ | The set of DNN deployed on the $j$th edge server |
| $Cap_j$ | The set of DNN's capacities on the $j$th edge server |
| $c_k$ | The unit cost of running one instance of the $k$th DNN model |
| $y_{jk}$ | The number of the $k$th DNN's instances on the $j$th edge server. |
| $U_E^t$ | The total operational overhead in the $t$th time slot |
| $x_{ijk}$ | The decision variable for model selection |
| $a_{ik}$ | The inference accuracy of processing the $i$th task request through the $k$th model |
| $U_A^t$ | The total inference accuracy loss for all task requests in the $t$th time slot |

inference accuracy loss for all tasks in the $t$th time slot is defined as:

$$U_A^t = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(1-a_{ik})x_{ijk}^t, \tag{2}$$

where $a_{ik}$ is the inference accuracy of processing the $i$th task request through the $k$th model.

## C. Problem Formulation

To improve the overall performance of edge inference systems, our work aims to jointly optimize the edge server operational overhead and inference accuracy loss. This problem is defined as follows:

$$\mathbf{P1}: \min \sum_{t=0}^{T} U_E^t + U_A^t \tag{3}$$

subject to:

$$\sum\nolimits_{i=0}^{I} x_{ijk}^t \cdot A_i^t \le c_k^t y_{jk}^t, \forall c_k \in C, \tag{3a}$$

$$\sum\nolimits_{j=0}^{J}\sum\nolimits_{k=0}^{K} x_{ijk}^t = 1, \forall s_i \in S, \tag{3b}$$

$$x_{ijk}^t \in [0,1], y_{jk}^t \in \mathbb{N}, \forall s_i \in S, \forall c_j, c_k \in C, \tag{3c}$$

where the (3a) aims to ensure that the number of task requests processed by the edge server will not exceed the sum of all task inference requests received. Equation (3b) guarantees that all inference requests reach the destination edge server. Equation (3c) specifies the value range of the decision variable and ensures that the number of running DNN model instances is an integer value. A summary of the key notations is provided in Table I.

## IV. CoMS Overview

**P1** can be regarded as the model selection problem, which is difficult to estimate accurately in the long-term optimization process. In particular, the difficulty of solving this problem increases significantly when the number of end devices is large and multiple edge servers are required to be introduced for collaborative processing DLTs. Given the complexity of the above problem, this paper considers splitting it into an independent model selection problem and a collaborative model selection problem. Meanwhile, we propose a collaborative DNN model selection scheme based on deep reinforcement learning and greedy algorithms (CoMS) to solve them, aiming to achieve a reasonable trade-off between the inference accuracy loss and edge server operational overhead. CoMS mainly consists of independent model selection and collaborative model selection stages. In the former, to meet the real-time requirements, we introduce a deep reinforcement learning-based scheme to adaptively and quickly obtain the preliminary allocation result for task requests. In the latter, we focus on the reasonable use of idle resources and design a greedy algorithm-based scheme to achieve accurate matching between task requests and DNN instances, thereby further improving the revenue of the system. The framework of CoMS is shown in Fig. 2.

## A. Independent Model Selection

In the independent model selection phase, we focus on each edge server independently matching appropriate DNN instances for arriving task requests. Specifically, this involves determining which DNN model instances in a given edge server should handle which arriving task requests, aiming to achieve a balance between inference accuracy and operational overhead initially. This problem can be reduced to the multi-knapsack problem. The proof is as follows.

*Theorem 1:* The model selection problem **P1** is NP-hard.

*Proof:* We prove the NP-hardness of **P1** by reducing from the Multiple Knapsack Problem (MKP), which is known to be NP-hard [43]. Given $N$ items and $M$ knapsacks, each item $i$ has a weight $w_i$ and a value $v_i$. Each knapsack $j$ has a maximum capacity $W_j$. The MKP selects suitable items put into knapsacks, ensuring that the total capacity of each knapsack does not exceed its maximum capacity while maximizing the total value of the items in all knapsacks. In **P1**, the task' type $i$ corresponds to the item $i$ in the MKP, the amount of a certain task request $A_i^t$ corresponds to the item weight $w_i$ in the MKP, the sum of inference accuracy loss and overhead $\sum_{t=0}^{T} U_E^t + U_A^t$ represents the value of the item in the MKP and $cap_{jk}$ corresponds to the maximum capacity $W_j$ in each backpack in the MKP. It can be seen that the solution to **P1** is also the optimal solution to the MKP. Therefore, **P1** is an NP-hard problem.

The emergence of deep reinforcement learning provides new ideas for solving NP-hard problems [44]. Meanwhile, Proximal Policy Optimization (PPO) has outstanding performance in many deep reinforcement learning schemes [45]. To this end, we employ the PPO reinforcement learning algorithm to solve **P1**. However, when the number of task requests fluctuates greatly, if they are direct input into the traditional PPO reinforcement learning scheme will bring the following adverse effects: 1) increasing the complexity of the state space in PPO, making the learning process more difficult, 2) hurting the training stability, especially at the early stages of training, the model may have difficulty in adapting to extreme or uncommon states, resulting

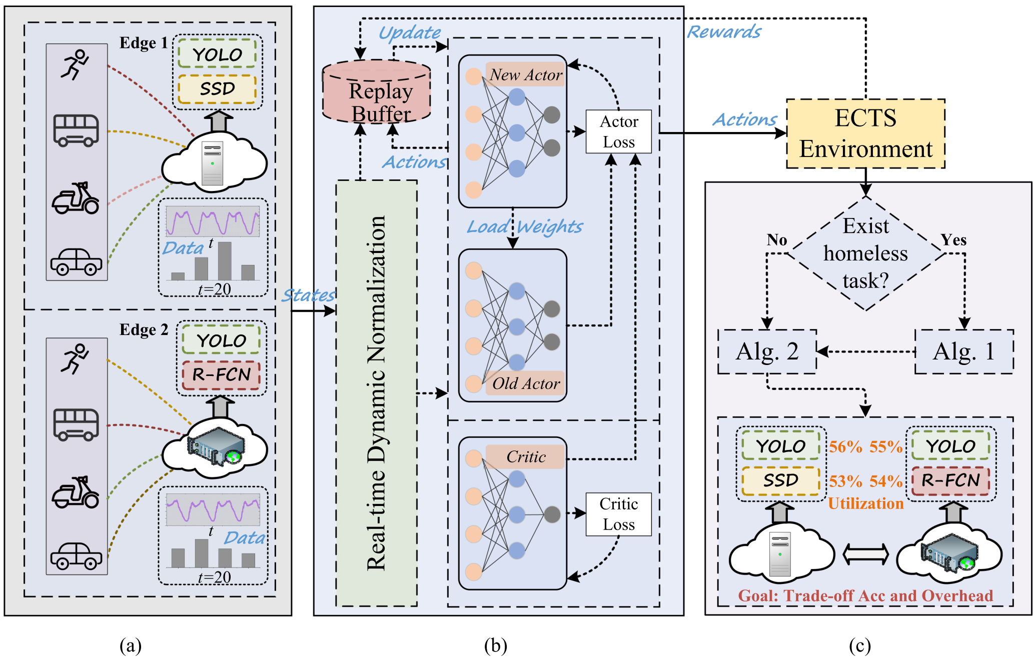


Fig. 2. CoMS overview. In (a), different edge servers will receive different task requests (*e.g.*, pedestrian pictures, bus pictures, bicycle pictures, and car pictures). In general, the number of different types of task requests usually differs. Meanwhile, the number of task requests received by different edge servers is also different and changes over time. The above information is usually used as the environment state as input for reinforcement learning schemes. In (b), the real-time dynamic normalization strategy is first used to preprocess the environmental state information. Next, the PPO reinforcement learning algorithm is used to infer this information to obtain the strategy (*i.e.*, action) of model selection. Finally, PPO feedback on this action to the ECTS environment. In (c), we need to determine whether there are 'homeless tasks' after the ECTS environment accepts the action from (b). If it exists, we perform Algorithm 1 to prevent the edge server from dropping task requests due to overload. Otherwise, we will directly execute Algorithm 2 and be guided by utilization, achieve collaboration between edge servers, and trade-off the inference accuracy loss and edge server operational overhead in the edge computing system. (a) Real-time dynamic heterogeneous tasks. (b) First stage: independent model selection. (c) Second stage: collaborative model selection.

in a slow or unstable learning process. The common way to overcome the above problems is to normalize the data. However, unlike traditional scenarios, the task requests processed in our work are generated in real-time and their number fluctuates greatly, which is not suitable to be processed by traditional normalization algorithms. To this end, we design a real-time dynamic normalization strategy, aiming to accelerate the convergence of PPO and enhance its generalization ability.

*1) Real-Time Dynamic Normalization Strategy:* Assuming that $X$ is the input raw data, we can initialize it with (4), where $X$ is used to represent the sequence consisting of the number of various task requests assigned to each edge server and the DNN instance state in the edge server.

$$X = 2 \cdot \frac{X - minValue}{maxValue - minVale} - 1, \tag{4}$$

where $minValue$ and $maxValue$ are the minimum value and maximum value of the input raw data respectively. They can be updated via (5) and (6),

$$minValue = \min(minValue, \min(X)), \tag{5}$$

$$maxValue = \max(maxValue, \max(X)), \tag{6}$$

where the initial value of $minValue$ is the maximum number, and the initial value of $maxValue$ is the negative of $minValue$.

*2) Model Selection Scheme Based on PPO Reinforcement Learning:* After obtaining the normalized input, we introduce the PPO deep reinforcement learning algorithm to determine which task request is processed by which DNN model. Next, we will describe the scheme in detail.

- *PPO framework:* PPO belongs to a category of Actor-Critic algorithm, which has superior performance and is expected to solve problem **P1**. The Actor module of PPO consists of an input layer, a hidden layer, and an output layer. The dimension of the input layer is the product of the number of task types and the number of DNN types, plus the number of DNN types, *i.e.*, $I \times K + K$. The dimension of the output layer is the product of the number of task types and the number of DNN types, *i.e.*, $I \times K$. The input and output dimensions of the hidden layer are both 64. The Critic module has a similar structure to the Actor module. The difference is that the output dimension of Critic is 1.
- *State space and action space:* The PPO algorithm needs to reasonably assign task requests to appropriate DNN instances for processing. In this paper, we draw on the idea of the partial offloading strategy and consider assigning each task to each DNN instance in proportion. At the same time, the state information of each DNN instance in the edge server will be changed accordingly. Therefore, we can represent the above two kinds of information as the environmental state, then the state space is $I \times K + K$, which contains the normalized value of the number of task requests on each DNN instance (*i.e.*, $I \times K$) and the state information of the DNN instance (*i.e.*, $K$).

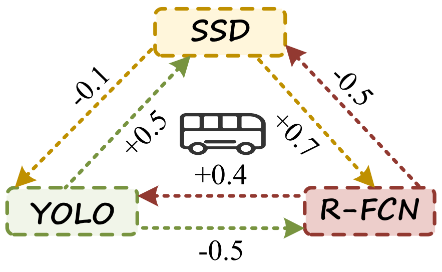


Fig. 3. An example of bus picture task requests transferring between different DNN instances.

The action space is $\{-1,1\} \times (I \times K)$, which represents the proportion of each DNN instance transferring a certain type of task request to other DNN instances, where the positive and negative signs indicate the direction of transfer. Fig. 3 gives an example of bus picture task requests transferring between different DNN instances. To illustrate with the SSD instance example, if the action generated by the Actor is -0.1 (yellow dotted line), it means that 10% of the bus picture task requests in the SSD instance will be transferred to the YOLO instance for processing. If the action generated by the Actor is +0.7 (yellow dotted line), it means that 70% of the bus picture task requests in the SSD instance will be transferred to the R-FCN instance for processing. The remaining part will be processed locally.

- *Reward shaping:* In the PPO algorithm, the agent will receive a certain reward after performing an action. The goal is to optimize the cumulative reward through training to obtain the proper DNN model selection solution. The reward obtained in the $t$th time slot can be expressed as,

$$reward = -\frac{U_E^t + U_A^t}{\sum_{i=0}^{I} A_i^t}, \tag{7}$$

where $reward$ is defined as the sum of the average edge server operational overhead and average inference accuracy loss incurred by processing each task request in the $t$th time slot. Considering that the goal of deep reinforcement learning is to maximize the reward function, while our work aims to minimize the objective function, the value of (7) is converted into a negative value. Through the long-term interactive adjustment between the state, action, and reward function, the agent will gradually learn and adapt to the current scenario, and thus be able to solve the problem **P1** by reasonably assigning DNN model instances for arriving task requests.

## B. Collaborative Model Selection

The solution obtained during the independent model selection phase in Section IV-A is only a preliminary result. This is because when faced with a large number of task requests, this solution may cause some edge servers to be overloaded, or even be unable to handle all arriving task requests, damaging the quality of service and user experience. To this end, the collaborative model selection phase aims to achieve a more fine-grained model selection strategy. The core idea is to decide what strategy based on with/without 'homeless task' (*i.e.*, task requests that cannot be processed by the current edge servers in the $t$ th time slot) to adopt to maximize the overall performance of the system and improve the quality of service and user experience. Next, we will introduce the designed algorithm in detail.

*1) Task Schedule Strategy Based on Greedy Algorithm:* Faced with 'homeless tasks', previous solutions usually drop them, making it difficult to ensure high quality of service and user experience. To solve this problem, we propose a model selection strategy based on greedy algorithm, as shown in Algorithm 1. The key to this strategy is to comprehensively consider the task request allocation of DNN instances in all edge servers and forward 'homeless task' to DNN instances that still have processing capability and better performance, thereby improving the quality of service and user experience.

Algorithm 1 accepts the current server state information $CurSS$, the dependency matrix of task $TD$, the number of task type $TT$, the number of DNN type $DT$, the number of server $ST$, the current server id $CurSID$, and the utilization of the DNN instance $U$ and the queue of homeless task $HTQ$ as input, aiming to output the updated current server state information $CurSS$. In Algorithm 1, it is necessary to traverse the number of each task in $HTQ$ (Line 1 to Line 30). The decision to exit the loop depends on whether all tasks in $HTQ$ have been processed (Line 2 to Line 4). Then, each 'homeless tas' on the current server is greedily forwarded to another edge server for processing based on the utilization ratio (Line 5 to Line 29). However, when the amount of tasks that need to be processed exceeds the processing capacity of the selected edge server: 1) To ensure higher inference accuracy, we prioritize using the same DNN instance on other edge servers to process the remaining tasks without damaging inference accuracy, as shown in Fig. 4(a). 2) Suppose the selected DNN instance on any server cannot handle the remaining tasks. In that case, we again consider selecting the model instance with the smallest utilization to handle the remaining tasks to ensure that the edge server is not overloaded, as shown in Fig. 4(b).

The main operations of Algorithm 1 include 4 layers of loop calculations, located in Line 1, Line 5, Line 7, and Line 9 respectively. In the worst case, the loop calculation in Line 1 requires iterating $TT$ rounds and its time complexity is $O(TT)$. The loop calculation in Line 5 needs to iterate $DT$ rounds and its time complexity is $O(DT)$. The loop calculation in Line 9 needs to iterate $ST$ rounds and its time complexity is $O(ST)$. Unlike them, the loop calculation in Line 7 depends on $UFlag$. We assume that in the worst case, the maximum number of executions of this loop is $Z$, so its time complexity is $O(Z)$. Therefore, the time complexity of Algorithm 1 is $O(TT * DT * ST * Z)$. In fact, $TT$, $DT$, and $ST$ are usually less than 10. $Z$ is mainly affected by $HTQ$, which counts the categories and number of 'homeless tasks'. In general, 'homeless tasks' account for only a small part, so the value of $HTQ$ is relatively small. To summarize, although the time complexity of Algorithm 1 is $O(TT * DT * ST * Z)$, it is still within the acceptable range.

*2) Fine-Grained Collaborative DNN Model Selection Strategy:* Without 'homeless tasks', edge servers will not be overloaded. Nevertheless, the utilization of each server is not

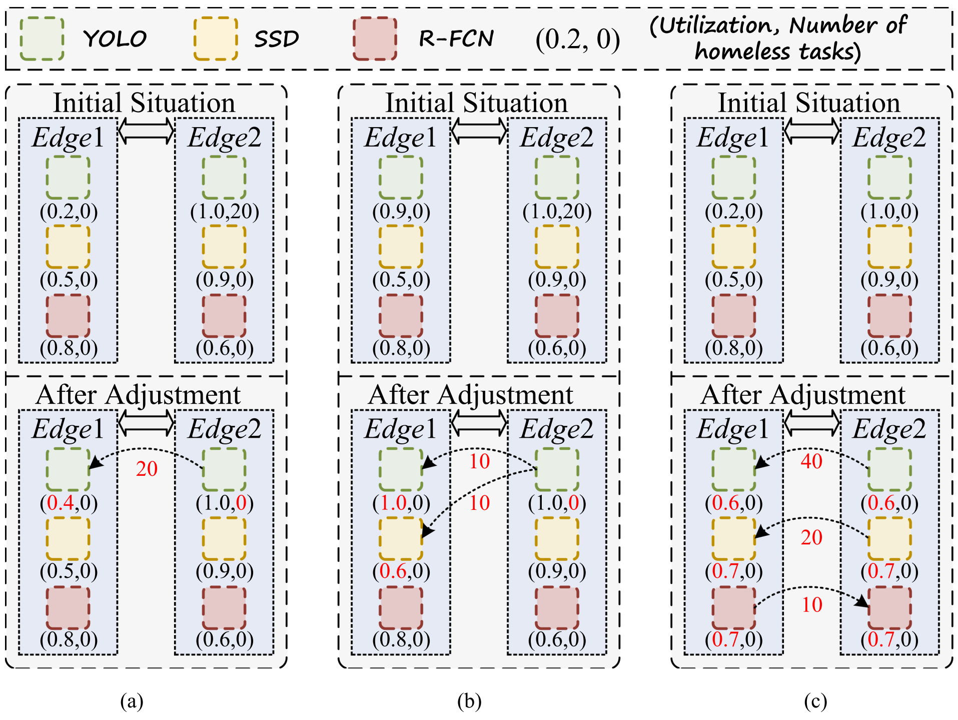


Fig. 4. The example of Algorithm 1 and Algorithm 2 in Fig. 2(c). In this example, Edge 1 and Edge 2 communicate with each other and the capacity of each class of model instance is 100. Specifically, in (a), there are 20 homeless tasks in the YOLO instance in Edge 2. We prioritize forwarding homeless tasks to the same instance in Edge 1 (*i.e.*, YOLO) for processing without compromising the system's inference accuracy. In (b), the remaining capacity of the YOLO instance in Edge 1 alone cannot handle all the tasks from the YOLO instance in Edge 2. To avoid the adverse effects of model instance overload, we select the model instance with the smallest utilization in Edge 1 (*i.e.*, SSD) to process the remaining homeless tasks. In (c), considering that unbalanced server utilization will lead to a waste of system resources and affect the quality of service and user experience, we introduce the utilization of each class of model instance as a guide to adjust the distribution of tasks again. (a) Alg. 1(Case1). (b) Alg. 1(Case2). (c) Alg. 2.

balanced and there are even idle servers, resulting in a waste of system resources. Considering that the utilization of each edge server directly affects its processing performance, especially under high load, the processing performance of the server will be reduced accordingly, resulting in a significant increase in the time required to process the same task, thereby affecting the quality of service and user experience. To this end, we propose an efficient fine-grained collaborative model selection strategy, which employs the utilization of each class of DNN instances as a guide to adaptively adjusting the task allocation of DNN instances, thereby increasing the overall benefits of the system, as shown in Algorithm 2 and Fig. 4(c).

Algorithm 2 takes the state information of all servers $SS$, the number of task types $TT$, the number of DNN types $DT$, the number of servers $ST$, the DNN instance capacity $DCap$ and the maximum number of DNNs in the edge server $MaxNumDIS$ as input, and aims to output the number of DNN instances deployed in each edge server. In Algorithm 2, we first count the sum of tasks existing on DNN instances in each edge server (Line 1 to Line 3). Next, according to the sum of tasks in all edge servers on different DNNs divided by the capacity of the corresponding DNN instances, the number of different DNN instances required can be obtained (Line 4). To balance the load between edge servers, we hope the number of DNN instances started is reasonable in the corresponding edge server. To this end, we obtain the number of DNNs required to be deployed by dividing the maximum number of DNNs deployed on each edge server by the sum of DNNs and multiplying by the sum of DNN instances required (Line 5). However, the above solution is approximately optimal and may not always be sufficient to handle all tasks. Therefore, we need to obtain the number of DNNs that still require deployment (Line 6). Even if the corresponding number of DNNs is obtained, it is still a challenge to deploy them on which edge server. To quickly solve this problem, we use utilization as a guide to greedy finding edge servers suitable for deploying the added DNN instances, thereby achieving the purpose of saving overhead (Line 8 to Line 21).

The main operation of Algorithm 2 is loop calculation, where Line 1 to Line 3 is the first loop and its time complexity is $O(ST)$. Line 8 to Line 21 is the second loop calculation, and its computational complexity is $O(DT)$. Line 9 to Line 20 is the third loop calculation, whose stopping condition is that $RemainD[j]$ is empty. Assuming the maximum value of $RemainD[j]$ is $Z$, the time complexity of the third loop calculation is $O(Z)$. Line 12 to Line 16 is the fourth loop calculation, and its time complexity is $O(ST)$. Therefore, the time complexity of Algorithm 2 is $O(ST + DT * Z * ST)$. In fact, $ST$ and $DT$ are usually less than 10. And $Z$ usually represents the DNN that requires additional startup, so its value is usually smaller. In summary, although the time complexity of Algorithm 2 is $O(ST + DT * Z * ST)$, it is still within the acceptable range.

**Algorithm 1:** Task Schedule Strategy Based on Greedy Algorithm.

```
Input : The current server state information $CurSS$;
        The dependency matrix of task $TD$;
        The number of task type $TT$;
        The number of DNN type $DT$;
        The number of server $ST$;
        The current server id $CurSID$;
        The utilization of DNN $U$;
        The queue of homeless task $HTQ$;
Output: The current server state information $CurSS$.
for $i$ in $(0, TT)$ do
    if $sum(HTQ) = 0$ then
        break
    end
    for $j$ in $(0, DT)$ do
        $UFlag \leftarrow True$, $IFlag \leftarrow True$;
        while $UFlag$ do
            $MinU \leftarrow MaxValue$, $MinID \leftarrow 0$;
            for $k$ in $(0, ST)$ do
                if $CurSID \neq k$ then
                    if $U[k][TD[i][j]] \leq MinU$ then
                        $MinU \leftarrow U[k][TD[i][j]]$;
                        $MinID \leftarrow k$;
                    end
                end
            end
            if $abs(1 - MinU) < MinValue$ then
                $UFlag \leftarrow False$
            else
                Update $CurSS, HTQ, U$;
                if $HTQ[i] = 0$ then
                    $UFlag \leftarrow False$, $IFlag \leftarrow False$;
                end
            end
        end
        if $IFlag = False$ then
            break
        end
    end
end
return $CurSS$
```

**Algorithm 2:** An Efficient Fine-Grained Collaboration DNN Model Selection Strategy.

```
Input : The state information of all servers $SS$;
        The number of task type $TT$;
        The number of DNN type $DT$;
        The number of server $ST$;
        The DNN instance capacity $Cap$;
        The maximum number of DNNs in the edge server
        $MaxNumDIS$;
Output: The number of DNNs in the server
        $NumDIS$.
for $i$ in $(0, ST)$ do
    $SumTID[i] \leftarrow sum(SS[i].reshape(TT, DT))$;
end
$NeedTD = \lceil sum(SumTID)/Cap \rceil$;
$NumDIS = \lfloor NeedTD * (\frac{MaxNumDIS}{sum(MaxNumDIS)}) \rfloor$;
$RemainD = NeedTD - sum(NumDIS)$;
if $sum(RemainD) \neq 0$ then
    for $j$ in $(0, DT)$ do
        while $RemainD[j]$ do
            $MinU \leftarrow 1$;
            $MinIndex \leftarrow 0$;
            for $k$ in $(0, ST)$ do
                if $\frac{NumDIS[k][j]}{MaxNumDIS[k][j]} \leq MinU$ then
                    $MinU = \frac{NumDIS[k][j]}{MaxNumDIS[k][j]}$;
                    $MinIndex = k$;
                end
            end
            $RemainD[j] - -$;
            $NumDIS[MinIndex][j] + +$;
        end
    end
end
return $NumDIS$;
```

## V. Experiments

In this section, we describe the experimental setup and results. The results obtained with CoMS are compared with three baseline algorithms for model selection in edge computing. The Python program implementation of CoMS is available in [46].

### A. Experimental Setup

*Dataset:* We validate CoMS on two real-world datasets, *i.e.*, PeMSD4 and PeMSD8. These datasets are widely used in experimental tests in the transportation field (*e.g.*, Internet of Vehicles) [49]. PeMSD4 contains traffic data from 307 sensor nodes in the San Francisco Bay Area, and the data spans January to February 2018. PeMSD8 collected traffic data from 170 sensors in San Bernardino from July to August 2016. In the simulation experiment, the sensors included in the above two datasets can be regarded as end devices, and the collected data can be considered as generated task requests.

*Baseline Algorithms:* This paper demonstrates the advantages of CoMS by comparing CoMS with three baseline algorithms, *i.e.*, EdgeAdaptor [4], the classical PPO [48], and Random [47]. EdgeAdaptor utilizes multiple DNN instances to provide inference services for multiple task requests in a single-server edge computing system. However, it is difficult to perceive environmental changes and cannot benefit other servers. Similarly, classic PPO only focuses on running on a single-server and cannot benefit neighboring servers. The difference is that classic PPO can adapt to the environment and give reasonable solutions through long-term state, action, and reward function interaction. Random means that arriving task requests are randomly sent to any DNN instance for processing, and it is difficult to obtain ideal results.

*Parameter Settings:* All the results are evaluated on Intel(R) Xeon(R) Silver 4112 2.2GHz CPU and 13GB of memory. We assume that the end device generates 5 different types of task

TABLE II
INFERENCE ACCURACY

| DNN \ Task | **people** | **car** | **bus** | **bike** | **dog** |
|---|---|---|---|---|---|
| **YOLOv2** | 0.731 | 0.671 | 0.700 | 0.719 | 0.784 |
| **SSD** | 0.794 | 0.761 | 0.794 | 0.801 | 0.870 |
| **R-FCN** | 0.758 | 0.728 | 0.733 | 0.749 | 0.833 |

TABLE III
THE RESOURCE CAPACITY OF THE EDGE SERVER FOR VARIOUS DNNS

| Dataset | ID \ DNN | **YOLOv2** | **SSD** | **R-FCN** |
|---|---|---|---|---|
| **PeMSD8** | **Edge 0** | 40 | 30 | 20 |
| | **Edge 1** | 39 | 30 | 43 |
| | **Edge 2** | 30 | 50 | 10 |
| | **Edge 3** | 50 | 100 | 20 |
| **PeMSD4** | **Edge 0** | 130 | 270 | 70 |
| | **Edge 1** | 25 | 30 | 43 |
| | **Edge 2** | 10 | 30 | 10 |
| | **Edge 3** | 30 | 10 | 30 |

requests, *i.e.*, $I = 5$. These task requests will be responded to by 3 different DNN models deployed in edge servers, *i.e.*, $K = 3$, where the number of edge servers is $J = 4$. Table II shows the inference accuracy obtained by different DNN models in processing various task requests [4]. The capacity of the different DNN models (*i.e.*, YOLOv2, SSD, and R-FCN) is referenced from [4], *i.e.*, [180, 200, 190], which denotes the maximum amount of task requests that can be processed on the DNN instance in each time slot. Table III shows the maximum number of DNN instances that the edge server can start, *i.e.*, the resource capacity of the edge server for various DNNs. $N$ is determined by the dataset, $N$ is 170in PeMSD8, and $N$ is 307in PeMSD4. The training parameters of CoMS and the classical PPO are both set as follows: the learning rate of the Actor module is 0.0003, the learning rate of the Critic module is 0.001, the discount factor is 0.99, and the random seed is equal to 0. Other parameter settings can be found in [46].

*Evaluation Metrics:* This paper uses the average Inference Accuracy Loss (IAL), the average edge server Operational Overhead (OO), the average Trade-Off Overhead (TOO, trade-off between OO and IAL), and the success ratio (succRatio) as evaluation metrics, which are key metrics for evaluating system performance [20]. Here, the succRatio represents the proportion of task requests successfully processed by the DNN instance.

## B. Results and Analysis

*1) The Convergence of CoMS:* As shown in Fig. 6, we evaluate the convergence of CoMS. The two curves in the figure are the reward functions of the classic PPO algorithm and our proposed scheme CoMS respectively. It can be seen that the reward function of classic PPO is difficult to converge and is correlated with the fluctuation of task requests (Fig. 5). In contrast, the rewards of CoMS gradually increase over time and stabilize after 10,000 time frames. The reason is that the real-time dynamic normalization strategy limits fluctuating task requests to a smaller interval, which alleviates the problem of unstable training and accelerates model convergence.

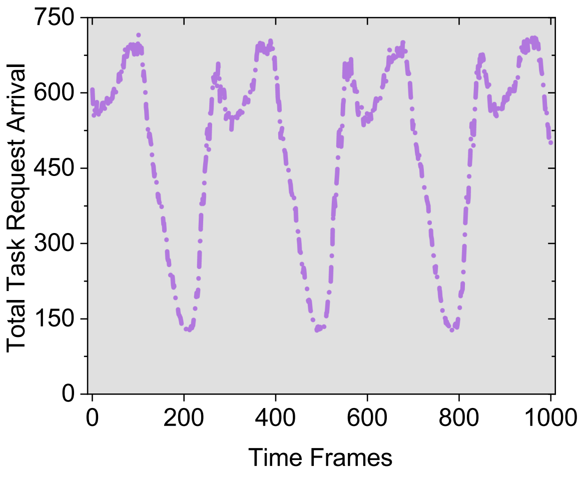


Fig. 5. An example of the total task requests for each time slot in PeMSD8.

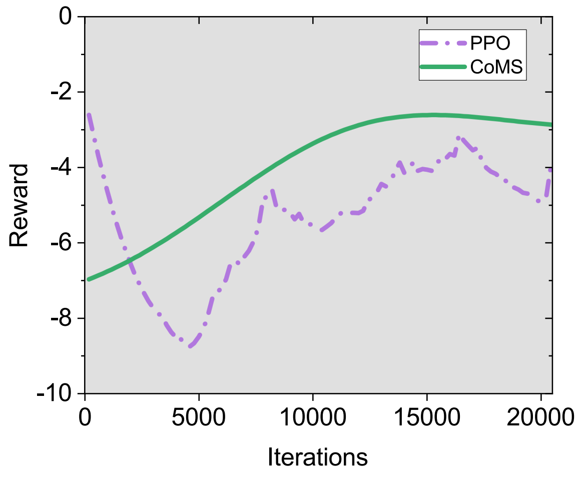
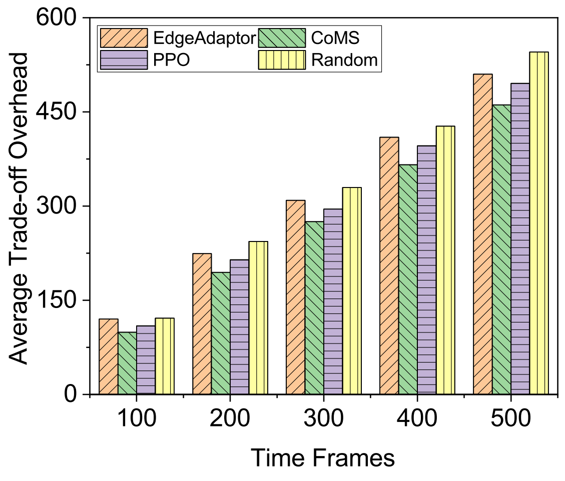


Fig. 6. Comparison of the reward.

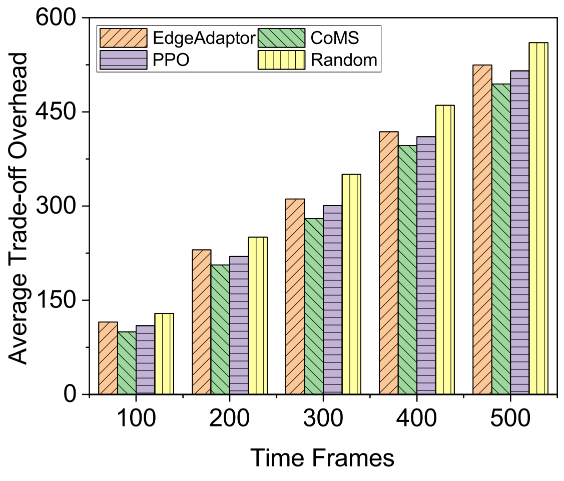
TABLE IV
PERFORMANCE COMPARISON

| Dataset | Methods \ Metrics | **OO** | **IAL** | **TOO** | **succRatio** |
|---|---|---|---|---|---|
| **PeMSD8** | **Random**[47] | 293.12 | 288.38 | 581.50 | 0.86 |
| | **PPO**[48] | 286.07 | 250.87 | 536.94 | 0.90 |
| | **EdgeAdaptor**[4] | 323.45 | **222.39** | 545.84 | 0.95 |
| | **CoMS** | **266.16** | 240.43 | **506.59** | **1** |
| **PeMSD4** | **Random**[47] | 284.87 | 286.56 | 571.43 | 0.84 |
| | **PPO**[48] | 274.09 | 261.52 | 535.61 | 0.91 |
| | **EdgeAdaptor**[4] | 310.43 | **234.09** | 544.52 | 0.97 |
| | **CoMS** | **258.40** | 254.19 | **512.59** | **1** |

*2) Comprehensive Performance Comparison:* We compare the results of our proposed model selection method CoMS with three baseline algorithms including EdgeAdaptor, PPO, and Random, as shown in Table IV. Overall, CoMS is near-optimal on the PeMSD8 and PeMSD4 datasets using all metrics, which demonstrates the advancement of our scheme. Specifically, CoMS achieves TOO and succRatio improvement compared with the state-of-the-art model (*i.e.*, EdgeAdaptor), *i.e.*, TOO is improved by 7.2% and 5.9%, and succRatio is improved by 5.3% and 3.1% on PeMSD8 and PeMSD4, respectively. Furthermore, we can make the following observations: (1) Random has the worst performance. This observation indicates that randomly



(a)



(b)

Fig. 7. Performance comparison of different algorithms under Long-Term continuous constraints. (a) Performance comparison on PeMSD8. (b) Performance comparison on PeMSD4.

TABLE V
THE ABLATION TEST OF COMS

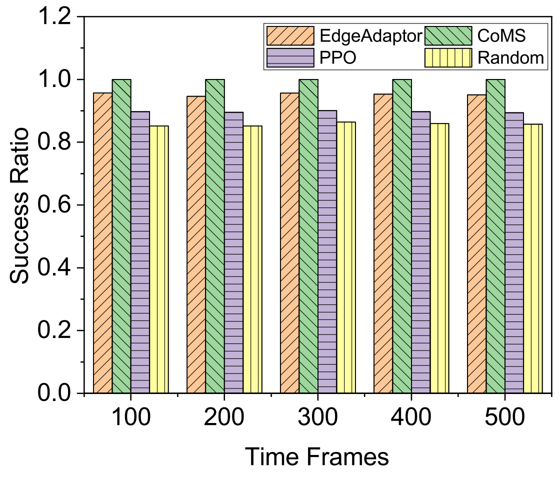
| Dataset | Metrics / Methods | **OO** | **IAL** | **TOO** | **succRatio** |
|---|---|---|---|---|---|
| **PeMSD8** | **CoMS(w/o_norm)** | 274.12 | 251.45 | 525.57 | 1 |
| | **CoMS(w/o_homeless)** | 290.66 | 268.50 | 559.16 | 0.87 |
| | **CoMS(w/o_fine)** | 300.60 | 249.39 | 549.99 | 1 |
| | **CoMS** | **266.16** | **240.43** | **506.59** | **1** |
| **PeMSD4** | **CoMS(w/o_norm)** | 282.50 | 316.794 | 599.59 | 1 |
| | **CoMS(w/o_homeless)** | 265.62 | 265.42 | 531.05 | 0.82 |
| | **CoMS(w/o_fine)** | 281.03 | 254.19 | 535.22 | 1 |
| | **CoMS** | **258.40** | **254.19** | **512.59** | **1** |

matching task requests to DNN instances hardly gives satisfactory results. (2) PPO is a solution that can interact with the environment and adaptively change its own policies. Although it is worse than the state-of-the-art algorithm EdgeAdaptor in terms of IAL and succRatio, its OO and TOO are superior. This result shows that PPO can better achieve a trade-off between edge server operational overhead and accuracy loss. (3) Compared with other solutions, EdgeAdaptor tries its best not to drop task requests while ensuring lower inference accuracy loss, but the edge server operating overhead is too high. The reason is that the lack of collaboration between edge servers causes some edge servers to be overloaded and increases their operational overhead. To summarize, Random is worse than CoMS. This is because CoMS has optimization goals and can guide it in a better direction. CoMS performs better compared to PPO and EdgeAdaptor. This is because CoMS not only realizes collaboration between edge servers, but can also adaptively adjust its own policies. In addition, the introduction of real-time dynamic normalization strategy greatly reduces the impact of CoMS from task fluctuations and accelerates its convergence.

*3) Ablation Test:* To further explore the impact of the fusion of multiple parts in CoMS, we compare CoMS with CoMS(w/o_norm), CoMS(w/o_homeless), and CoMS(w/o_fine), where CoMS(w/o_norm), CoMS(w /o_homeless), and CoMS (w/o_fine) are part of CoMS and are described as completing model selection without considering normalization, 'homeless tas' judgment, and fine-grained collaboration respectively. As shown in Table V, CoMS performs better than CoMS(w/o_norm), CoMS(w/o_homeless), and CoMS(w/o_fine). Specifically, CoMS (w/o_norm) performs worse than CoMS. This is because when the real-time dynamic normalization strategy is not used, the fluctuating task request volume will reduce the performance of the PPO. CoMS (w/o_homelss) is worse because some edge servers will drop some tasks appropriately due to overload. CoMS (w/o_fine) does not consider fine collaboration task allocation, which will lead to high utilization of some edge servers, thus increasing the edge server operational overhead.

*4) Performance Comparison Under Long-Term Continuous Constraints:* Figs. 7 and 8 show the performance comparison of multiple model selection schemes on the two datasets under long-term continuous constraints. It can be seen that the TOO of each method increases over time. The reason is that task requests are generated in real-time and processing each task incurs the corresponding edge server operational overhead and inference accuracy loss. The succRatio is only relevant to the strategy and hardly changes over time. In particular, our CoMS always achieves the best results regarding TOO and succRatio, and its numerical results show that CoMS improves TOO and succRatio by at least 3.5% and 3.3% on both datasets, respectively. This is because, 1) a collaborative approach guarantees that arriving task requests will not be dropped due to overload of the DNN instance, thereby maintaining a high succRatio, 2) the fine-grained collaboration strategy guided by the utilization ratio enables CoMS to reduce the edge server operational overhead while maintaining high accuracy, thereby obtaining a lower average trade-off overhead.

*5) Utilization Comparison Under Different Algorithms:* To explore the reasons for the performance differences between different schemes, we count the utilization of DNN instances in each edge server at a certain time, as shown in Figs. 9 and 10.


(a)

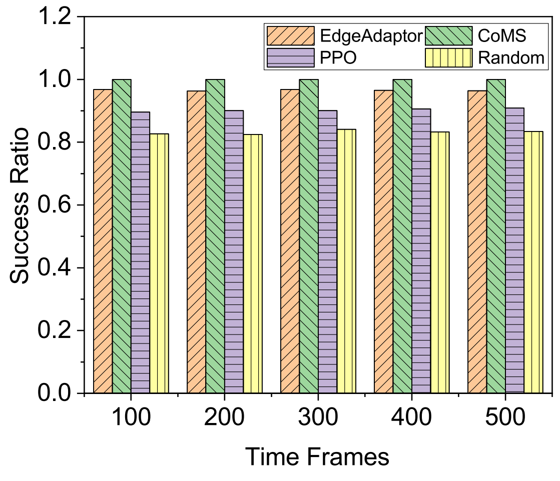
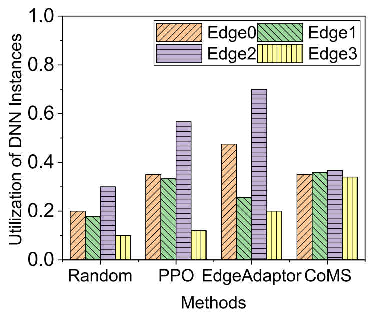
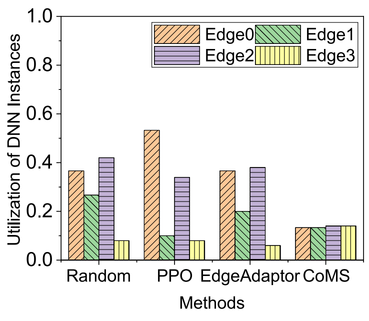

(b)

Fig. 8. Success ratio comparison of different algorithms under Long-Term continuous constraints. (a) Performance comparison on PeMSD8. (b) Performance comparison on PeMSD4.


(a)


(b)

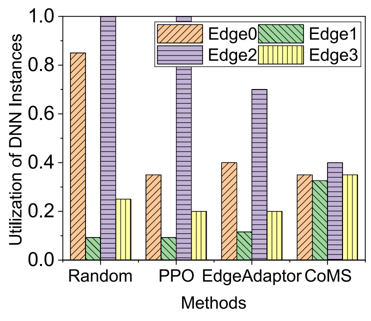
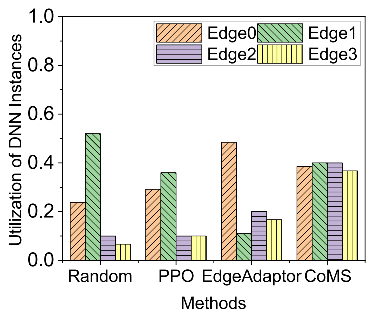
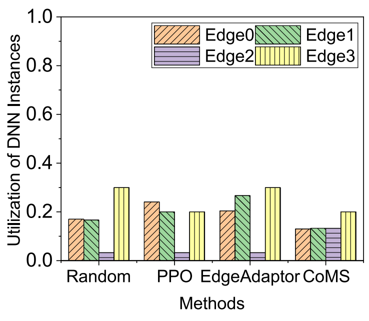

(c)

Fig. 9. Utilization comparison of different algorithms under different DNN instances on PeMSD8. (a) Utilization comparison on YOLOv2 instance. (b) Utilization comparison on SSD instance. (c) Utilization comparison on R-FCN instance.


(a)


(b)

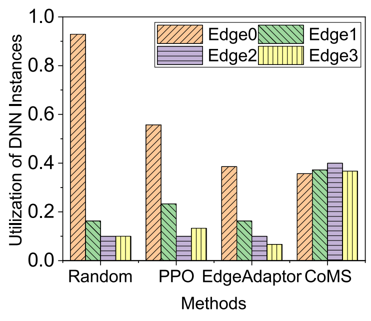

(c)

Fig. 10. Utilization comparison of different algorithms under different DNN instances on PeMSD4. (a) Utilization comparison on YOLOv2 instance. (b) Utilization comparison on SSD instance. (c) Utilization comparison on R-FCN instance.

Firstly, we find that the utilization of different DNN instances is different. This is because using different DNN instances to process the same task usually results in different edge server operational overhead and inference accuracy loss. To ensure better performance of the system, it is necessary to match each task with an appropriate DNN instance, resulting in differences in the utilization of different DNN instances. Meanwhile, it can be seen that the utilization of DNN has reached 100% in some methods, which shows that there is a high probability of current packet loss, making the success ratio less than 100%. Last but not least, thanks to the collaborative strategy, our CoMS can guarantee similar utilization of DNN instances of the same type in different edge servers, so it can better reduce overall overhead. In general, a large difference in utilization rate can harm the performance of DNN instances, *e.g.*, increasing overhead.

## VI. Conclusion

In this paper, we propose a fine-grained edge Collaborative Model Selection scheme (CoMS) for heterogeneous task requests, which aims to facilitate collaboration between edge servers and achieve more effective model selection, thereby improving the overall revenue of the system. Specifically, we first present a PPO reinforcement learning scheme based on real-time dynamic normalization strategy, which aims to alleviate the training instability problem caused by sudden increases in DLTs and accelerate the model convergence, thereby enhancing the efficiency of model selection. Then, we design a model selection strategy based on greedy algorithm that aims to prevent discarding some DLTs due to edge server overload. Finally, an efficient fine-grained collaborative model selection strategy is introduced to promote cooperation between edge servers, aiming at achieving a balance between the inference accuracy loss and edge server operational overhead. We conduct extensive experiments on two real-world datasets. The results show that the proposed scheme CoMS reduces the average trade-off overhead by 4.2% to 12.8% and improves the success ratio by 3% to 16% compared with three baselines.

However, data drift will occur in the long-term operation of real scenarios (*i.e.*, the distribution between the data for training the model and the data generated later is too different), which makes the initially deployed inference model no longer applicable. Therefore, continuous training of deployed inference models will be a hot topic in future work. In addition, the data for training the model comes from various edge servers. To avoid privacy leaks, the algorithm designed considering this security factor is more compatible with actual scenarios.

## References

[1] L. Wang et al., "Gecko: Resource-efficient and accurate queries in real-time video streams at the edge," in *Proc. IEEE INFOCOM 2024-IEEE Conf. Comput. Commun.*, 2024, pp. 257–266.

[2] R. Greer, A. Gopalkrishnan, N. Deo, A. Rangesh, and M. Trivedi, "Salient sign detection in safe autonomous driving: AI which reasons over full visual context," 2023, *arXiv:2301.05804*.

[3] L. Ren, Z. Jia, Y. Laili, and D. Huang, "Deep learning for time-series prediction in IIoT: Progress, challenges, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 11, 2023, doi: 10.1109/TNNLS.2023.3291371.

[4] K. Zhao et al., "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge DNN inference serving at scale," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5870–5886, Oct. 2023.

[5] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tut.*, vol. 22, no. 2, pp. 869–904, Secondquarter 2020.

[6] H. Zhou, T. Wu, X. Chen, S. He, D. Guo, and J. Wu, "Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6144–6159, Oct. 2023.

[7] M. Jafri, S. Srivastava, N. K. Venkategowda, A. K. Jagannatham, and L. Hanzo, "Cooperative hybrid transmit beamforming in cell-free mmWave MIMO networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6023–6038, May 2023.

[8] Z. Zhou, M. Shojafar, J. Abawajy, H. Yin, and H. Lu, "ECMS: An edge intelligent energy efficient model in mobile edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 238–247, Mar. 2022.

[9] Y. Zhang et al., "Cooperative edge caching: A multi-agent deep learning based approach," *IEEE Access*, vol. 8, pp. 133212–133224, 2020.

[10] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 447–457, Jan. 2020.

[11] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023.

[12] W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu, and L. Zhang, "Image-adaptive YOLO for object detection in adverse weather conditions," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 2, pp. 1792–1800.

[13] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.

[14] B. Singh, H. Li, A. Sharma, and L. S. Davis, "R-FCN-3000 at 30fps: Decoupling detection and classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1081–1090.

[15] B. Chen, A. Bakhshi, G. Batista, B. Ng, and T.-J. Chin, "Update compression for deep neural networks on the edge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3076–3086.

[16] Z. Li et al., "Curriculum temperature for knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 2, pp. 1504–1512.

[17] Q. Zhang et al., "Platon: Pruning large transformer models with upper confidence bound of weight importance," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 26809–26823.

[18] Y. Shang, Z. Yuan, B. Xie, B. Wu, and Y. Yan, "Post-training quantization on diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 1972–1981.

[19] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, and H. Kwon, "CLIO: Enabling automatic compilation of deep learning pipelines across IoT and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–12.

[20] J. Xie, Z. Zhou, T. Ouyang, X. Zhang, and X. Chen, "Fair DNN model selection in edge AI via a cooperative game approach," in *Proc. 2023 IEEE 43rd Int. Conf. Distrib. Comput. Syst.*, 2023, pp. 383–394.

[21] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "{INFaaS}: Automated model-less inference serving," in *Proc. 2021 USENIX Annu. Tech. Conf.*, 2021, pp. 397–411.

[22] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE INFOCOM 2020-IEEE Conf. Comput. Commun.*, 2020, pp. 257–266.

[23] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 3836–3851, Nov. 2022.

[24] I. T. Christou, N. Kefalakis, J. K. Soldatos, and A.-M. Despotopoulou, "End-to-end industrial IoT platform for quality 4.0 applications," *Comput. Ind.*, vol. 137, 2022, Art. no. 103591.

[25] M. Al-Amin et al., "Fusing and refining convolutional neural network models for assembly action recognition in smart manufacturing," in *Proc. Inst. Mech. Engineers, Part C, J. Mech. Eng. Sci.*, vol. 236, no. 4, pp. 2046–2059, 2022.

[26] C. Chen, C. Wang, B. Liu, C. He, L. Cong, and S. Wan, "Edge intelligence empowered vehicle detection and image segmentation for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 13023–13034, Nov. 2023.

[27] T. Shen et al., "{SOTER}: Guarding black-box inference for general neural networks at the edge," in *Proc. 2022 USENIX Annu. Tech. Conf.*, 2022, pp. 723–738.

[28] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, 2021.

[29] F. He, T. Liu, and D. Tao, "Why ResNet works? residuals generalize," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5349–5362, Dec. 2020.

[30] S. Chavhan et al., "Edge computing AI-IoT integrated energy-efficient intelligent transportation system for smart cities," *ACM Trans. Internet Technol.*, vol. 22, no. 4, pp. 1–18, 2022.

[31] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," in *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[32] U. Ojha, Y. Li, A. Sundara Rajan, Y. Liang, and Y. J. Lee, "What knowledge gets distilled in knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023,vol. 36, pp. 11037–11048.
[33] Y. Tian, H. Chen, T. Guo, C. Xu, and Y. Wang, "Towards higher ranks via adversarial weight pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 1189–1207.
[34] E. Nehme, O. Yair, and T. Michaeli, "Uncertainty quantification via neural posterior principal components," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 37128–37141.
[35] O. Zohar, S.-C. Huang, K.-C. Wang, and S. Yeung, "Lovm: Language-only vision model selection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 33120–33132.
[36] B. Lu, J. Yang, L. Y. Chen, and S. Ren, "Automating deep neural network model selection for edge inference," in *Proc. 2019 IEEE 1st Int. Conf. Cogn. Mach. Intell.*, 2019, pp. 184–193.
[37] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. S. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.
[38] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A {Low-Latency} online prediction serving system," in *Proc. 14th USENIX Symp. Networked Syst. Des. Implementation*, 2017, pp. 613–627.
[39] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021.
[40] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2020.
[41] X. Huang and S. Zhou, "Latency guaranteed edge inference via dynamic compression ratio selection," in *Proc. 2020 IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–6.
[42] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.
[43] C.-Y. Hsieh, Y. Ren, and J.-C. Chen, "Edge-cloud offloading: Knapsack potential game in 5G multi-access edge computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 11, pp. 7158–7171, Nov. 2023.
[44] M. Kim et al., "Learning collaborative policies to solve np-hard routing problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 10418–10430.
[45] D. S. Gadiraju, V. Lalitha, and V. Aggarwal, "An optimization framework based on deep reinforcement learning approaches for prism blockchain," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2451–2461, Jul./Aug. 2023.
[46] CoMS, 2024. [Online]. Available: https://github.com/xuaikun/CoMS.git
[47] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[48] J. Queeney, Y. Paschalidis, and C. G. Cassandras, "Generalized proximal policy optimization with sample reuse," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 11909–11919.
[49] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 01, pp. 922–929.