Game-Based Adaptive FLOPs and Partition Point Decision Mechanism With Latency and Energy-Efficient Tradeoff for Edge Intelligence

Xin Niu, Yajing Huang, Zhiwei Wang, Chen Yu, *Member, IEEE*, and Hai Jin, *Fellow, IEEE*

***Abstract*—As the product of the combination of edge computing and artificial intelligence, edge intelligence (EI) not only solves the problem of insufficient computing capacity of the end device, but also can provide users with various types of intelligent services. However, offline and online model partitioning methods respectively have problems of poor adaptability to the real computing environment and delayed feedback. In addition, previous work on optimizing energy consumption through model partitioning often ignores the latency of intelligent services. Similarly, the energy consumption of end devices and edge servers is usually not considered when optimizing latency. Therefore, we propose game-based adaptive floating-point operations and partition point decision mechanism (GAFPD) to efficiently find the optimal partition point that reduces latency and improves energy efficiency simultaneously in a dynamically changing computing environment. Numerous simulation experiments and robot-based EI system experiments show that GAFPD can simultaneously reduce the latency of intelligent services and improve the energy efficiency of edge devices, while exhibiting strong adaptability to bandwidth changes.**

***Index Terms*—Edge intelligence, model partitioning, latency and energy consumption optimization, dynamically changing computing environment.**

I. Introduction

With the boom of artificial intelligence (AI) applications and services, deep neural network (DNN) [1], as a typical technology with the prominent superiority in deep learning, has been widely applied in various intelligent services, including smart healthcare [2], object detection [3], autonomous driving [4] and so on. At the same time, the rapid development of mobile computing is driving the popularity of end devices, the International Data Center (IDC) predicts that billions of end devices will be connected to the Internet and generate hundreds of millions of bytes of data at the edge of the network by 2025 [5]. Driven by AI and mobile computing, there is an urgent need to push AI to the network edge, so as to make the best use of data at the network edge and explore the computing potential of edge devices (end devices and edge servers). As an emerging computing paradigm that sinks resources and services to the edge of the network, edge computing [6] is undoubtedly a good choice. Under this tendency, the combination of AI and edge computing is inevitable, which has also led to a new cross-research, namely edge intelligence (EI) [7].

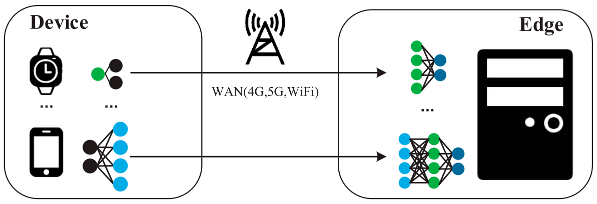
With the popularity and application of DNNs, more and more intelligent services are provided through DNNs in EI. Furthermore, the end device usually offloads computing tasks to the edge server. However, compared with cloud centers, the limited computing resources of edge servers are difficult to satisfy the demands of massive computing tasks represented by DNNs. With the improvement of the computing capacity of the end equipment [8], partitioning the DNN model and offloading part of the computing task to the edge server is a proven solution (Fig. 1(a)). In the process of providing intelligent services, latency is an important factor affecting quality of service (QoS) [9]. Among numerous intelligent services, most of them are latency-sensitive, such as autonomous driving, object detection and so on. Therefore, latency optimization of intelligent services in EI is very important. In addition, one of the obvious drawbacks of the end device is that its energy is limited [10]. In order to be able to provide sustainable and high-quality intelligent services, the energy consumption optimization in EI cannot be ignored.

Studies have shown that partitioning the DNN model by layer can reduce latency or energy consumption [11]. Of course, the selection of partition points is critical when partitioning the DNN model. The existing partition point selection methods are mainly classified into two categories: offline optimization [12], [13], [14], [15] and online learning [16], [17], [18], [19]. The offline optimization [12], [13], [14], [15] approaches treat the selection of partition points as a static global optimization problem, aiming to find the optimal or near-optimal partition point. However, the computing environment where EI provides intelligent services is changing dynamically, offline optimization approaches may be ineffective because they are highly complex and cannot make timely decisions. Although it is possible to obtain the optimal partition point through online learning [16],

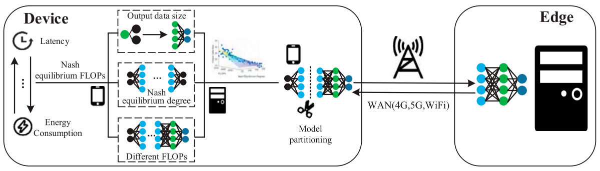
Manuscript received 4 July 2023; revised 28 November 2023; accepted 8 January 2024. Date of publication 15 January 2024; date of current version 13 March 2024. This work was supported by NSFC under Grant 62272179. Recommended for acceptance by N. Fisher. *(Corresponding author: Chen Yu.)*

The authors are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Big Data Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China (e-mail: xinniu@hust.edu.cn; hjy777@hust.edu.cn; wangzw@hust.edu.cn; yuchen@hust.edu.cn; hjin@hust.edu.cn).

Digital Object Identifier 10.1109/TC.2024.3354033



(a) Traditional pipeline for model partitioning.



(b) GAFPD pipeline for model partitioning.

Fig. 1. Comparison between traditional and GAFPD pipeline for model partitioning. The traditional model partitioning only optimizes latency or energy consumption. GAFPD first obtain Nash equilibrium FLOPs with respect to latency and energy consumption. Then combine the output data size, Nash equilibrium degree and FLOPs of partition points to build a regression model. Finally, GAFPD determines the optimal partition point with latency and energy consumption tradeoff.

[17], [18], [19], it takes a period of time for them to learn to get the optimal partition point. In addition, the reduction of latency and the improvement of energy efficiency typically come at the expense of the energy consumption of edge devices and QoS for intelligent services, respectively. Therefore, how to efficiently find the optimal partition point that can reduce latency and improve energy efficiency at same time in a dynamically changing computing environment is currently a key issue in EI that needs to be addressed.

In order to conquer the above challenges, we propose game-based adaptive floating-point operations (FLOPs) and partition point decision mechanism (GAFPD), which consists of game-based adaptive FLOPs decision mechanism (GAFDM) and partition point decision mechanism (PDM). As shown in Fig. 1(b), we demonstrate how GAFPD operates. Firstly, GAFDM obtains the minimum latency and lowest energy consumption of edge devices to complete the computing task at the current moment, along with the corresponding partition points. Moreover, estimate the latency of edge server to complete the computing task offloaded by the end device. Secondly, estimate the latency and energy consumption of the end device to complete the computing task alone. On this basis, gain the ability of model partitioning to reduce latency and improve energy efficiency. Thirdly, obtain the Nash equilibrium FLOPs with respect to latency and energy consumption. Finally, combine the output data size, Nash equilibrium degree and FLOPs of partition points, PDM constructs a regression model and selects the optimal partition point that reduces latency and improves energy efficiency simultaneously. The main contributions of this paper are as follows:

- **Research on DNN models**: We allocate the DNN models to the end device by layer, analyzing the relationship among latency, energy consumption, FLOPs allocated to the end device, and output data size of each partition point. We find that allocating reasonable FLOPs to the end device can reduce latency and improve energy efficiency simultaneously. Furthermore, the impact of the output data size of each partition point on latency and energy consumption cannot be ignored.
- **GAFDM**: Based on the relationship among latency, energy consumption, and FLOPs allocated to the end device, the GAFDM determines the Nash equilibrium FLOPs allocated to the end device through the game with respect to latency and energy consumption.
- **PDM**: Considering the impact of the output data size of partition points on latency and energy consumption, combined with Nash equilibrium degree and FLOPs of partition points, we design the PDM to select the optimal partition point that achieve the tradeoff between latency and energy consumption.
- **GAFPD for theoretical feasibility and simulation efficiency:** Theoretical analysis and numerous performance evaluations show that GAFPD not only theoretically computationally feasible, but also can reduce latency and improve energy efficiency simultaneously.

The reminder of this paper is organized as follows. In Section II, we discuss related work briefly. Section III shows the system model, research on DNN model, problem formulation, and solution in detail, and Section IV gives a detailed description of GAFPD. In Section V, we present various performance evaluation and Section VI draws the conclusion.

II. Related Work

Recently, academia and industry pay more attention to EI. However, while benefiting various fields, there are key issues in EI that need to be addressed, such as latency optimization [15], [17], [20], [21] and energy consumption optimization [13], [16], [22]. Then, we will discuss and analyze the latency and energy consumption optimization in detail.

As an important research direction to improve QoS in EI, latency optimization has attracted the attention of many researchers. While model compression [23], [24] and model early exit [18], [25] can accelerate the DNN inference, these methods result in a loss of accuracy and are not suitable for intelligent services with high accuracy. Therefore, the model partitioning that has no effect on accuracy is a good choice. To address poor real-time performance as well as low quality of user experience in EI, Li et al. [26] proposed the device-edge collaborative inference framework—Edgent, which combined model partitioning and right sizing. The experimental results demonstrated that Edgent can achieve low-latency services in enabling on-demand EI. Xue et al. [27] proposed a DNN inference acceleration offloading scheme based on model partitioning, which optimized the inference latency and reduces the computing pressure on the end device. Meanwhile, Ren et al. [28] proposed an efficient model partitioning method based on deep reinforcement learning, which made the best use of the effective resources of edge devices to efficiently complete computing tasks under the premise of ensuring accuracy. In order to facilitate the partition of DNN models, Lin et al. [21] proposed to convert the

DNN model into graph, and then heuristically assigned partitioned sub-models to available processors. Experimental results demonstrated that the proposed mechanism can achieve the lowest latency compared to other state-of-the-art mechanisms.

The energy consumption optimization is also a key issue in the sustainable development of EI [29], [30]. Considering that the end device is constrained by the limited computing capability, Zeng et al. [31] designed the CoEdge, which utilized available computation resources of edge devices and partitioned the DNN model. Furthermore, the CoEdge achieved at least 25.5% energy savings. While optimizing the network structure of DNNs can reduce energy consumption [32], it inevitably resulted in a decline in the overall performance of DNNs. In order to ensure the performance of DNNs, Xue et al. [33] combined model partitioning and computing offloading to design a low-energy-efficient strategy, which was a good solution to the problem that low-battery capacity end devices cannot support efficient DNNs inference. Ghasemi et al. [34] proposed a framework for edge servers collaborate with end devices to provide intelligent services, which employed Markov decision process to determine which device completes the computing task at each layer of a DNN model. The framework not only met the demand of latency of intelligent services, but also effectively reduced system energy consumption. Aiming at the complex problems of computing task offloading, collaborative computing, and resource allocation in EI, Tan et al. [35] formulated a non-convex mixed integer optimization problem and used reinforcement learning to solve it, which minimized the energy consumption of the computing task.

A careful investigation of the above works finds that they have the following limitations: (1) many researches, e.g., [18], [24], [27], [28], [32], [33], [34], have the problem of delayed feedback in optimizing latency or energy consumption. These methods may be ineffective when dealing with real-world systems with real-time connectivity and dynamically changing computing environments, because they are not able to make optimal or near-optimal decisions in a timely manner. (2) various existing researches, e.g., [18], [24], [27], [28], the energy consumption of edge devices is often ignored when optimizing latency. Similarly, [32], [33], [34], latency is not noticed during energy consumption optimization. Aiming at the above problems, we propose the GAFPD that is capable of efficiently selecting the optimal partitioning point in dynamically changing computing environments to reduce latency and improve energy efficiency simultaneously.

III. System Model, Research on DNN Models, Problem Formulation and Solution

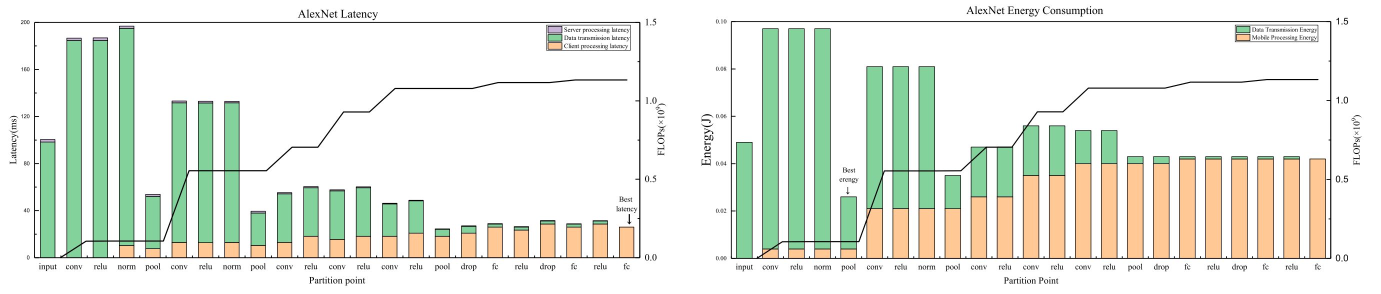
A. System Model

In the EI, the edge server S is placed alongside a base station. Within the coverage of cellular network, S federates m end devices $U = \{u_1, u_2, ..., u_m\}$ to provide intelligent services to users, where one end device corresponds to one user. Due to different intelligent services adopt different DNN models, the FLOPs that edge devices need to complete are different. For a DNN model with n layers $L = \{l_1, l_2, ..., l_n\}$, there are $n+1$ partition points $N = \{0, 1, 2, ..., n\}$. We use $D = \{d_1, d_2, ..., d_n\}$ to denote the output data size of each layer of the DNN model. For each $l_k \in L (1 \leq k \leq n)$, it outputs the data of size d_k. When partitioning the DNN model, any partition point $s \in N (0 < s < n)$ partitions the DNN model into two parts, where u_i complete the computing tasks from layer 1 to s and the result of layer s is transmitted to S, which completes the computing tasks from layer $s+1$ to n. In particular, $s = 0$ indicates that the computing tasks are completely completed by S and $s = n$ indicates that the computing tasks are completely completed by u_i.

TABLE I
Main Notations

Notation	Meaning
S	The edge server.
L	The set of DNNs model layers.
l_k	The k-th layer of the DNNs model.
N	The set of partition points.
s	The partition point.
D	The set of output data size.
d_k, F_k	The output data size and FLOPs of l_k.
U	The set of end devices.
m, n	The number of end devices and layers.
ς	The effective switching capacitance.
f_i^u, f_i^S	The computing resource of u_i and S.
F_T, F_E	The FLOPs allocated to u_i when latency and energy consumption are lowest.
α, β	The ability of model partitioning to reduce latency and improve energy efficiency.
P_i, R_i^t	The transmission power and transmission rate of u_i.
B, h_i, N_0	The channel bandwidth, channel gain, and noise power.
E_i^u, E_i^t	The energy consumption of u_i to complete computing task and transmit data.
E_i^S	The inference energy consumption of S to complete computing task offloaded by u_i.
T_i	The latency of computing task allocated to u_i.
t_i^u, t_i^S	The inference latency of u_i and S.
t_i^t	The transmission latency of u_i.
x_k, y_k	The boolean indicator of whether the computing task of l_k is allocated to u_i and S.
C_i^u, C_i^S	The number of computing resources consumed by u_i and S to complete one FLOP.
H	The case that the computing task cannot be completed.
h_T, h_E	The latency-optimal and energy consumption-optimal FLOPs that allocated to the end device when the computing task cannot be completed.
s_T, s_E	The partition points with satisfactory latency and energy consumption.
z_T, z_E	The mapping functions that determine the FLOPs that allocated to the end device with satisfactory latency and energy consumption.
$q(s)$	The partition function that determines the FLOPs allocated to the player.
f_s	The Nash equilibrium degree of the partition point s.
P_i^f	The regression model of u_i evaluates the fitness of the partition point.

After partitioning the DNN model, the energy consumption to complete the computing task consists of three main components: the inference energy consumption E_i^u of u_i, the transmission energy consumption E_i^t of u_i, and the inference energy consumption E_i^S of S. Generally, the unit of energy consumption is Joule (J). When u_i complete the computing task, we use f_i^u to denote the computing resources (i.e., the number of clock cycles per second of the CPU or GPU) that



(a) Latency and FLOPs breakdown.

(b) Energy consumption and FLOPs breakdown.

Fig. 2. Latency, energy consumption and FLOPs of AlexNet at different partition points.

the end device can provide. The energy consumption per unit of computing resources is $\varsigma(f_i^u)^2$, where ς is the effective switching capacitance depending on the chip architecture [36]. And the unit of ς is Farad (F). F_k is the FLOPs of layer k, and the number of computing resources consumed by u_i to complete one FLOP is C_i^u. Then the inference energy consumption of u_i is

$$E_i^u = \sum_{k=1}^{s} F_k C_i^u \varsigma (f_i^u)^2 \tag{1}$$

In this paper, we utilize orthogonal frequency division multiple access (OFDMA) to enable the communication among edge devices [37]. There are a total of m end devices, and OFDMA divides the total bandwidth B into m orthogonal sub-channels of size B/m, denoted by $\overline{B}$. When $U = \{u_1, u_2, ..., u_m\}$ communicate with S, S allocates the m sub-channels to $U = \{u_1, u_2, ..., u_m\}$. For the computing task that u_i offloads to S, we use P_i and h_i to respectively denote the transmission power and channel gain of u_i communicating with S. Due to the interference among different sub-channels [36], then the transmission rate is

$$R_i^t = \overline{B}\log_2\left(1 + \frac{P_i h_i}{N_0 + \sum_{u_j \in U\setminus\{u_i\}} P_j h_j}\right) \tag{2}$$

where N_0 is the background noise power. The units of B, P_i, N_0, and R_i^t respectively are Hertz (Hz), Watt (W), W, and bit/s (bps).

According to equation (2), the transmission energy consumption E_i^t is

$$E_i^t = P_i d_k / R_i^t \tag{3}$$

where d_k denotes the output data size of l_k.

For the edge server S, the computing resource it can provides is f_i^S, and the computing resource consumed by S to complete one FLOP is C_i^S. Then its energy consumption to complete the offloaded computing task is

$$E_i^S = \sum_{k=s+1}^{n} F_k C_i^S \varsigma (f_i^S)^2 \tag{4}$$

Here, we focus on the energy consumed by u_i, including E_i^u and E_i^t. We can obtain the energy consumption of u_i

$$E_i = E_i^u + E_i^t \tag{5}$$

Similarly, the latency of edge devices to complete the computing task consists of three main components: the inference latency of u_i, the transmission latency, and the inference latency of S. The inference latency of u_i and S to complete the computing task at the l_k layer respectively are $t_{i,k}^u = F_k/(C_i^u f_i^u)$ and $t_{i,k}^S = F_k/(C_i^S f_i^S)$. Furthermore, the computing resources owned by u_i and S determine the inference latency. The output data size of l_k is d_k. If the partition point is $s = k$, then the transmission latency is $t_{i,k}^t = d_k/R_i^t$. Therefore, the latency to complete the computing task is

$$T_i = \sum_{k=1}^{n} x_k t_{i,k}^u + (1 - x_k) t_{i,k}^S + y_k t_{i,k}^t \tag{6}$$

where $x_k, y_k \in \{0, 1\}$. When the computing task of l_k is completed by the end device, $x_k = 1$, otherwise, $x_k = 0$. If $s \leqslant k$, we have $y_k = 0$, otherwise $y_k = 1$.

B. Research on DNN Models

Here, taking AlexNet [38] as an example, we allocate AlexNet to the end device by layer. Then, we analyze the relationship among latency, energy consumption, and the FLOPs allocated to the end device. About the edge devices, we select the NUC with 2.80GHz 8× 11th Gen Intel(R) Core(TM) i7-1165G7 as the end device[1], and we use the device equipped with the 32G NVIDIA TESLA V100 as the edge server platform. The edge devices use the AlexNet to perform image classification over CIFAR-10 [39].

Latency and energy consumption characteristics of DNN models by layer—As shown in Fig. 2, each histogram in Fig. 2(a) and Fig. 2(b) respectively represents the latency and the energy consumption of end-to-edge when partitioning the AlexNet at different layer. In Fig. 2, the leftmost and rightmost histograms respectively represent the scenarios where the edge server and the end device complete the computing task alone. For convenience, we abbreviate them as Edge-only and End-only. Furthermore, we also use broken line to represent the FLOPs allocated to the end device in Fig. 2. With the amount of FLOPs allocated to the end device increases, the inference latency of the end device is gradually increasing. Meanwhile, the energy consumption of the end device is also

[1]https://www.intel.com/content/www/us/en/products/docs/boards-kits/nuc/edge-compute.html

gradually increasing. The latency and the energy consumption of transmitting data are different when the partition points are different, the main reason is that the output data size of AlexNet varies at different partition points. Therefore, if you want to optimize latency and energy consumption, you need to allocate appropriate FLOPs to the end device. Besides, you also should consider the output data size at each partition point, because it determines the transmission latency and energy consumption.

Key observations—(1) The latency and the energy consumption of transmitting data respectively are the primary factors determining overall latency and energy consumption; (2) Although the edge server has significant computing advantages compared to the end device, the impact of data transmission results in the End-only sometimes having lower latency and higher energy consumption than Edge-only; (3) The appropriate FLOPs allocated to the end device can reduce latency and improve energy efficiency of edge devices.

C. Problem Formulation

The edge devices want to provide low latency intelligent services to users with as little energy consumption as possible in EI, but the partition points with minimum latency and lowest energy consumption sometimes are different [11]. Therefore, how to obtain the optimal partition point that can reduce latency and improve energy efficiency at the same time is the key problem that needs to be solved at present. Considering that latency and energy consumption are determined by the FLOPs allocated to the end device and the output data size of the partition point. Therefore, we will obtain the optimal partition point that simultaneously reduces latency and improves energy efficiency in the following two steps. Firstly, we find the FLOPs allocated to the end device where the latency and energy consumption reach Nash equilibrium. According to the obtained Nash equilibrium FLOPs, we can obtain the Nash equilibrium degree of each partition point. Secondly, combined with the output data size, Nash equilibrium degree, and FLOPs of partition points, we establish the regression model and select the optimal partition point that reduces latency and improves energy efficiency at same time.

In order to get the Nash equilibrium FLOPs, we define the following problem. Firstly, we define a convex, closed complete subset X on $\mathbb{R}^2$, where (s_T, s_E) denotes the pair of partition points with satisfactory latency and energy consumption. When $s_T = s_E$, it means that the partition points with satisfactory latency and lowest energy consumption are the same, and we add (s_T, s_E) to X. Moreover, we use H to denote the case that the computing task cannot be completed, at this point, the partition points with satisfactory latency and lowest energy consumption are different. Suppose that for any point $X \cup \{H\}$, there are two mapping functions z_T and z_E that determine the FLOPs allocated to the end device, whose corresponding sets are defined as

$$Z = \{(z_T, z_E) : z(s) = F_T, z(s) = F_E, s \in X\} \tag{7}$$

for the H, we have $h = (z_T(H), z_E(H)) = (0, 0)$.

When selecting the partition point, it is assumed that the user is rational, that is, the Nash equilibrium FLOPs can be found. Then, the problem we are going to solve is able to defined as follows:

Definition 1: Bargaining problem. If (Z, h) satisfies the following properties: (1) $h \in Z$; (2) for any (F_T, F_E) in Z, there is always $F_T \geq h_T, F_E \geq h_E$; (3) Z is convex, bounded, and closed. Then (Z, h) is a bargaining problem.

According to Definition 1, find the FLOPs allocated to the end device where the latency and energy consumption satisfactory is a bargaining problem. Nash proposed and proved that every bargaining problem (Z, h) has a corresponding bargaining solution $F^* = (F_T^*, F_E^*)$ [40] and satisfies the following conditions [41],

$$\arg \max_{0 \leq \alpha, \beta \leq 1} (F_T - h_1)^{\alpha} (F_E - h_2)^{\beta} \tag{8}$$

where $(F_T, F_E) \in \mathbb{R}^2, (F_T, F_E) \geq (h_1, h_2)$, α and β respectively denote the ability of model partitioning to reduce latency and improve energy efficiency.

For the ability of model partitioning to reduce latency α, $\alpha = 0$ when the minimum latency T_i of model partitioning is greater than the latency ${T_i}^a$ in which the end device completes the computing task alone. Otherwise, we assign the ratio of T_i to ${T_i}^a$ to α. Similarly, for the ability of model partitioning to reduce energy consumption β, $\beta = 0$ when ${E_i}^t$ is greater than ${E_i}^a$ that the end device completes the computing task alone. Otherwise, we assign the ratio of E_i to ${E_i}^a$ to β. In particular, when a large number of end devices offload computing tasks to the edge server causing the edge server to overload (i.e., $t_i^S > T_i^a$), we set $\alpha = 1$ and $\beta = 0$.

Besides, we count the FLOPs that allocated to the end device and the output data size of each partition point. Then, we establish the regression model P_i^f of u_i to evaluate the suitability of s to u_i. The input to P_i^f consists of F_s that allocated to the end device, the output data size d_s, and the Nash equilibrium degree f_s, where F_s, d_s and f_s are all normalized values. In addition, f_s is the FLOPs difference between the partition point s and the Nash equilibrium FLOPs. Finally, we select the partition point with the smallest P_i^f as the optimal partition point. Therefore, the partition point decision problem can be defined as:

$$\min \quad P_i^f(F_s, d_s, f_s), \tag{9}$$

$$\text{s.t.} \quad 0 \leq s \leq n, \tag{9a}$$

$$0 \leq F_s, d_s, f_s \leq 1. \tag{9b}$$

D. Nash Bargaining Solution

We formulate the problem of finding the FLOPs allocated to the end device that simultaneously optimize latency and energy consumption as a dynamic bargaining game with complete information. The essence of the bargaining game lies in how the players divide the desired item. In this paper, we define $F' = F_2 - F_1$ as the resource that two players want to divide, where $F_1 = \min(F_T, F_E), F_2 = \max(F_T, F_E)$, and $[0, F']$ is the bargaining range for two players. We denote the partition function as $q(s) : [0, F'] \rightarrow \mathbb{R}$, let q_1' and q_2' ($q_1', q_2' \in [0, F']$ and $q_1' \leq q_2'$) respectively represent the bargaining results for the two players. Nash [40], [41] proved that $q_1' + q_2' \leq F'$ and Eq. (7) achieves its maximum value if and only if $q_1' + q_2' = F'$. Thus, we can obtain the following theorem.

Theorem 1: When $\alpha = 1$ and $\beta = 1$, the Nash equilibrium solution of the bargaining problem (Z, h) is $z_1 = z_2 = (F_2 - F_1)/2$ and the maximum value of Eq. (7) is $(F_1 - F_2)^2/4$. Otherwise, the Nash equilibrium solution is $z = F_1 + (1-\beta)(F_2 - F_1)/(1-\alpha\beta)$ and the maximum value of Eq. (7) is $\alpha(1-\beta)^2(F_1 - F_2)^2/(1-\alpha\beta)^2$, where $F_1 = \min(F_T, F_E), F_2 = \max(F_T, F_E)$.

Proof: When Eq. (7) takes the maximum value, we can obtain the Nash equilibrium solution of the game. Due to $(h_1, h_2) = (0, 0)$, we need to obtain the maximum value of $F_T^{\alpha} F_E^{\beta}$ to get the Nash equilibrium solution.

First, we consider the case of $\alpha = \beta = 1$. We need to find a point that maximizes the value of $F_T F_E$, that is, find the values of F_1 and F_2 corresponding to the maximum value of $F_1 F_2$. Since $F' = F_2 - F_1$ and $q_1' + q_2' = F'$, we can obtain

$$(F_T - d_1)^{\alpha}(F_E - d_2)^{\beta} = -\left(z_1 - \frac{F_2 - F_1}{2}\right) + \frac{(F_2 - F_1)^2}{4} \tag{10}$$

Obviously, the maximum value of $(F_T - h_1)^{\alpha}(F_E - h_2)^{\beta}$ is $(F_1 - F_2)^2/4$, and the corresponding value of q_1' is $(F_2 - F_1)/2$. Then the FLOPs that the end device need to complete is $(F_T + F_E)/2$.

Then we discuss the general case where α and β are not simultaneously equal to 1. The two players alternate their bidding, and in each round, each player attempts to reduce the other player's share and increase its own share.

We assume that the two players reach a consensus after r rounds of the game. The share that one player can obtain is q_1', and q_1' is equal to α or β times the result of the previous round of bargaining. In the bargaining game with complete information, the other player knows that the player will be satisfied with the share q_1' received in round r. Besides, the offer of the player in round $r-1$ will not exceed $\alpha q_1'$ or $\beta q_1'$. To maximize the payoff of the first player, the other player of the game will make an offer $(\alpha q_1', 1 - \alpha q_1')$ or $(\beta q_1', 1 - \beta q_1')$ in round $r-1$.

For the other player, it knows that the player who gets $\alpha q_1'$ or $\beta q_1'$ share in round $r-1$ will offer $\beta(1 - \alpha q_1')$ or $\alpha(1 - \beta q_1')$ in $r-2$ round. Therefore, for the other player, it will offer $(1 - \beta(1 - \alpha q_1'), \beta(1 - \alpha q_1'))$ or $(1 - \alpha(1 - \beta q_1'), \alpha(1 - \beta q_1'))$ in round $r-1$, in other words, the share it can get in round $r-1$ is $1 - \beta(1 - \alpha q_1')$ or $1 - \alpha(1 - \beta q_1')$.

Therefore, we can get $1 - \beta(1 - \alpha q_1') = q_1'$ or $1 - \alpha(1 - \beta q_1') = q_1'$, then $q_1' = (1-\beta)/(1-\alpha\beta)$ or $q_1' = (\beta - \alpha\beta)/(1-\alpha\beta)$. Since the range of the game is $[0, F_2 - F_1]$, we can get the result of the game is $F_T = F_1 + (1-\beta)(F_2 - F_1)/(1-\alpha\beta)$ or $F_T = F_2 - (F_2 - F_1)\beta(1-\alpha)/(1-\alpha\beta)$, and the maximum value of $(F_T - d_1)^{\alpha}(F_E - d_2)^{\beta}$ is $\alpha(1-\beta)^2 (F_1 - F_2)^2/(1-\alpha\beta)^2$. □

E. Partition Point Decision Model

Here, we model the ability of each partition point to simultaneously optimize latency and energy consumption. As defined in Section III-C, the partition point for simultaneously reducing latency and improving energy efficiency is determined by F_s, d_s, and f_s jointly. Therefore, for each partition point, we first measure the latency and energy consumption under varied configurable parameters. Then, compared with the end device to complete the computing task alone, we quantify the ability of each partition point to reduce latency and improve energy efficiency. Next, we establish a regression model for partition points to judge their ability to simultaneously optimize latency and energy consumption. We use logarithmic or linear functions as regression functions, and the percentage reduction in latency and energy consumption as performance metrics.

In the regression function, every variable(F_s, d_s, and f_s) plays an indispensable role in assessing the partition point's ability to reduce latency and improve energy efficiency simultaneously. In the Section III-B, we have elaborated the impact of the FLOPs allocated to the end device and the output data size of each partition point on latency and energy consumption. The f_s denotes how far the partition point s deviates from the Nash equilibrium FLOPs. The importance of each of the aforementioned variables can be derived through training.

As previously mentioned, it is an analysis step required for each DNN model to establish the regression model. The established PDM can directly assist edge devices to select the optimal partition point that simultaneously reduces latency and improves energy efficiency without additional overhead.

IV. Game-Based Adaptive FLOPs and Partition Point Decision Mechanism

In order to solve the problem defined in Section III, we propose GAFPD (Fig. 3). As shown in Algorithm 1, for any end device u_i, our proposed mechanisms consist of the following four main steps: (1) Obtain the model partition points with minimum latency and the lowest energy consumption, along with their corresponding partition points. Moreover, estimate the latency of edge server to complete the computing task offloaded by the end device (lines 1-3); (2) Estimate the inference latency and energy consumption when the end device complete the computing task alone, obtain the ability to reduce latency and improve energy efficiency that the edge server collaborate with the end device to complete computing task (lines 4-18); (3) A bargaining game based on the above information is established, then obtain Nash equilibrium FLOPs with satisfactory latency and energy consumption (line 19); (4) Obtain the partition point through PDM (line 20).

A. Game-Based Adaptive FLOPs Decision Mechanism

The purpose of GAFDM is to obtain the Nash equilibrium FLOPs that reduce latency and improve energy efficiency at same time. As shown in Algorithm 2, one player of the game first gives its bid, that is, the FLOPs F_1. The other player gives the FLOPs F_2. In each round of the game, both players aim to reach a Nash equilibrium with respect to FLOPs by making certain concessions until the difference between their strategies is less than or equal $|F_1 - F_2| \leq 10^{-3}$. We discuss the following two cases, when $\alpha = \beta = 1$, the concession function of two

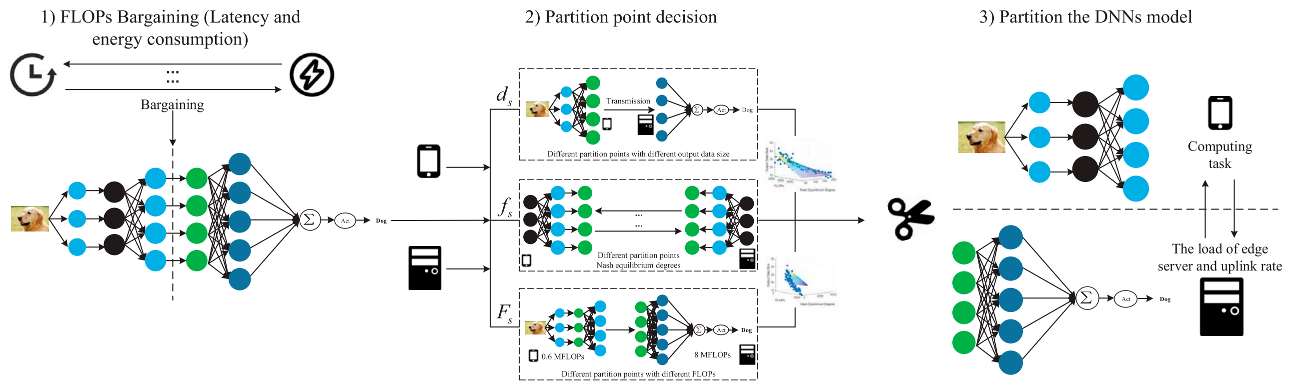

1) FLOPs Bargaining (Latency and energy consumption)
Bargaining
2) Partition point decision
d_s
f_s
F_s
Transmission
Different partition points with different output data size
Different partition points Nash equilibrium degrees
0.6 MFLOPs
8 MFLOPs
Different partition points with different FLOPs
3) Partition the DNNs model
Computing task
The load of edge server and uplink rate
Dog
Act

Fig. 3. Game-based adaptive FLOPs and partition point decision mechanism.

Algorithm 1 Game-based adaptive FLOPs and partition point decision mechanism (GAFPD)

```
Input: The end device $u_i$, the $FLOPs_i$ that $u_i$ need to complete, the output data size $D_i = \{d_i^1, d_i^2, ..., d_i^n\}$ of each partition point of the selected DNNs model, the regression model $P_s^f$.
Output: Partition point $s_i$.
1:  $T_i, s_i^T \leftarrow$ get minimum latency and partition point;
2:  $E_i, s_i^E \leftarrow$ get lowest energy consumption and partition point;
3:  $t_i^S \leftarrow$ get the inference latency of $S$;
4:  $E_i^a, T_i^a \leftarrow$ get energy consumption and latency when $u_i$ completes the inference task alone;
5:  if $t_i^S > T_i^a$ then
6:      $\alpha \leftarrow 1, \beta \leftarrow 0$;
7:  else
8:      if $T_i > T_i^a$ then
9:          $\alpha \leftarrow 0$;
10:     else
11:         $\alpha \leftarrow 1 - T_i/T_i^a$;
12:     end if
13:     if $E_i^t > E_i^a$ then
14:         $\beta \leftarrow 0$;
15:     else
16:         $\beta \leftarrow 1 - E_i/E_i^a$;
17:     end if
18: end if
19: $F_i^N \leftarrow GAFDM(\alpha, \beta, FLOPs_i, E_i, s_i^E, T_i, s_i^T)$;
20: $s_i \leftarrow PDM(FLOPs_i, D_i, F_i^N, P_s^f, s_i^E, s_i^T)$;
21: return $s_i$.
```

players of the game is $f(r) = F'/[(\alpha+\beta) + e^{-2(r-1)}]$, where r denotes the number of rounds in the game, and $F' = F_2 - F_1$ in every round of the bargaining process. In particular, when $0 \le \alpha < 1$ and $0 \le \alpha < 1$, after one round of the game, the two players reach an agreement on the FLOPs value according to Theorem 1, resulting in $F_1 = F_1 + F'(1-\beta)/(1-\alpha\beta)$. Finally, the two players determine the Nash equilibrium FLOPs for u_i with satisfactory latency and energy consumption, which is $F_1 + \min(F_E, F_T)$.

Algorithm 2 Gamed-based adaptive FLOPs decision mechanism (GAFDM)

```
Input: $\alpha$, $\beta$, $E_i$, $s_i^E$, $T_i$, $s_i^T$, and $FLOPs_i$.
Output: Nash equilibrium FLOPs $F_1 + \min(F_E, F_T)$.
1:  $F_1 = 0$;  $F_2 = \max(F_E, F_T) - \min(F_E, F_T)$;  $F' = F_2 - F_1$; $r = 1$;
2:  while $\Delta_1, \Delta_2 \leftarrow MAX$ to $\Delta_1 \le 10^{-3}$ or $\Delta_2 \le 10^{-3}$ do
3:      if $\alpha = 1$ and $\beta = 1$ then
4:          Calculate $F_1 = F_1 + f(r)$, $F_2 = F_2 - f(r)$ where $f(r) = F'/[(\alpha+\beta) + e^{-2(r-1)}]$ and $\Delta_2 = |F_1 F_2 - (F_E - F_T)^2/4|$;
5:      else
6:          $F_1 = F_1 + F'(1-\beta)/(1-\alpha\beta)$;
7:          $F_2 = F_2 - F'\beta(1-\alpha)/(1-\alpha\beta)$;
8:      end if
9:      $\Delta_1 = |F_1 - F_2|$, $F' = F_2 - F_1$, $r = r + 1$;
10: end while
11: return $F_1 + \min(F_E, F_T)$.
```

B. Partition Point Decision Mechanism

As shown in Algorithm 3, after getting the Nash equilibrium FLOPs allocated to the end device u_i, we first calculate the Nash equilibrium degree for the partition points between s_i^E and s_i^T. Then, we calculate the ratio of each partition point's Nash equilibrium degree to the maximum Nash equilibrium degree among these partition points. Subsequently, we obtain the normalized Nash equilibrium degree for each partition points. Similarly, we normalize the FLOPs allocated to u_i and the output data size at each partition point between s_i^E and s_i^T. Next, we use the regression model $P_{i,j}^f$ to calculate the fitness for each partition point. Finally, we sort $P_{i,j}^f$ in non-decreasing order and select the point $s_i^{s'}$ with the minimum $P_{i,e'}^f$ as the final partition point.

C. A Working Example

We illustrate how GAFDM determines the Nash equilibrium FLOPs and PDM determines the final partition point through the following example. There are four computing tasks that require the end device and edge server to complete, the FLOPs allocated to the end device with the lowest energy consumption

Algorithm 3 Partition point decision mechanism (PDM)

```
Input: $FLOPs_i$, $D_i$, $F_i^N$, $P_s^f$, $s_i^E$, $s_i^T$.
Output: Partition point $s_i$.
1:  $s_i^s = \min(s_i^E, s_i^t)$, $s_i^e = \max(s_i^E, s_i^t)$;
2:  for each $s_i^j$ in $(s_i^s, s_i^e)$ do
3:      $f_i^j = |F_i^j - F_i^N|$;
4:  end for
5:  Normalize of $\{F_i^s, F_i^{s+1}, ...., F_i^e\}$, $\{d_i^s, d_i^{s+1}, ...., d_i^e\}$ and
    $\{f_i^s, f_i^{s+1}, ...., f_i^e\}$;
6:  for each $s_i^j$ in $(s_i^s, s_i^e)$ do
7:      $P_{i,j}^f \leftarrow P_i^f(F_i^j, d_i^j, f_i^j)$;
8:  end for
9:  Sort $P_{i,j}^f$ according to $P_{i,s'}^f \leq P_{i,s'+1}^f \leq ... \leq P_{i,e'}^f$;
10: $s_i \leftarrow s_i^{s'}$;
11: return $s_i$.
```

TABLE II
RELATED INFORMATION AND RESULTS ABOUT BARGAINING

Task	$F_E(FLOPs)$	$F_T(FLOPs)$	(α, β)	Bargaining reslut
1	1 M	0.5 M	(1,1)	0.75 M
2	1.5 M	1 M	(1,0.8)	1 M
3	1 M	0.5 M	(0.5,1)	1 M
4	2 M	1 M	(1, 1)	1.5 M

and minimum latency are presented in Table II. Besides, we use the binary group (α, β) to denote the ability of model partitioning to reduce latency and energy consumption. After obtaining the Nash equilibrium FLOPs, we use the PDM to determine the partition point, and the required information is presented in Table III. In Table II and Table III, task refers to computing task.

As shown in Table II, we take the computing task 1 as an example to illustrate the bargaining, the FLOPs of the end device with the lowest energy consumption and minimum latency respectively are 1 M and 0.5 M. The two players will play the game on the FLOPs that allocated to the end device, and the concession function of the game is $f(r) = 0.5/(2 + e^{-2(r-1)})$. First, the player with the lowest energy consumption proposes that the FLOPs allocated to the end device are $5/6$ M, while the acceptable FLOPs for the player with the minimum latency are $2/3$ M. Therefore, the two players cannot reach an agreement and need to continue the game. According to Algorithm 2, after a certain number of rounds of the game, the two players finally reach an agreement on 0.75 M. As for the computing task 2, because α and β are not simultaneously equal to 1, they reach an agreement on 1 M after one round of game according to Theorem 1. Similarly, for the computing tasks 3 and 4, the final results of the games respectively are 1 M and 1.5 M.

As shown in Table III, taking the computing task 1 as an example, we illustrate how to determine the final partition point. First, we calculate the Nash equilibrium degree of the partition points from the minimum latency to the lowest energy consumption, and the Nash equilibrium degree of partition points 1, 2, and 3 of computing task 1 respectively are 0.25, 0, and 0.25. Then, we normalize the FLOPs, Nash equalization degree, and output data size of the partition points. The normalized results of the above three values corresponding to the partition points 1, 2, and 3 of computing task 1 respectively are (0.22, 0.33, 0.45), (0.5, 0, 0.5), and (0.45, 0.1, 0.45). Finally, using the regression model to evaluates the fitness of each partition point. Here, we assume that the regression model is $P_1^f = F_1 + f_1 + d_1$, where F_1, f_1, and d_1 respectively represent normalized results of FLOPs, Nash equilibrium degree, and output data size. The fitness of partition points 1, 2, and 3 of computing task 1 are (1.15, 0.43, 1.4). Therefore, the final partition point of computing task 1 is 2. Similarly, we can get that the final partition point of computing task 2, 3, and 4 respectively are 1, 2, and 2.

TABLE III
RELATED INFORMATION AND RESULTS ABOUT PDM

Partition point	FLOPs(M)				Output data size			
	Task 1	Task 2	Task 3	Task 4	Task 1	Task 2	Task 3	Task 4
0	0	0	0	0	0.25	0.25	0.25	0.25
1	0.5	1	0.5	1	1	1	0.75	2
2	0.75	1.5	1	1.5	0.25	1.5	1.25	2.5
3	1	2.5	1.5	2	1	2.5	2	3

D. Theoretical Analysis

For any end device u_i and edge server S, the computational complexity of getting $T_i, s_i^T, E_i, s_i^E, E_i^a, T_i^a$, and T_i^S is constant. In addition, the computational complexity of obtaining the ability to reduce latency and improve energy efficiency that edge devices complete computing task is also constant. Subsequently, we obtain Nash equilibrium FLOPs with satisfactory latency and energy consumption through gaming. When $\alpha = 1$ and $\beta = 1$, we can get the Nash equilibrium FLOPs in ten rounds of game. Otherwise, we can get the Nash equilibrium FLOPs in one round of game. In other words, the computational complexity of u_i obtains Nash equilibrium FLOPs is constant. Then, we calculate and normalize the Nash equilibrium degree of the partition points between s_i^E and s_i^T, the computational complexity is $O(n)$. Similarly, the computational complexity of normalizing the FLOPs allocated to u_i and the output data size at each partition point between s_i^E and s_i^T is also $O(n)$. Next, we calculate the fitness of the aforementioned partition points through the regression model $P_{i,j}^f$ and sort the results, the computational complexity is $O(n \cdot logn)$. There are a total of m end devices in the coverage of S. Therefore, the computational complexity of selecting partition points for edge devices is $O(mn \cdot logn)$.

In summary, our proposed GAFPD can select the optimal partition point that reduces latency and improves energy efficiency with polynomial complexity, that is, it is computationally feasible.

V. Experiment Evaluations

In this section, we first give a brief introduction to the experimental settings, mainly including experimental parameters, DNN models, dataset, benchmarks and so on. Then, we verify the feasibility of our proposed mechanism through a series of simulation experiments. Here, we mainly focus on the optimization of latency and energy consumption, adaptability to

TABLE IV
LATENCY SPEEDUP AND ENERGY CONSUMPTION SAVE OF DIFFERENT MECHANISMS COMPARED TO END-ONLY

	ResNet18		MobileNet		VGG16	
	Latency	Energy consumption	Latency	Energy consumption	Latency	Energy consumption
End-only	**200 ms**	**0.203 J**	**100 ms**	**0.173 J**	**610 ms**	**2.296 J**
Edge-only	+94.406%	-36.946%	+93.406%	-63.006%	-52.113%	-95.818%
Neurosurgeon-L	0	0	0	0	-52.113%	-95.819%
Neurosurgeon-E	+94.406%	-36.946%	+93.406%	-63.006%	-52.113%	-95.819%
GAFPD	+29.568%	-14.286%	+1.806%	-50.867%	-52.113%	-95.819%

changes in bandwidth, the impact of the number of end devices on performance, and the impact of the transmission power on performance. Finally, we deploy GAFPD on an EI system based on robots to verify its effectiveness.

A. Experiment Settings

We consider the scenario where a single server combines multiple end devices to provide intelligent services to users. In this scenario, the number of end devices are randomly distributed in [1,100], and the bandwidth resources of edge devices is $B = 40$ MHz. The computing task can be completed by the end device with $f_i^u = [0.1, 2.1]$ GHz, and the energy consumption per unit of computing resource is $\varsigma(f_i^u)^2 = [0, 4 \times 10^{-9}]$ J. When the end device offloads the computing task to the edge server, the transmission power is $P_i = 0.1$ W, the noise power $N_0 = 10^{-9}$ W, the channel gain h_i is uniformly distributed in $[10^{-5}, 10^{-3}]$, and the transmission rate is $R_i^t = [0, 5]$ Mb/s [42]. Then, the edge server completes the offloaded computing task with $f_i^S = [0.1, 2.1]$ GHz, and the energy consumption per unit of computing resource is $\varsigma(f_i^u)^2 = [0, 1.5 \times 10^{-9}]$ J. In addition, the number of computing resources consumed by the end device and edge server to complete one FLOP respectively are $C_i^u = 1/32$ and $C_i^S = 1/32$. That is, the end device and edge server are capable of performing 32 single-precision floating-point calculations per clock cycle.

Then, we select three DNNs models (ReseNet18 [43], MobileNet [44], and VGG16 [45]) to evaluate the performance of GAFPD for image classification over CIFAR-10 [39] on the machine learning platform Pytorch. Moreover, we select the following benchmarks for comparison.

Benchmarks. (1) End-only: All computing tasks are completed by the end device. (2) Edge-only: The end device offloads all computing tasks to the edge server. (3) Neurosurgeon with optimal latency (Neurosurgeon-L) [11]: The Neurosurgeon selects the partition point with optimal latency. (4) Neurosurgeon with optimal energy consumption (Neurosurgeon-E) [11]: The Neurosurgeon selects the partition point with optimal energy consumption. (5) Autodidactic neurosurgeon (ANS) [17]: The ANS obtains the partition point with optimal latency through the built-in learning module.

B. Simulation Results

In this subsection, we analyze the experimental results. It is important to note that the real execution time of the GAFPD and benchmarks is between 0 and 1 ms. Compared to the time taken by edge devices edge devices to complete computing tasks, the real execution time of each mechanism is negligible. In other words, the real execution time of each mechanism does not affect the selection of partition points.

TABLE V
LATENCY SPEEDUP AND ENERGY CONSUMPTION SAVE OF DIFFERENT MECHANISMS COMPARED TO NEUROSURGEON-L

	ResNet18		MobileNet		VGG16	
	Latency	Energy consumption	Latency	Energy consumption	Latency	Energy consumption
Neurosurgeon-L	**200 ms**	**0.203 J**	**100 ms**	**0.173 J**	**292.109 ms**	**0.096 J**
End-only	0	0	0	0	+108.826%	+2291.667%
Edge-only	+94.406%	-36.946%	+93.406%	-63.006%	0	0
Neurosurgeon-E	+94.406%	-36.946%	+93.406%	-63.006%	0	0
GAFPD	+29.568%	-14.286%	+1.806%	-50.867%	0	0

TABLE VI
LATENCY SPEEDUP AND ENERGY CONSUMPTION SAVE OF DIFFERENT MECHANISMS COMPARED TO NEUROSURGEON-E

	ResNet18		MobileNet		VGG16	
	Latency	Energy consumption	Latency	Energy consumption	Latency	Energy consumption
Neurosurgeon-E	**388.812 ms**	**0.128 J**	**193.406 ms**	**0.064 J**	**292.109 ms**	**0.096 J**
End-only	-48.561%	+58.594%	-48.295%	+170.312%	+108.826%	+2291.667%
Edge-only	0	0	0	0	0	0
Neurosurgeon-L	-48.561%	+58.593%	-48.295%	+170.312%	0	0
GAFPD	-33.352%	+35.938%	-47.362%	+32.812%	0	0

1) Latency and energy consumption: We conduct a series of simulation experiments based on the above experiment settings. We select End-only, Neurosurgeon-L, and Neurosurgeon-E as baselines, and the experimental results are shown in Tables IV, V, and VI. In the Tables, the bold data in the first row represents the baseline, and we have recorded its latency and energy consumption. Subsequently, we record the percentage improvement in latency and energy consumption compared to the baseline for different mechanisms. Among them, positive numbers indicate the percentage increase in latency and energy consumption compared to the baseline for different mechanisms, while negative numbers indicate the percentage reduction. Now, we proceed to analyze the experimental results.

In Table VI, for lightweight DNN models, such as ResNet18 and MobileNet, Neurosurgeon-L has similar latency and energy consumption compared to End-only. This is due to the end device's computing capacity being efficiently handle the computing tasks of these lightweight DNN models. We also find that the ratio of the increase in latency of GAFPD to the decrease in energy consumption is 2.07 for ResNet18, while the ratios for Edge-only and Neurosurgeon-E are 2.56. Similarly, for MobileNet, GAFPD is able to trade a 1.806% increase in latency for a 50.867% reduction in energy consumption. In other words, GAFPD can achieve greater energy consumption savings with a lower incremental cost in latency. For DNN models with high complexity, such as VGG16, completely offloading computing tasks to the edge server is undoubtedly a superior option.

As mentioned above, in Table V, for lightweight DNN models, Neurosurgeon-L has similar latency and energy consumption compared to End-only. Therefore, we do not present repeated comparisons between Neurosurgeon-L and other mechanisms. For VGG16 with high complexity, the latency and energy consumption of GAFPD are comparable to those of Neurosurgeon-L, indicating that GAFPD is feasible.

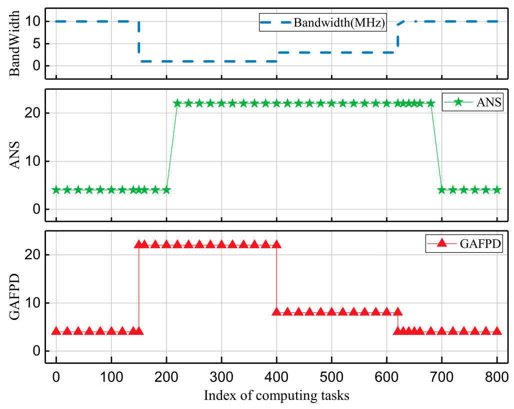


Fig. 4. Partition point selection of GAFPD and ANS under changing bandwidths.

In Table VI, for ResNet18 and MobileNet, when compared to Neurosurgeon-E, GAFPD demonstrates a reduction in latency, albeit at the cost of increased energy consumption. And compared with End-only and Neurosurgeon-L, GAFPD shows the smallest percentage increase in energy consumption. Finally, for VGG16 with high complexity, the latency and energy consumption of GAFPD are the same as those of Neurosurgeon-E, and the selected partition point is optimal, which shows that GAFPD is feasible.

2) Impact of bandwidth: The above experimental results prove that GAFPD can reduce latency and improve energy efficiency at same time. Furthermore, to prove the adaptability of GAFPD to changing bandwidths, we perform the image classification task over CIFAR-10 with AlexNet to evaluate the adaptability of GAFPD when the bandwidth changes.

First of all, we compare the performance of GAFPD and ANS under changing bandwidth conditions, the experimental results are shown in Fig. 4. When the bandwidth is 10 MHz, the partition point selected by GAFPD and ANS for AlexNet is 4. When the bandwidth is reduced from 10 MHz to 1 MHz, GAFPD quickly adjusts the partition point to 22, resulting in a latency of 78 ms and an energy consumption of 0.105 J. However, the partition point selected by ANS remains at 4, resulting in a latency of 278.817 ms and energy consumption of 0.099 J. Meanwhile, it takes a period of adaptation for ANS to adjust the partition point to 22. Compared to ANS, GAFPD achieves a reduction of 72.02% in latency at the expense of only 5.71% increase in energy consumption. Subsequently, when the bandwidth increases from 1 MHz to 3 MHz, GAFPD promptly adjusts the partition point to 8, resulting in a latency of 95.752 ms and an energy consumption of 0.07 J. At this point, ANS has adjusted the division point to 22 through online learning, and GAFPD can sacrifice a 22.72% increase in latency for a 33.33% reduction in energy consumption. Finally, when the bandwidth increases from 3 MHz to 10 MHz, the selected partition point for GAFPD is 0, resulting in a latency of 38.545 ms and an energy consumption of 0.019 J. On the other hand, the latency of ANS is 78 ms, while its energy consumption is 0.105 J. Overall, when the bandwidth changes, GAFPD not only outperforms ANS, but also solves the problem of delayed feedback of ANS.

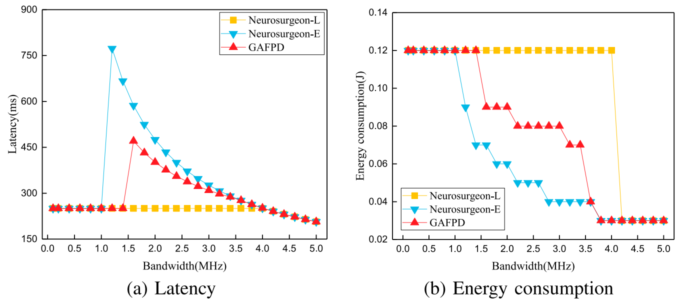


Fig. 5. Latency and energy consumption of AlexNet under different bandwidths.

Furthermore, we test the ability of different mechanisms to adapt to changing bandwidth. According to the analysis of the results of Tables VI, V, and IV, it can be concluded that for lightweight DNN models, the latency and energy consumed of Neurosurgeon-L are the same as those of End-only. And Neurosurgeon-E has the same latency and energy consumption as Edge-only. Therefore, we choose Neurosurgeon-L, Neurosurgeon-E, and GAFPD to partition the AlexNet when the bandwidth varies between 0.1 MHz to 5 MHz. The results of these experiments are shown in Fig. 5.

In Fig. 5, Fig. 5(a) and Fig. 5(b) respectively depict the latency and energy consumption of different model partitioning mechanisms under different bandwidths. When the bandwidth is greater than 1 MHz, the latency of Neurosurgeon-E increases rapidly, and simultaneously, its the energy consumption decreases rapidly. With the increase of bandwidth, the energy consumption of data transmission becomes lower than that of the end device completing the computing task alone. So, the end device offload the computing task to the edge server. However, the latency introduced by data transmission is much greater than that of the end device completing computing tasks alone. With the increase of bandwidth, latency generated by data transmission gradually decreases, and both the latency and energy consumption of Neurosurgeon-E gradually decrease. When the bandwidth is greater than 3.5 MHz, the latency and energy consumption of Neurosurgeon-E become comparable to those of GAFPD. When the bandwidth is greater than 4.0 MHz, the latency and energy consumption of the three model partitioning mechanisms (Neurosurgeon-L, Neurosurgeon-E, and GAFPD) become comparable, indicating that they have chosen a similar partition point.

Based on the simulation results above, it is evident that GAFPD takes into account the impact of latency and energy consumption on edge devices in selecting the partition point as the bandwidth changes. In addition, when the bandwidth is greater than 1.5 MHz and less than 3.5 MHz, GAFPD prioritizes energy efficiency over latency, making a strategic tradeoff between the two. When latency is not a critical requirement for the computing task, it is a more energy-efficient choice for end devices with limited energy. based on the results of simulation experiments, it can be observed that GAFPD adaptively selects the overall optimal partition point in response to changes in bandwidth.

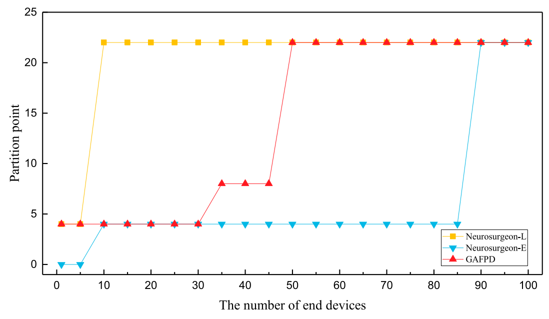
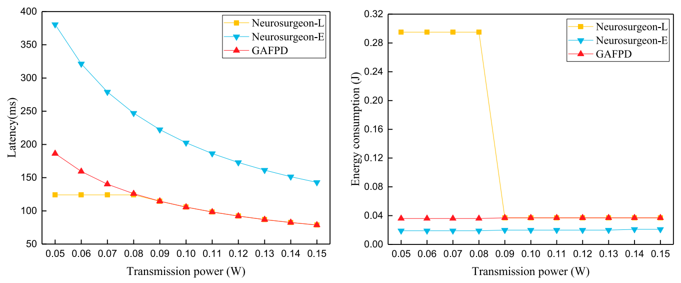


Fig. 6. Partition point selection of different mechanisms under different number of end devices.



(a) Latency (b) Energy consumption

Fig. 7. Latency and energy consumption of AlexNet under different transmission powers.

3) Impact of the number of end devices: In order to evaluate the impact of network scale on the performance of GAFPD. We begin with an initial number of end devices set at 1 and increment the number of end devices by 5 each time until the total number of end devices reaches 100. Then, we record the partition points selected by different mechanisms as the number of end devices changes. The experimental results are presented in Fig. 6.

According to Fig. 6, when the number of end devices is less than 30, the partition point selected by GAFPD corresponds to that of Neurosurgeon-L or Neurosurgeon-E. The main reason is that the goal of GAFPD is to select the optimal partition point that can simultaneously reduce latency and improve energy efficiency. That is, GAFPD aims to achieve greater energy consumption reduction or latency reduction, while limiting the increase of latency or energy consumption. When the number of end devices ranges between 30 and 45, the allocated bandwidth per end device diminishes as the number of end devices increases. Therefore, GAFPD selects the 8-th layer of AlexNet with smaller intermediate layer output as the partition point for the newly connected end device to achieve the tradeoff between latency and energy consumption. When the number of end devices exceeds than 45, the high load of the edge server causes higher inference latency. Moreover, communication between edge devices results in increased latency and energy consumption. Therefore, GAFPD allocates the computing task to the end device to achieve the tradeoff between latency and energy consumption. When the number of end devices exceeds 85, Neurosurgeon-L, Neurosurgeon-E, and GAFPD all choose to allocate the computing task to the end device. The main reason is that the latency and energy consumption of offloading computing tasks to the edge server are much higher than those of end devices to complete computing tasks alone.

4) Impact of the transmission power: In order to evaluate the impact of transmission power changes on performance, we initiate the transmission power of the end device at 0.05 W and gradually increase it by increments of 0.01 W until it reaches 0.15 W. Similarly, we perform the image classification task over CIFAR-10 with AlexNet. Then, we compare the latency and energy consumption of Neurosurgeon-L, Neurosurgeon-E, and GAFPD to when completing the computing task. The experimental results are shown in Fig. 7.

From the Fig. 7, when the transmission power is less than 0.09 W, the latency of Neurosurgeon-L is less than that of Neurosurgeon-E and GAFPD. And the energy consumption of Neurosurgeon-L is higher than that of Neurosurgeon-E and GAFPD. The main reason is that when the transmission power is low, the communication latency between the end device and the edge server is much greater than the latency that the end device completes the computing task alone. However, the energy consumption of data transmission is less than that of the end device to complete the computing task alone. As the transmission power increases, the communication latency between edge devices gradually decreases. Therefore, when the transmission power is greater than 0.09 W, the latency of Neurosurgeon-L and GAFPD is equal and lower than that of Neurosurgeon-E, i.e., the partition points selected by Neurosurgeon-L and GAFPD is the same. At the same time, the energy consumption of Neurosurgeon-L and GAFPD is equal and higher than that of Neurosurgeon-E.

With the change of transmission powers, GAFPD comprehensively considers the impact of latency and energy consumption on edge devices to select partition points. In particular, when the transmission power is greater than 0.09 W, GAFPD is able to exchange a 43.24% increase in energy consumption for an 81.83% reduction in latency. Therefore, we can conclude that the GAFPD is able to select the optimal partition point to reduce latency and improve energy efficiency as the transmission power changes.

C. Experimental Results on EI System

We deploy GAFPD on an EI system based on the RoboMaster University AI Challenge (RMUA)[2], and then evaluate its efficacy in terms of latency reduction and energy efficiency improvement. The EI system based on the RMUA platform is shown in Fig. 8, which is composed of end devices, a router, and an edge server. Furthermore, we briefly illustrate the completion process of the computing task together with Fig. 8. The end device is equipped with NVIDIA Jetson TX2, and it has a battery with an initial capacity of 4700 mAh. The edge server is equipped with an NVIDIA GeForce RTX2060 GPU, one of NVIDIA's latest offerings for servers.

1) Latency and energy consumption: Here, we first deploy End-only, Edge-only, Neurosurgeon-L, Neurosurgeon-E, and GAFPD on the EI system based on the RMUA platform.

[2]https://www.robomaster.com/zh-CN/robo/icra?djifrom=rmu1

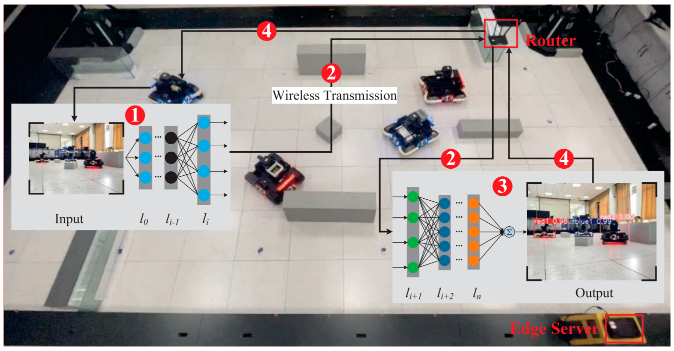


Fig. 8. An EI system based on RMUA. (① The end device captures the current battle environment through its camera, and then utilizes the GAFPD deployed on the end device to obtain the partition points. After that, the NVIDIA Jetson TX2 embedded in the end device performs the corresponding computing tasks; ② The end device transmits the computing result to the edge server via the router; ③ The edge server completes the remaining computing tasks and obtains the final inference result, which is to distinguish between enemy and friendly robots; ④ The edge server transmits the final inference result to the end device via the router.)

TABLE VII
Latency Speedup and Energy Consumption Save of Different Mechanisms Compared to End-Only Based on RMUA

	ResNet18		MobileNet		VGG16	
	Latency	The number of inferences	Latency	The number of inferences	Latency	The number of inferences
End-only	**303 ms**	**95512**	**112 ms**	**98242**	**1104 ms**	**6384**
Edge-only	+190.429%	+90.171%	+475.892%	+183.327%	-76.721%	+4844.612%
Neurosurgeon-L	0	0	0	0	-76.721%	+4844.612%
Neurosurgeon-E	+190.429%	+90.171%	+475.892%	+183.327%	-76.721%	+4844.612%
GAFPD	+60.726%	+27.035%	+129.464%	+88.885%	-76.721%	+4844.612%

TABLE VIII
Latency Speedup and Energy Consumption Save of Different Mechanisms Compared to Neurosurgeon-L Based on RMUA

	ResNet18		MobileNet		VGG16	
	Latency	The number of inferences	Latency	The number of inferences	Latency	The number of inferences
Neurosurgeon-L	**303 ms**	**95512**	**112 ms**	**98242**	**257 ms**	**315664**
End-only	0	0	0	0	+329.572%	-97.978%
Edge-only	+190.429%	+90.171%	+475.892%	+183.327%	0	0
Neurosurgeon-E	+190.429%	+90.171%	+475.892%	+183.327%	0	0
GAFPD	+60.726%	+27.035%	+129.464%	+88.885%	0	0

Then, we fully charge the end device and count the number of inferences that each mechanism can complete when the end device depletes its battery. In addition, we also measure the latency associated with each mechanism for completing a single inference. For convenience, we utilize the number of inferences to assess the energy consumption of each mechanism. Similar to the simulation, we select End-only, Neurosurgeon-L, and Neurosurgeon-E as baselines. The experimental results are presented in Tables VII, VIII, and IX. These tables record the latency of different mechanisms when completing one inference using various DNN models. They also record the number of inferences that can be completed with the energy consumption of the end device with a 4700 mAh capacity. Among them, the column showing the number of inferences indicates the percentage increase (positive) or decrease (negative) of the number of inferences for different mechanisms compared to the baseline.

Similar to the simulation results, the Neurosurgeon-L and End-only have similar effects on reducing latency and energy

TABLE IX
Latency Speedup and Energy Consumption Save of Different Mechanisms Compared to Neurosurgeon-E Based on RMUA

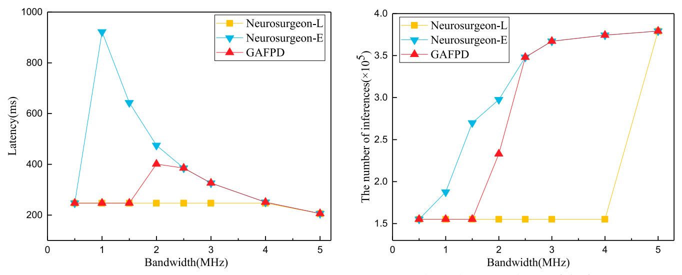
	ResNet18		MobileNet		VGG16	
	Latency	The number of inferences	Latency	The number of inferences	Latency	The number of inferences
Neurosurgeon-E	**880 ms**	**181636**	**645 ms**	**278346**	**257 ms**	**315664**
End-only	-65.568%	-47.416%	-82.636%	-64.705%	+329.572%	-97.978%
Edge-only	0	0	0	0	0	0
Neurosurgeon-L	-65.568%	-47.416%	-82.636%	-64.705%	0	0
GAFPD	-44.659%	-33.199%	-60.155%	-33.333%	0	0

consumption in Table VII. When using lightweight DNN models like ResNet18 and MobileNet, the percentage increase in the number of inferences for GAFPD is smaller compared to Edge-only and Neurosurgeon-E. However, in terms of latency, GAFPD causes a much smaller percentage increase compared to both Edge-only and EOPM. The latency generated by the Edge-only and Neurosurgeon-E is too high for latency-sensitive computing tasks in the EI system based on the RMUA platform. GAFPD is a superior choice to reduce energy consumption, albeit with a certain increase in latency. For VGG16, it is undoubtedly a better choice to hand over computing tasks to the edge server, which can not only reduce latency, but also greatly reduce the energy consumption.

In Table VIII, for ResNet18 and MobileNet, End-only has the same latency and the number of inferences that can be completed as Neurosurgeon-L, so we no longer repeat the analysis of them. For VGG16 with high complexity, compared with the experimental results in Table VII, the latency of Neurosurgeon-L is much smaller than that of End-only, and Neurosurgeon-L can complete 48.45$\times$ more computing tasks compared to End-only. In addition, the GAFPD incurs the same latency and can complete the same amount of inferences as Neurosurgeon-E, which is the optimal result, indicating that our proposed GAFPD is feasible.

In Table IX, it can be seen that the Edge-only has the same latency and can complete the same amount of inferences as Neurosurgeon-E. Since computing tasks are handed over to the edge server, Neurosurgeon-E and Edge-only can complete more inferences. However, limited by the bandwidth, the inference latency of Neurosurgeon-E and Edge-only is higher than that of several other mechanisms. For ResNet18 and MobileNet, although the latency of GAFPD is higher than that of End-only and Neurosurgeon-L, the number of inferences that GAFPD can complete is higher than that of End-only and Neurosurgeon-L. Especially for energy-constrained end devices, it is important to reduce energy consumption within tolerable latency. For VGG16, the latency of GAFPD and the number of inferences that GAFPD can complete are comparable to those of Neurosurgeon-E, which shows that our proposed GAFPD is feasible.

2) Impact of bandwidth: Next, we evaluate the adaptability of GAFPD to bandwidth changes. Here, we use AlexNet to perform image classification over CIFAR-10. Similar to the previous simulation experiment, we select three model partitioning mechanisms: Neurosurgeon-L, Neurosurgeon-E, and GAFPD. We measure the latency and the number of inferences of the EI system based on RMUA under different bandwidths. The results are shown in Fig. 9.



(a) Latency (b) The number of inferences

Fig. 9. Latency and the number of inferences of AlexNet under different bandwidths on the EI system based on the RMUA platform.

In Fig. 9, it can be seen that when the bandwidth is low (0.5 MHz), the latency and the number of inferences that the three model partitioning mechanisms can complete on the EI system based on RMUA are the same. This is because when the bandwidth is low, data transfer takes a significantly longer time and consumes a considerable amount of energy. As a result, all three model partitioning mechanisms opt for the end device to complete the computing task. When the bandwidth increases to 1 MHz, the number of inferences that the EI system can complete gradually increases. At the same time, the latency of Neurosurgeon-E reaches its maximum value. With the continuous increase of bandwidth, the latency of Neurosurgeon-E gradually decreases, and the number of inferences gradually increases. When the bandwidth is greater than 2.5 MHz, Neurosurgeon-E and GAFPD select the same partition point. When the bandwidth reaches 5 MHz, the three model partitioning mechanisms we selected all select the same partition point. This is because as the bandwidth increases, the latency and energy consumption of data transmission are lower, Neurosurgeon-E and GAFPD tend to choose the same partition point. When the bandwidth is high enough, the latency and energy consumption of data transmission are lower than that of end device complete the computing task alone, and the three model partitioning mechanisms tend to choose the same partition point.

With the change of bandwidth, GAFPD neither optimizes the latency separately like Neurosurgeon-L, resulting in an increase in energy consumption, nor does it optimize the energy consumption separately like Neurosurgeon-E, increasing latency and affecting the QoS. In addition, when the bandwidth varies between 1.5 MHz and 4 MHz, GAFPD can trade a small increase in latency off a large increase in the number of inferences (improve energy efficiency). Therefore, it can be seen that GAFPD can adaptively select the overall optimal partition point in response to changes in bandwidth.

VI. Conclusion

Previous work to optimize latency or energy consumption through model partitioning has problems with limited adaptability to dynamically changing computing environments and delayed feedback. In addition, they often neglect the impact of energy consumption on edge devices when optimizing latency, or result in increased latency as a consequence of reducing energy consumption. The computational characteristics of DNN models demonstrate that the latency and energy consumption of edge devices in EI are not only related to the FLOPs allocated to end devices, but are also influenced by the output data size at each partition point of the DNN model. Therefore, we propose GAFPD to efficiently find the optimal partition point that can reduce latency and improve energy efficiency simultaneously. Theoretical analysis and numerous experiments have been proved that GAFPD can optimize latency and energy consumption simultaneously while adapting to changing environments. Furthermore, the experimental results on the EI demonstrate the effectiveness of GAFPD in reducing latency and improving energy efficiency.

References

[1] R. Miikkulainen et al., "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 293–312.

[2] C. Scheuermann, T. Binderberger, N. von Frankenberg, and A. Werner, "Digital twin: A machine learning approach to predict individual stress levels in extreme environments," in *Adjunct Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput., Proc. ACM Int. Symp. Wearable Comput.*, 2020, pp. 657–664.

[3] X. Dai et al., "Dynamic head: Unifying object detection heads with attentions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7373–7382.

[4] H. Lee, Y. Choi, T. Han, and K. Kim, "Probabilistically guaranteeing end-to-end latencies in autonomous vehicle computing systems," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3361–3374, Dec. 2022.

[5] X. Niu, C. Yu, and H. Jin, "CRSM: Computation reloading driven by spatial-temporal mobility in edge-assisted automated industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9283–9291, Dec. 2022.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[7] 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," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[8] A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, "DORY: Automatic end-to-end deployment of real-world DNNs on low-cost IoT MCUs," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1253–1268, Aug. 2021.

[9] G. Zhu et al., "Pushing AI to wireless network edge: An overview on integrated sensing, communication, and computation towards 6G," *Sci. China Inf. Sci.*, vol. 66, no. 3, 2023, Art. no. 130301.

[10] C. Jiang et al., "Energy aware edge computing: A survey," *Comput. Commun.*, vol. 151, pp. 556–580, Feb. 2020.

[11] Y. Kang et al., "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 615–629, 2017.

[12] Y. Duan and J. Wu, "Joint optimization of DNN partition and scheduling for mobile cloud computing," in *Proc. 50th Int. Conf. Parallel Process.*, 2021, pp. 1–10.

[13] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.

[14] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 3017–3030, May 2023.

[15] Z. Zeng, C. Liu, Z. Tang, K. Li, and K. Li, "AccTFM: An effective intra-layer model parallelization strategy for training large-scale transformer-based models," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4326–4338, Dec. 2022.

[16] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.

[17] L. Zhang, L. Chen, and J. Xu, “Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning,” in *Proc. Web Conf.*, 2021, pp. 3111–3123.

[18] F. Dong et al., “Multi-exit DNN inference acceleration based on multi-dimensional optimization for edge intelligence,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5389–5405, Sep. 2023.

[19] H. Dai, J. Wu, Y. Wang, and C. Xu, “Towards scalable and efficient Deep-RL in edge computing: A game-based partition approach,” *J. Parallel Distrib. Comput.*, vol. 168, pp. 108–119, Oct. 2022.

[20] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, “Dependent task offloading for edge computing based on deep reinforcement learning,” *IEEE Trans. Comput.*, vol. 71, no. 10, pp. 2449–2461, Oct. 2022.

[21] P. Lin, Z. Shi, Z. Xiao, C. Chen, and K. Li, “Latency-driven model placement for efficient edge intelligence service,” *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 591–601, Mar./Apr. 2022.

[22] H. Yuan, D. Guo, G. Tang, and L. Luo, “Online energy-aware task dispatching with QoS guarantee in edge computing,” *Chin. J. Internet Things*, vol. 5, no. 2, pp. 71–77, 2021.

[23] L. Wang et al., “Context-aware deep model compression for edge cloud computing,” in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 787–797.

[24] S. Liu, G. Yu, R. Yin, J. Yuan, L. Shen, and C. Liu, “Joint model pruning and device selection for communication-efficient federated edge learning,” *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 231–244, Jan. 2022.

[25] W. Ju, D. Yuan, W. Bao, L. Ge, and B. B. Zhou, “eDeepSave: Saving DNN inference using early exit during handovers in mobile edge environment,” *ACM Trans. Sensor Netw. (TOSN)*, vol. 17, no. 3, pp. 1–28, 2021.

[26] 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.

[27] M. Xue, H. Wu, R. Li, M. Xu, and P. Jiao, “EosDNN: An efficient offloading scheme for DNN inference acceleration in local-edge-cloud collaborative environments,” *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 248–264, Mar. 2022.

[28] P. Ren, X. Qiao, Y. Huang, L. Liu, C. Pu, and S. Dustdar, “Fine-grained elastic partitioning for distributed DNN towards mobile web AR services in the 5G era,” *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3260–3274, Nov./Dec. 2022.

[29] Q. Wang, Y. Xiao, H. Zhu, Z. Sun, Y. Li, and X. Ge, “Towards energy-efficient federated edge intelligence for IoT networks,” in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Piscataway, NJ, USA: IEEE Press, 2021, pp. 55–62.

[30] V. Hayyolalam, M. Aloqaily, Ö. Özkasap, and M. Guizani, “Edge intelligence for empowering IoT-based healthcare systems,” *IEEE Wireless Commun.*, vol. 28, no. 3, pp. 6–14, Jun. 2021.

[31] 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 Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.

[32] I. Chakraborty, D. Roy, I. Garg, A. Ankit, and K. Roy, “Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence,” *Nature Mach. Intell.*, vol. 2, no. 1, pp. 43–55, 2020.

[33] M. Xue, H. Wu, G. Peng, and K. Wolter, “DDPQN: An efficient DNN offloading strategy in local-edge-cloud collaborative environments,” *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 640–655, Mar./Apr. 2022.

[34] M. Ghasemi, D. Rakhmatov, C.-J. Wu, and S. Vrudhula, “EdgeWise: Energy-efficient CNN computation on edge devices under stochastic communication delays,” *ACM Trans. Embedded Comput. Syst. (TECS)*, vol. 21, no. 5, pp. 1–27, 2022, Art. no. 66.

[35] L. Tan, Z. Kuang, J. Gao, and L. Zhao, “Energy-efficient collaborative multi-access edge computing via deep reinforcement learning,” *IEEE Trans. Ind. Informat.*, vol. 19, no. 6, pp. 7689–7699, Jun. 2023.

[36] X. Chen, L. Jiao, W. Li, and X. Fu, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[37] S.-H. Kim, S. Park, M. Chen, and C.-H. Youn, “An optimal pricing scheme for the energy-efficient mobile edge computation offloading with OFDMA,” *IEEE Commun. Lett.*, vol. 22, no. 9, pp. 1922–1925, Sep. 2018.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[39] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” Ontario, Canada: Univ. Toronto, 2009.

[40] J. F. Nash Jr., “The bargaining problem,” *Econometrica*, vol. 18, no. 2, pp. 155–162, 1950.

[41] J. Nash, “Two-person cooperative games,” *Econometrica*, vol. 21, no. 1, pp. 128–140, 1950.

[42] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, “Joint task offloading and resource allocation for energy-constrained mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000–4015, Jul. 2023.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[44] A. G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, *arXiv:1704.04861*.

[45] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*.

Xin Niu received the B.S. degree in Internet of Things engineering and the M.S. degree in computer science and technology from the College of Computer Science, Chongqing University, Chongqing, China, in 2017 and 2020, respectively. He is currently working toward the Ph.D. degree with the Services Computing Technology and System Laboratory, Big Data Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, under the guidance of Prof. Chen Yu. His research interests include edge intelligence, edge computing, and ubiquitous computing.

Yajing Huang received the B.S. degree from the Honors College of Northwestern Polytechnical University, China, in 2021. She is currently working toward the M.S. degree with the Services Computing Technology and System Laboratory, Big Data Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, under the guidance of Prof. Chen Yu. Her research interests include edge intelligence, edge computing, and reinforcement learning.

Zhiwei Wang received the B.S. degree from the College of Computer Science and Technology of Chongqing University, China, in 2021. He is currently working toward the M.S. degree with the Services Computing Technology and System Laboratory, Big Data Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, under the guidance of Prof. Chen Yu. His research interests include edge intelligence, edge computing, and deep learning.

Chen Yu (Member, IEEE) received the B.S. degree in mathematics and the M.S. degree in computer science from Wuhan University, Wuhan, China, in 1998 and 2002, respectively, and the Ph.D. degree in information science from Tohoku University, Sendai, Japan, in 2005. From 2005 to 2006, he was a Japan Science and Technology Agency Postdoctoral Researcher with the Japan Advanced Institute of Science and Technology. In 2006, he was at Japan Society for the Promotion of Science Postdoctoral Fellow with the Japan Advanced Institute of Science and Technology. Since 2008, he has been with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, where he is currently a Professor and a Special Research Fellow, working in the areas of ubiquitous computing, edge computing, and industrial internet.

Hai Jin (Fellow, IEEE) received the Ph.D. degree in computer engineering from HUST, in 1994. He is a Cheung Kung Scholars Chair Professor of computer science and engineering with the Huazhong University of Science and Technology (HUST), China. He was awarded a German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Germany, in 1996, and the Excellent Youth Award from the National Science Foundation of China, in 2001. He is the Chief Scientist of China Grid, the largest grid computing project in China, and the Chief Scientist of the National 973 Basic Research Program Project of Virtualization Technology of Computing System, and Cloud Security. His research interests include computer architecture, virtualization technology, cluster computing and cloud computing, peer-to-peer computing, network storage, and network security. He is a member of the ACM.