DNN Inference Acceleration and Reliable Task Offloading in Mobile-Edge Computing

Changke Wang, Xiaowen Huang, Wenqian Zhang*, Guanglin Zhang*
College of Information Science and Technology, Donghua University, Shanghai, China
{2222100, huangxiaowen}@mail.dhu.edu.cn, {wqzhang, glzhang}@dhu.edu.cn

***Abstract*—Most computation-intensive tasks are offloaded to high-performance edge servers, making the acceleration of deep neural networks (DNNs) a prominent research topic in mobile-edge computing networks. Deploying computationally demanding deep learning networks on resource-constrained mobile devices poses significant challenges. We first propose the overlapped partitioning DNNs, along with mobility-aware task offloading, focusing on analyzing the trade-offs between DNN inference acceleration and offloading reliability design insights. We then establish a collaborative DNN partitioning and offloading scenario, derive an acceleration and reliability model for DNN inference, and define the collaborative partitioning and offloading problem. Finally, we propose two algorithms: self-attention deterministic Policy Gradient (SAD) and optimal allocation algorithm (OAA). SAD globally optimizes task offloading strategies for mobile-edge DNNs, while OAA fine-tunes the strategy, optimizing inference acceleration and reliability between neighboring communication nodes. The simulation evaluations demonstrate the proposed scheme's superiority.**

***Index Terms*—Mobile-edge computing, Overlapped partitioning, DNN acceleration, Task Offloading.**

I. Introduction

Mobile-edge computing (MEC) is proving to be a valuable solution, enabling resource-constrained devices such as smartphones and smart wearables to handle resource-intensive applications. Many AR, VR, and AI-based applications, including virtual worlds and digital twins, involve large-scale and latency-sensitive Deep Neural Network (DNN) computations. These tasks are challenging for user devices to handle independently. To achieve large-scale intelligent computing at the mobile-edge, support for computationally intensive and latency-sensitive DNN inference tasks is critical.

With the continuous development of mobile computing chips and the enhanced computational power of mobile devices, it remains difficult to meet the demands of large-scale data processing and computational power requirements. For instance, in image classification using VGG16 with an input image of 224×224 pixels from the ImageNet dataset, 569M FLOPs are required for inference. Approximately 3.2 seconds and 0.18 seconds are needed when using a Raspberry Pi 5 and Jetson Nano, respectively. Vehicles, mobile phones, mobile robots, and VR all demand significant computational power

This work is supported in part by the National Natural Science Foundation of China under Grant 62301307, in part sponsored by Shanghai Pujiang Programme, and in part by the Chenguang Program of Shanghai Education Development Foundation and Shanghai Municipal Education Commission.

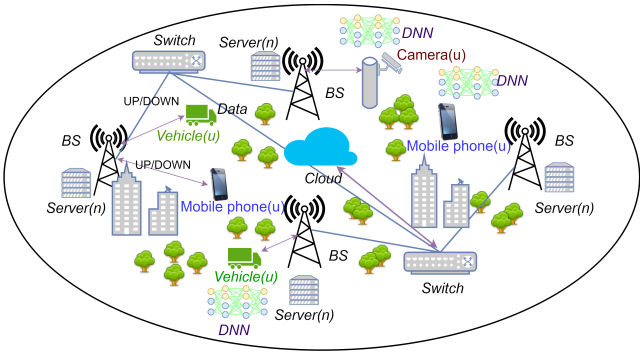


Fig. 1. Task offloading scenario in MEC.

[1]. However, the computational power of edge devices remains inadequate to satisfy real-time computing tasks. Therefore, accelerating DNN inference and employing effective task offloading strategies are necessary in mobile-edge computing.

Many studies on task offloading in mobile edge computing networks focus on resource allocation, edge device collaboration, and coordination between edge and cloud servers, but they often overlook fine-grained task offloading and the collaboration between edge and mobile devices. Most high-performance task offloading research emphasizes partitioning DNNs to accelerate computation [2], [3]. Some approaches employ horizontal partitioning of DNN layers, offloading parts of the data to different edge devices. However, this results in frequent communication between nodes, leading to resource wastage and increased delays. Alternatively, vertical partitioning of DNN layers involves computing the first few layers on the mobile device, with the remaining layers offloaded to edge devices. This still demands significant resources on the mobile device or edge device, and does not fully leverage edge collaboration. Furthermore, these studies often neglect the reliability of DNN task offloading, particularly considering the instability of wireless connections in mobile environments. In autonomous driving, stable task offloading and accelerated DNN inference are critical for real-time image recognition by vehicle sensors. Efficient offloading ensures rapid decision-making, while task acceleration reduces delays, improving safety and driving performance. Therefore, DNN inference acceleration and reliable offloading play a vital role in autonomous driving systems.

In this paper, we propose a resource management scheme

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

for task offloading that leverages edge collaboration. As illustrated in Fig. 1, the diagram shows DNN task offloading in a small area, where tasks from mobile phones, cameras, and vehicles are offloaded to a base station (BS). The edge server processes these tasks and sends the results back. When the resources of the edge server are insufficient, tasks are offloaded to other edge nodes. Therefore, our scheme targets scenarios involving multiple devices serviced by multiple BSs, enabling the collaborative operation of several edge nodes. A time-slot system model is proposed, which comprehensively considers task scheduling on BSs, allocation of computational resources at the BSs, and the transmission rate allocation across both wireless and wired links.

The objective of the optimization problem is to minimize the total task processing delay while maintaining the long-term stability of task queues across all edge servers. Consequently, overlapping partitions are employed to offload tasks to disparate edge devices, thereby achieving a balance between accelerating DNN inference and ensuring reliability, resulting in a collaborative partitioning and offloading (CPO) problem. To address this issue, we propose the self-attention deterministic policy gradient (SAD), which is constructed upon the deep deterministic policy gradient (DDPG) framework [4] and incorporates a self-attention mechanism. The self-attention mechanism facilitates the extraction of additional features from the current task's state, thereby enabling the generation of a global offloading strategy. Subsequently, the optimal allocation algorithm (OAA) employs a Markov decision process (MDP) to predict the trajectory of mobile devices, calculate the communication rate between devices and nodes, and fine-tune the global offloading strategy, ultimately producing the optimal allocation strategy. Extensive simulations are conducted, varying key parameters such as task offloading rate, the number of mobile devices, and link transmission rates. The results validate the trade-offs between task processing delay and task offloading rate, as well as the effectiveness of the algorithm. The main contributions of this paper are as follows:

- We propose an overlapped DNN partitioning that addresses the trade-off between inference acceleration and offloading reliability in edge computing scenarios.
- To address this challenge, we design the SAD-OAA task offloading algorithm. The SAD generates a globally optimized offloading strategy based on the current environment, while the OAA fine-tunes the communication nodes within the strategy to tackle the NP-hard problem.
- Extensive simulation experiments, supported by performance analysis and comparisons with other algorithms, demonstrate the superiority of the proposed approach in both inference acceleration and offloading reliability.

II. RELATED WORK

In terms of model accuracy, Xie et al. [5] employed the Nash Bargaining Game (NBG) to select DNN models, aiming to maximize the inference accuracy of the entire system. In terms of DNN inference partitioning, [2] used a policy gradient algorithm to dynamically determine the partitioning of DNNs and applied a heuristic resource allocation algorithm (HCRA) to minimize inference delay under energy constraints. A dynamic programming-based strategy for both intra-layer and inter-layer model partitioning is proposed in [6], which adapts to the constraints of edge resources. Zeng [3] developed CoEdge, which dynamically allocates workloads based on computational capacity and network conditions, enabling heterogeneous edge devices to collaborate for DNN inference.

Zhang [7] introduces Effect-DNN, which formulates the optimization problem as a dynamic MINLP and uses convex optimization and heuristic algorithms to balance latency and energy consumption through collaborative partitioning and task offloading. In [8], a collaborative inference system supports custom fine-grained scheduling and adapts to various CNN architectures. A proportional synchronization scheduler effectively balances computation and synchronization, enhancing inference efficiency on heterogeneous edge devices.

III. PRELIMINARY

A. Overlapping partitions

Overlapping partitioning divides the input data into overlapped segments, which helps maintain data continuity while reducing the overall input data volume. As shown in Fig. 2, the mobile device (u) is connected to two edge nodes (n_1, n_2). It is assumed that the two edge nodes have the same computational capacity, communication success rate, and communication bandwidth.

To demonstrate the effectiveness of overlapped partitioning, we compare the following three solutions: Solution I, full offloading: The mobile device offloads the entire DNN model. Solution II, non-overlapped partition offloading: The input data is divided into two non-overlapped parts. Solution III, overlapped partition offloading: the input data is divided into two overlapped partitions, each covering two-thirds of the original data. For performance evaluation, three metrics are defined: acceleration ratio, communication success rate, and recognition success rate. The acceleration ratio is the ratio of the DNN inference time, including inference time and the offloading and transmission time to the BS, compared to Solution I. The communication success rate refers to the success rate of offloading from user devices to the BS. The communication success probabilities for Solutions I, II, and III are configured as 80%, 90%, and 85%, respectively, while the classification recognition rates are 93.9%, 73.35%, and 86.32%. The communication success rate and recognition accuracy for Solutions I and II are calculated for individual partitions. The classification accuracy is obtained by training the ResNet50 model on the ImageNet100 dataset. Partitioned models are trained on images segmented from the training set, with separate models for the upper and lower segments. The recognition accuracy for full offloading is derived by summing the weights of the DNN inference results from both partitions and selecting the classification with the highest value. In Fig. 3, the communication success rates of Solutions I, II, and III represent the probability that at least one partition's transmission is completed. The recognition success rate is the product of the

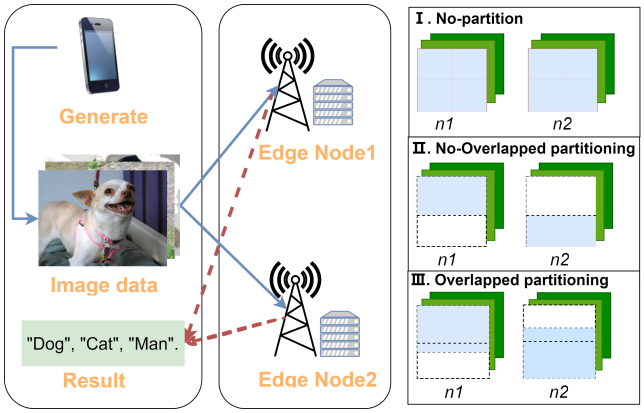
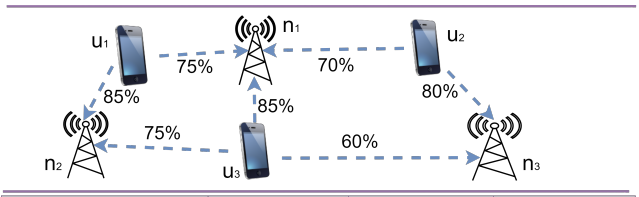


Fig. 2. DNN offloading model in MEC.



	Ⅰ	Ⅱ	Ⅲ
Acceleration Ration	1x	2x	1.5x
Trans Success Ration	96.00%	99.00%	97.75%
Offloading Recognition Success Ration	90.14%	78.28%	86.53%

Fig. 3. Connection Probabilities and Offloading Success Rates for Mobile Devices to Edge Nodes.

communication success rate and classification accuracy. For Solutions II and III, it requires the cumulative sum of single-partition offloading and full partition offloading. The resulting values are 90.14%, 78.28%, and 86.53%, with corresponding acceleration factors of 1x, 2x, and 1.5x, respectively. This indicates that Solution III achieves a reliable balance between acceleration and offloading recognition success rates.

B. Optimized Allocation Offloading

Device mobility can affect the communication rate, leading to variations in communication success rate. As shown in Fig. 3, the connection probabilities between multiple mobile devices and several edge nodes range from 60% to 85%. For the mobile device u_3 in Fig. 3, the connection probabilities for offloading to nodes n_2 and n_3 are 75% and 60%, respectively, with an offloading success rate is 90%. If the offloading is redirected to nodes n_1 and n_2 with connection probabilities of 80% and 75%, the total offloading probability increases to 96.25%. We take into account the different connection probabilities of each node, offloading tasks to more reliable nodes to maximize the offloading success rate.

IV. CPO Problem

A. System Overview

First, we designate the mobile devices set as $\mathcal{U} = \{1, 2, \ldots, u, \ldots, U\}$. Roadside BSs are defined as edge nodes, denoted as $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$. The system operates in discrete time, defined as $\mathcal{T} = \{1, 2, \ldots, t, \ldots, T\}$, the length of each time slot t is τ. The entire system process involves four steps. Step 1: mobile devices and roadside BSs upload relevant information, including location, computing capabilities, and tasks, to the cloud server. The cloud server runs an algorithm to compute the offloading strategy and sends instructions to all mobile devices and edge nodes. Then, mobile devices and edge nodes perform task allocation in each time slot. Step 2: the mobile devices perform overlapped partitioning of the input data, offloading the task to the designated edge node via U2I communication [9]. Step 3: edge nodes receive the tasks and process them in the order they are received, and tasks that exceed the time limit are canceled. Step 4: after completing the task processing, the edge nodes send the results back to the mobile devices. In this work, we assume that the tasks of each mobile device are generated independently and identically distributed, and the task types are the same.

B. Relative Distance State Analysis:

This section models the relative distance between the mobile device and the edge node. A discrete MDP is used to predict the next position of the mobile device, allowing the calculation of the relative distance between the user's mobile device and the edge node. Time s is defined differently from t, s represents discrete intervals greater than t. The state of the mobile device u at s is defined as $MS_s^u = \{p_{s,x}^u, p_{s,y}^u, \theta_s^u\}^T$, position $p_s^u = \{p_{s,x}^u, p_{s,y}^u\}^T$, $p_s^u \in \mathbb{R}$ and angle $\theta_s^u \in [-\pi, \pi]$ at each s, where $\mathbb{R}$ denotes the real-valued size of the test area. The device's position is situated within the entire mobile area. The input of the current state transitioning to the next state is represented as $z_s^u = (v_s^u, \omega_s^u)$, where v_s^u denotes linear velocity and ω_s^u represents the angular yaw rate at a given time. The continuous-time differential-driven kinematics are discretized using a time step of α, resulting in a discrete-time kinematics formula for differential mobile devices.

$$MS_{s+1}^u = MS_s^u + \begin{pmatrix} \alpha\cos\theta_s^u & 0 \\ \alpha\sin\theta_s^u & 0 \\ 0 & \alpha \end{pmatrix} \begin{pmatrix} v_s^u \\ \omega_s^u \end{pmatrix} + \begin{pmatrix} w_x^u \\ w_y^u \\ w_\theta^u \end{pmatrix}. \quad (1)$$

For Eq. 1, MS_{s+1}^u denotes the positional state at the next time step. Control inputs are defined as $v_s^u \in [0, v_{\max}^u]$, with $w_x^u, w_x^u \in N_g(0, 0.04^2)$ and $w_\theta^u \in N_g(0, 0.004^2)$, Where N_g represents a noise probability function modeled by a Gaussian distribution. In the context of MDP trajectory prediction, the state transitioning cost is defined as follows:

$$\begin{aligned} l(MS_s^u, z_s^u) = {} & \tilde{p}_s^{uT} Q \tilde{p}_s^u + q(1 - \cos\tilde{\theta})^2 \\ & + z_s^{uT} R z_s^u + \text{penalty}. \end{aligned} \quad (2)$$

Based on the MDP trajectory prediction outlined in reference [10], the control action sequence z_s^u is derived using a MDP. The objective of calculating z_s^u is to minimize the state transitioning cost, as indicated in Eq. 2 from [10]. Here, $Q \in \mathbb{R}^{2\times 2}$ is a positive definite matrix, represents the penalty associated with position tracking error, $q > 0$ denotes the error in the penalty direction, and penalty indicates the adjusted penalty error. To minimize $l(MS_s^u, z_s^u)$, z_s^u is derived and substituted into Eq. 1 to calculate MS_{s+1}^u, representing the device's position at the next time step.

C. DNN Inference Cost Calculation Model

For the DNN models MobileNet, ResNet, and GoogleNet are respectively represented as loop structure, chain structure, and directed acyclic graph (DAG) structure. The loop structure facilitates inter-layer communication within and across multiple layers. The chain structure enables the sequential connection of layers. In contrast, the DAG structure has no loops but consists of multiple paths from one layer to another. For k types of DNN models, computation for these k types of models is categorized into convolutional layers, fully connected layers, and pooling layers, denoted by C_k, F_k, and P_k, respectively. The IDs for the three types of layers are denoted as i $(1 \le i \le C_k)$, j $(1 \le j \le F_k)$, and m $(1 \le m \le P_k)$. The computational workloads for C_k, F_k, and P_k, denoted as s_i, s_j, and s_m, are thus calculated using the following formulas:

$$s_i = H_i^{in} \cdot W_i^{in} \cdot (c_i^{in} \cdot ker_i^2 + 1) \cdot c_i^{out}. \tag{3}$$

$$s_j = (2 \cdot I_j - 1) \cdot O_j. \tag{4}$$

$$s_m = H_m^{in} \cdot W_m^{in} \cdot c_m^{in} \cdot ker_m^2. \tag{5}$$

The input and output of the convolutional layer are represented as $\langle H_i^{in}, W_i^{in}, c_i^{in}\rangle$, and $\langle H_i^{out}, W_i^{out}, c_i^{out}\rangle$, corresponding to the height, width, and number of channels of the layer's data. Similarly, for the pooling/activation layer m $(1 \le m \le P_k)$, the input and output are denotes as $\langle H_m^{in}, W_m^{in}, c_m^{in}\rangle$ and $\langle H_m^{out}, W_m^{out}, c_m^{out}\rangle$. ker_i^2 represents the kernel size of the convolutional layer. For the fully connected layer, I_j and O_j represent the length of the one-dimensional input and output arrays, respectively.

According to Eq. 3, Eq. 4, and Eq. 5, the computational cost of a DNN model inference task can be approximately expressed as:

$$CO_{un} = \sum_{i\in\{1,\ldots,C_k\}} s_i + \sum_{j\in\{1,\ldots,F_k\}} s_j + \sum_{m\in\{1,\ldots,P_k\}} s_m. \tag{6}$$

We partition and offload the data from u to n. Specifically, up_{un} represents the input matrix of output data, partitioned from u and offloaded to n. The uploaded data is denoted as $up_{un} = \langle H^{out}, W^{out}, C^{out}\rangle$. Each data point consists of 8 bits, and the transmission overhead between u and n, denoted as TO_{un}, can be computed as follows:

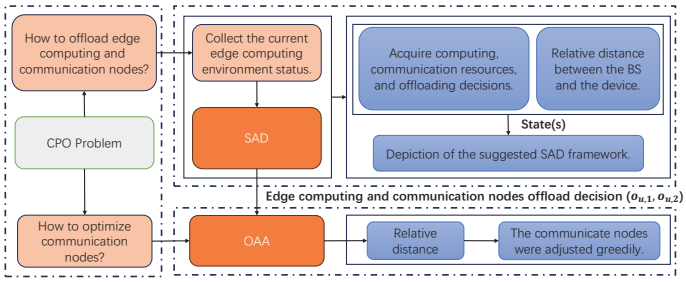


Fig. 4. Overview of CPO Problem Solutions.

$$TO_{un} = H^{out} \cdot W^{out} \cdot C^{out} \cdot 8 \text{ bit}. \tag{7}$$

D. Trans Delay and Inference Delay

First, we establish the channel fading model in mobile communication, and to calculate the communication rate, we employ the Rayleigh fading model. Let the distance between u and n be d_{un}. Therefore, the signal-to-noise ratio $\gamma(d_{un})$ is given by:

$$\gamma(d_{un}) = \frac{P_t}{N_0 \cdot 10^{\left(\eta \log_{10}\left(\frac{d_{un}}{10}\right)\right)}}. \tag{8}$$

Where P_t is the transmission power, N_0 is the noise power and η is the path loss exponent. Based on the formulation presented Eq. 8, let the bandwidth of edge node n be B, and the communication coverage range be β_n. Thus, the transmission rate TR_{un} is expressed as:

$$TR_{un} = \begin{cases} B\log_2(1+\gamma(d_{un})) & \text{if } d_{un} \le \beta_n \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

When DNN inference is performed on the edge nodes, the computational capacity of the edge server is represented by χ_n. The offloading delay (OD_{un}) is calculated by the following equation:

$$OD_{\text{un}} = \frac{TO_{\text{un}}}{TR_{\text{un}}} + \frac{CO_{\text{un}}}{\chi_n}. \tag{10}$$

Let ψ_u denote the computational capacity of the mobile device, while LD_{u} represents the local inference delay on the mobile device. The delay LD_{u} can be expressed as:

$$LD_{\text{u}} = \frac{CO_{\text{un}}}{\psi_u}. \tag{11}$$

V. PROPOSED SOLUTION

In the DNN task offloading environment, an overview of the CPO solution is presented in Fig. 4. We divide the CPO problem into two subproblems: edge computing and communication node allocation, and communication node fine-tuning. First, we obtain the current computational resources, communication resources, and the relative distances between the BS and devices in the DNN edge computing environment. Node allocation is performed using SAD-DDPG for output

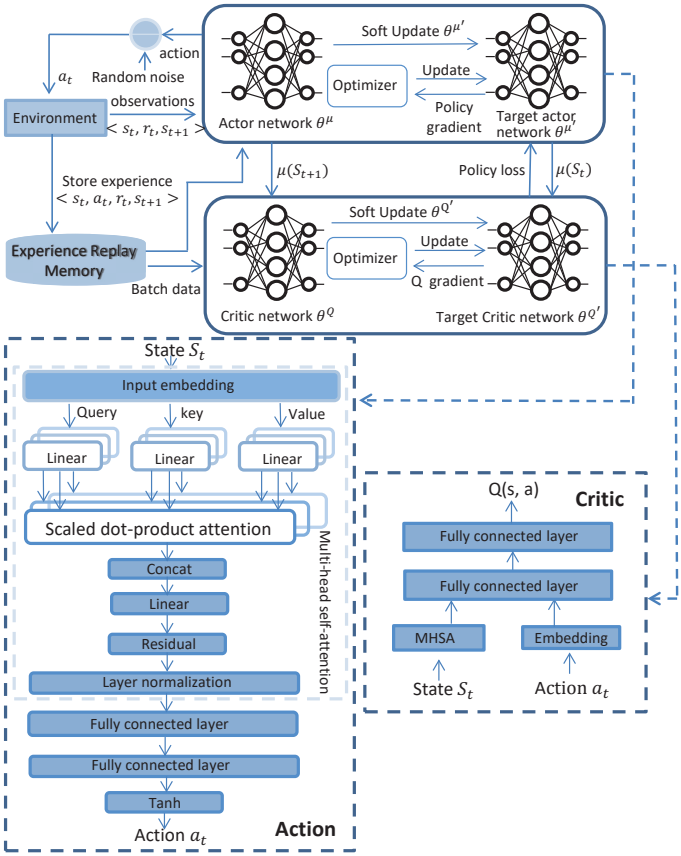


Fig. 5. Depiction of the suggested SAD framework.

node decision-making, followed by OAA for fine-tuning the communication nodes.

A. Reinforcement Learning Algorithm Architecture

The overall structure of SAD is presented as shown in Fig. 5. DDPG is designed specifically for continuous action spaces and consists of an actor network and a target network. The upper part of Fig. 5 shows the DDPG architecture. Our SAD algorithm improves DDPG, and incorporate multi-head self-attention (MUSA) into both the actor and critic networks. The actor network employs a self-attention structure to extract global information from the input data and identify implicit relationships between state elements. Subsequently, weighted attention is assigned to predict the policy distribution. Similarly, the critic network employs MUSA for state encoding through down-sampling. The action variable, denoted by a_t, is processed through an embedding layer, which establishes connections linking the encoded state and action.

During training, the agent receives an initial state s_t from the environment and generates an action a_t through the actor network, with random noise added for exploration. After executing the action, the next state s_{t+1} and reward r_t are obtained, and the sample $\langle s_t, a_t, r_t, s_{t+1} \rangle$ is stored in the experience buffer for updates. Here, a_t is defined as $\{(o_{1,1}, o_{1,2}), \ldots, (o_{u,1}, o_{u,2}), \ldots, (o_{U,1}, o_{U,2})\}$, which is $\mathcal{N}$ offloaded to n_1 and n_2. s_t is defined as $\{a_t, OC_n, OT_n, x_s^u\}$. Where OC_n and OT_n represent the remaining computation capacity and communication capacity at the current time t, s represents time points that are close to but less than t. The reward r_t is defined as $r_t = \sum_u \frac{1}{OD_{\text{un}}}$. The critic network minimizes loss by computing the Q-value for the current state and action, while the actor network updates the policy through policy gradients. Finally, target network parameters are synchronized to ensure consistency. The actor network is composed of two sub-layers: Multi-head self-attention (MHSA) layer and two fully connected layers (FCNs). In the first sub-layer, the input state is encoded, projecting the input sequence into query, key, and value vectors. Attention weights are computed, followed by residual connections and layer normalization. The output of the MHSA layer, denoted as MHSAOutput and is expressed in the form LayerNorm$(x+\text{MHSA}(x))$. The second sub-layer uses the Tanh function to map the FCN output to a range of -1 to 1, producing the final task offloading decision.

B. Optimization Allocation Algorithm

The overall design concept of OAA is summarized as follows. UP is defined as the communication upload node and CN as the calculation node. The actor output of the SAD network, denoted as $a_{u,n} = [UP, CN]$, and the MDP prediction in Section III use to obtain the next state of mobile device. The set of relative distances between u and u is $D_{u,n} = \{d_{1,1}, d_{1,2}, ..., \ d_{u,n}, \ ..., d_{U,N}\}$. According to Eq. 9, $P_{u,n} = \{p_{1,1}, p_{1,2}, ..., \ p_{u,n}, \ ..., p_{U,N}\}$ is calculated. Here, $p_{u,n}$ represents the communication rate between u and n, which is positive for communication nodes and 0 for non-communication nodes. The UP nodes are adjusted using a greedy approach, whereby optimal communication nodes are selected based on $P_{u,n}$ with the objective of improving the DNN offloading rate and reducing inference time. This approach is primarily aimed at enhancing the offloading reliability of DNN inference.

VI. PERFORMANCE EVALUATION

In this section, we first evaluate the accuracy of overlapped partitioning, showing minimal impact on accuracy. It then assesses the error rate of MDP trajectory prediction. Finally, simulations in various environments are conducted to evaluate the performance of SAD-OAA, focusing on the speedup ratio and offloading ratio.

A. Recognition Rate of DNN Overlapping Partitioning

ResNet network models are used as references for classification accuracy. After partitioning the image data with a 2/3 overlap, the classification accuracy of the DNN model decreases. The testing results for ResNet networks on the ImageNet100 dataset after training under overlapped partitioning are shown in Fig. 6, where the horizontal axis represents the index values of the testing dataset images, with each index corresponding to 5 images. The accuracy of the ResNet model for overlapped partition offloading is 3.8% lower compared to non-overlapped partitioning. This slight decrease in accuracy has negligible a impact on recognition performance, indicating that overlapped partitioning is a viable approach.

Fig. 6. The accuracy of ResNet no-overlapped partitioning and with 2/3 overlapped partitioning.

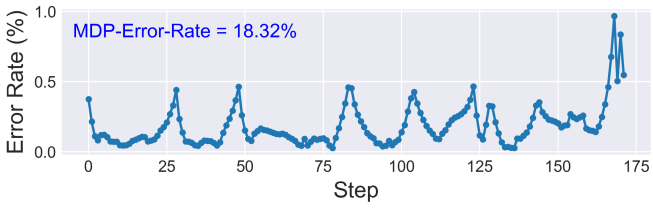
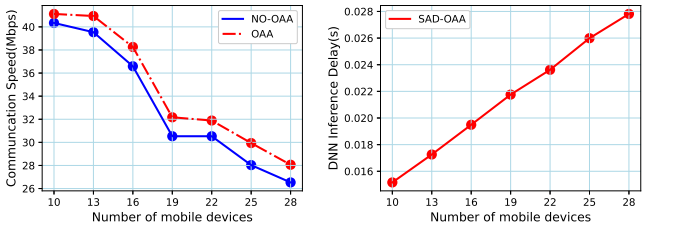


Fig. 7. Probability distribution of the relative distance error predicted by the MDP.

TABLE I. SIMULATION PARAMETERS

Parameter	Value
computation capacity of edge nodes	$[4, 5]$ GHz
wireless bandwidth of U2I	15 MHz
communication coverage range β_n of U2I	$[0, 800m]$
transmission power P_t of U2I	24 dBm
path loss exponent η	3
noise power N_0	-70 dBm
mobile device max speed $v^u_{\max}$	50 m/s
mobile device movement range $\mathbb{R}$	$[0, 3km] \times [0, 3km]$
task completion time limit	0.1 seconds



(a) Throughput from the device to the BS. (b) DNN inference delay from edge offloading.

Fig. 8. Communication throughput and latency of DNN inference offloading to edge.

B. Experiments setup

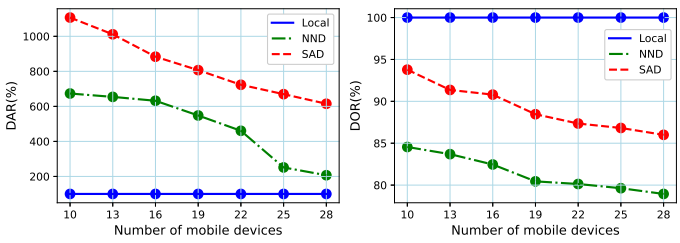
In our experiments, the simulation is conducted on a computer equipped with an Intel 12th-generation processor, 16GB of RAM, and an RTX 4090 graphics card. We first use the commonly employed ResNet network for image classification, which serves as our DNN task request. Overlapped partitioning is applied to the input images, dividing the image data into two-thirds. A custom mobile device trajectory map estimates the position every $\alpha = 0.2$ seconds, providing the next location and calculating the theoretical communication speed. Tasks generated by the mobile devices are random, with a new task being created every $\tau = 0.05$ seconds. The communication success rate for a single partition varies with distance and is defined as $\left(\frac{800-d_{un}}{3000} + 70\%\right) \times 85\%$. The data size of each task is $TO_{\text{data}} = [150, 224, 3] \times 8$ bits. The experiment involves 10 edge nodes, with the number of mobile devices set to $[10, 13, 16, 19, 22, 25, 28]$. The remaining primary parameters are summarized in Table I.

In terms of task allocation, communication, and computing resources are allocated through a deep learning strategy to formulate offloading strategies for DNN tasks. Once the DNN computing task is uploaded to the BS, the appropriate edge node is selected for task offloading. The DNN task is then offloaded to the edge node, where the node's server performs the DNN computation. Computational resources are shared among edge nodes through an interaction channel, thus improving efficiency and resource utilization.

The performance of DNN inference is compared under the following scenarios: (i) All DNN inference tasks process locally; (ii) The DNN inference tasks are fully offloaded to different edge nodes using the strategy generated by nearest neighbor offloading (NND), which selects the nearest edge node; (iii) Overlapped partitioning is offloaded to different edge nodes, employing our SAD-based offloading strategy. In the case of overlapped partitioning, OAA incorporates to conduct an additional set of experiments; (iv) Our proposed SAD-OAA strategy is used for offloading DNN tasks. However, the current solutions do not sufficiently demonstrate the superiority of our SAD-OAA strategy in the overlapped partition offloading scenario. Therefore, additional comparisons with other offloading strategies are made, implementing the following two approaches. (v) Dynamic resource allocation (DRA) [11], where offloading is based on the communication and computational resources of edge nodes; (vi) The use of the DDPG deep reinforcement learning algorithm to optimize DNN task offloading decisions [12].

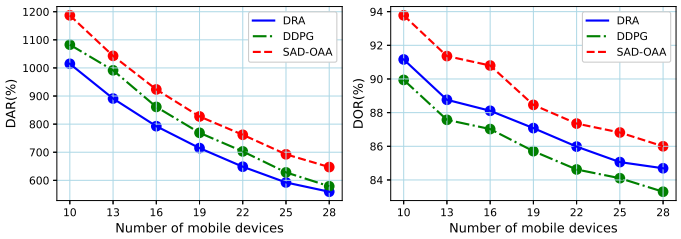
To evaluate performance, two key metrics are defined as follows:

- DNN Acceleration Ratio (DAR): It is the average ratio of LD_{u} in Eq. 11 to OD_{un} in Eq. 10.
- DNN Offloading Rate (DOR): It is the ratio of DNN inference tasks successfully completed by offloading to edge nodes.



(a) DNN Acceleration Ratio (b) DNN Offloading Rate

Fig. 9. Performance comparison under different numbers of UEs.



(a) DNN Acceleration Ratio (b) DNN Offloading Rate

Fig. 10. Performance comparison under numbers of UEs with different offloading strategies.

C. Experiments results and analysis

The predicted trajectories of 10 users are manually marked, and their paths are estimated. Based on Eq. 1, the next node's position is predicted. Fig. 7 shows the probability results of prediction errors for a mobile device tested on a custom-configured road. The horizontal axis represents the time steps to traverse the entire route, while the vertical axis represents the predicted probability values. The average prediction error between the predicted position and the actual position is 18.32%. The probability value is calculated as the error divided by the set threshold, with the maximum probability being 1. Fig. 8(a) shows the communication throughput between the device and the BS after applying the SAD strategy. The throughput decreases as the number of devices increases. When the number of devices reaches 16, a significant drop in throughput occurs because the BS is no longer idle. Fine-tuning the SAD output nodes using OAA improves the communication throughput.

The experiment is set up with 10 edge nodes, and the number of mobile devices is increased from 10. Fig. 9 presents the simulation results for schemes (i), (ii), and (iii) under the default settings. When the number of mobile devices is 10, a DOR of 89.12% and a DAR of 673.40% are achieved from schemes (ii) and (i), respectively, demonstrating the feasibility of offloading DNN tasks to edge nodes. Compared to scheme (ii), scheme (iii) exhibits an overall DAR improvement of 69.77% and a DOR increase of 9.60%, indicating that overlapped partitioning significantly enhanced DNN acceleration and offloading reliability. In scenarios using overlapped partitioning, under the DRA, DDPG, and SAD-OAA algorithms, Fig. 10 shows that scheme (iv) improves the DAR by 15.51% and 8.03% compared to schemes (vi) and (v), respectively, and the DOR by 3.12% and 2.11%, respectively. This further validates that SAD extracted more features under MUSA, contributing to significant global optimization, and with OAA fine-tuning the communication nodes, offloading reliability is further improved. Among them, Fig. 8(b) presents the average delay under DNN inference. When the number of devices reaches 28, the DNN inference delay can be kept within 0.03 seconds, which can meet the sensitivity requirements of DNN. Thus, as demonstrated by Figs. 9 and 10, the proposed SAD effectively allocates computation and communication nodes. Subsequently, fine-tuning the communication nodes through OAA enhances communication stability, leading to significant improvements in DNN inference acceleration and offloading stability.

VII. CONCLUSION

This paper proposes an optimization framework for DNN task offloading and inference on mobile-edge devices, addressing the partitioned offloading of DNN tasks, mobile position prediction, and communication resource optimization. A task offloading algorithm based on self-attention DDPG is designed and implemented, while efficient mobile position prediction and node selection are achieved through an MDP. OAA technology is introduced to resolve communication resource limitations, and the effectiveness of these methods is validated on a simulation platform. Simulation experiments demonstrate that overlapped partitioning and the SAD-OAA algorithm offer significant advantages in improving DNN acceleration and offloading reliability.

REFERENCES

[1] J. Mao, Z. Yang, W. Wen, C. Wu, L. Song, K. W. Nixon, X. Chen, H. Li, and Y. Chen, "Mednn: A distributed mobile system with enhanced partition and deployment for large-scale dnns," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 751–756.

[2] Y. Su, W. Fan, L. Gao, L. Qiao, Y. Liu, and F. Wu, "Joint DNN partition and resource allocation optimization for energy-constrained hierarchical edge-cloud systems," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3930–3944, 2023.

[3] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 595–608, 2021.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1509.02971.

[5] J. Xie, Z. Zhou, T. Ouyang, X. Zhang, and X. Chen, "Fair DNN model selection in edge AI via a cooperative game approach," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 383–394.

[6] P. Dai, B. Han, K. Li, X. Xu, H. Xing, and K. Liu, "Joint optimization of device placement and model partitioning for cooperative DNN inference in heterogeneous edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2024, doi: 10.1109/TMC.2024.3457793.

[7] X. Zhang, M. Mounesan, and S. Debroy, "Effect-DNN: Energy-efficient edge framework for real-time DNN inference," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 10–20.

[8] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "Deepslicing: Collaborative and adaptive CNN inference with low latency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2175–2187, 2021.

[9] C. Liu, L. Huang, and D. Zanliang, "A two-stage approach of joint route planning and resource allocation for multiple UAVs in unmanned logistics distribution," pp. 1–1, 01 2022.

[10] G. Sambaran and A. Nikolay, "Trajectory tracking with markov decision process," 2022, accessed: 2022-10-05. [Online]. Available: https://github.com/SambaranRepo/Trajectory-Tracking-with-Markov-Decision-Process.

[11] J. Xue and X. Guan, "Collaborative computation offloading and resource allocation based on dynamic pricing in mobile edge computing," *Computer Communications*, vol. 198, pp. 52–62, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422004352

[12] L. Chen, G. Gong, K. Jiang, H. Zhou, and R. Chen, "DDPG-based computation offloading and service caching in mobile edge computing," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2022, pp. 1–6.