Fine-grained Elastic Partitioning for Distributed DNN towards Mobile Web AR Services in the 5G Era

Pei Ren, Xiuquan Qiao, Yakun Huang, Ling Liu, *Fellow, IEEE,* Calton Pu, *Fellow, IEEE,* and Schahram Dustdar, *Fellow, IEEE*

Abstract—Web-based Deep Neural Networks (DNNs) enhance the ability of object recognition and has attracted considerable attention in mobile Web AR and other services. However, neither performing the DNN inference on mobile Web browsers locally nor offloading computations to the cloud can strike a balance between accuracy and efficiency; generally, rude methods are often accompanied by unsatisfactory accuracy. Collaborative approaches seem to fill this gap by coordinating the distributed hierarchical computing resources, especially in the 5G era, but it still faces challenges in the current solutions, such as the lack of (1) full use of 5G resources for the one point DNN computation partitioning schemes; (2) fine-grained branching mechanism; (3) efficient partitioning method; and (4) multi-objective optimization. To this end, we present the fine-grained elastic computation partitioning mechanism for distributed DNN in 5G networks. First, we elaborate two collaborative scenarios. Second, we study the DNN branching mechanism at layer granularity. Next, we propose a DNN computation partitioning algorithm based on deep reinforcement learning. Finally, we develop a mobile Web AR application as a proof of concept. The experiments were conducted in an actually deployed 5G trial network, and the results show the superiority of this collaborative approach. The common theme is, under the premise that Quality of Service (QoS) is satisfied, to balance multiple interests by orchestrating computations across heterogeneous computing platforms.

Index Terms—Mobile service computing, distributed deep neural networks, 5G networks, augmented reality, reinforcement learning

1 Introduction

The emergence of Augmented Reality (AR) [1] services greatly changes the way we interact with the real-world. Web-based AR (Web AR) in particular, which promises a lightweight and cross-platform AR experience that is challenging for current App-based approaches, shows the great potential of AR on mobile devices [2], [3]. Many factors contribute to the phenomenal growth of AR. The boom of Deep Neural Networks (DNNs) in the field of computer vision is one of the most important, as it provides an accurate object recognition solution which is the key for AR subscribers to enter and interact with the mixed-reality world. Web-based DNN has therefore recently become a research hotspot [4].

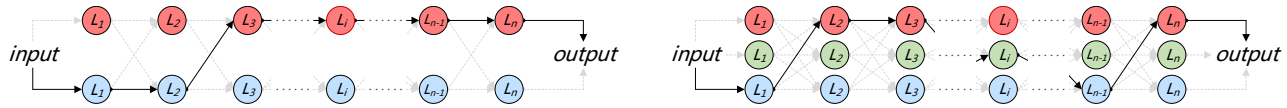
However, current DNN-based object recognition on mobile Web browsers leaves an unsatisfactory choice, either (1) perform DNN inference on the mobile Web browser, especially a built-in browser, with an unacceptable response latency and energy consumption due to its limited computing capability, or (2) offload DNN computations to the cloud, leading to increased deployment costs caused by the occupation of bandwidth and computing resources and also a degradation of the user experience in unstable networks.

To balance the interests of both service subscribers (i.e., maximized user experience) and provider (i.e., minimized deployment overhead), it is natural to consider the use of a collaborative computing approach. But the conventional collaboration between the end-user and cloud still faces significant challenges. Since AR is a computation- and data-intensive service, this kind of cloud-assisted approach will easily result in slowdowns due to the service congestions under high concurrency.

Fortunately, the "network edge" is emerging as another potential collaborator, which promises not only to alleviate concurrency of the central site in a distributed manner but also improve service performance due to the close distance to the subscribers. More generally, edge devices can refer to all equipments that can provide computing services between the data source and the destination [5]. With the deployment of the infrastructure for ubiquitous Mobile Edge Computing (MEC) [6] and Device-to-Device (D2D) [7] communication support in the 5G era, this "edge"-based collaborative approach will be a promising solution for mobile Web AR services.

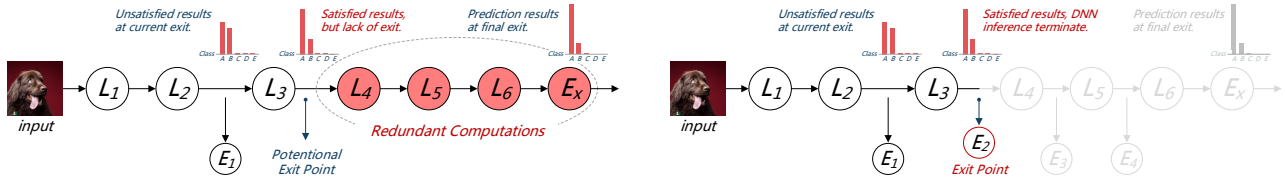
However, employing this collaborative approach over a computing hierarchy is still challenging for the practical application of mobile Web AR in 5G networks:

- The basis of collaborative computing is the computation partitioning. The state-of-the-art approach Neurosurgeon [8] was proposed for collaborative DNN inference but with only one partitioning point (i.e., coarse-grained partitioning), which fails to take full use of 5G resources: for optimal inference latency or energy saving, it will degenerate into a cloud-only solution in the experiment. 5G technologies promise a revolutionary network experience, a more flexible approach is therefore needed. Specifically, by adopting a fine-grained mechanism with multi-partitioning points, computation-intensive DNN layers will be assigned to edge or cloud server for inference acceleration while others will be completed on the mobile Web browsers to reduce the system deployment cost. A comparison of the DNN inference process is illustrated in Figure 1. However, this has not yet been studied throughly.
- The features learned at the early stage of a DNN are sufficient to provide credible recognition for simple samples [9]. But for current coarse-grained DNN branch-



(a) Coarse-grained DNN partitioning (b) Fine-grained DNN partitioning

Fig. 1. Comparison of the DNN inference process. (a) Current coarse-grained partitioning approach is based on only one partitioning point, which divides the DNN computations into mobile Web and remote cloud for execution. Although this partitioning point can be dynamically selected in different scenarios, it still lacks flexibility. (b) 5G network provides the basis for fine-grained elastic partitioning for distributed DNN. By adopting multi-partitioning points, the computation of each DNN layer will be independent of its "neighbors" to be distributed to the mobile Web (red), network edge (green), and remote cloud (blue). For example, assume that performing the L_i DNN computation on the mobile Web browser takes a lot of time and energy, generally, this DNN computation layer/block can be easily assigned to the network edge or remote cloud by our proposed fine-grained partitioning approach. However, this is impossible with a coarse-grained partitioning approach.



(a) Coarse-grained branching mechanism (b) Fine-grained branching mechanism

Fig. 2. Comparison of the DNN branching mechanism. (a) The coarse-grained DNN branching mechanism obviously causes redundant computations. For a pre-defined recognition accuracy threshold, although the intermediate result already meets the accuracy requirements after the process of the first three DNN layers for the given input sample, the subsequent DNN processing still needs to be completed to obtain the recognition results. (b) However, for our proposed fine-grained DNN branching mechanism, once the inference result of the intermediate DNN layer satisfies the recognition requirements, the DNN inference process will be terminated, which will effectively reduce the redundant computations.

ing mechanism [10], the inference processing can only be terminated when the next exit branch is reached, although the intermediate results already satisfy the recognition requirement as shown in Figure 2. Obviously, this will lead to redundant computations, and thus increase the occupation of computing resources and latency of the service response. In contrast, a fine-grained DNN branching mechanism can therefore significantly improve the inference efficiency as the recognition result can be returned in a timely fashion.

- Furthermore, for layer granularity DNN computation partitioning, because each layer is independent of the others, which will then result in an explosive growth in the partitioning decision space. For example, there are only 57 potential partitioning points for ResNet-56 when adopting Neurosurgeon, but will be about 5.23×10^{26} partitioning solutions within the "Device-Edge-Cloud" collaborative scenario for layer granularity DNN partitioning approach with multi-partitioning points. Current partitioning schemes rely on the enumerative approach to get the optimal decision, but obviously, a more efficient approach will be needed.
- Current collaborative approaches focus on the DNN inference latency or mobile energy consumption separately in the computation partitioning process. However, AR service subscribers are sensitive to response latency as well as energy consumption. Performing DNN inference on mobile Web browsers, especially built-in browsers, is more difficult than the App-based approaches, due to their limited computing efficiency of JavaScript. Therefore, both two factors need more attention in mobile Web AR services.

Given these concerns, it is desirable that a collaborative computing paradigm provide a win–win service provisioning solution in the 5G era. To this end, we first focus on the two collaborative computing scenarios (i.e., "vertical"–the collaboration between hierarchy computing resources and "horizontal"–D2D-based resources collaboration) for mobile Web AR services in 5G networks. Then we discuss the architecture design of the enhanced Elastic Computation Offloading (ECO) decision-making mechanism, which takes into account the network performance, computing capability, and customized requirements, simultaneously. Elastic computation partitioning relies on the fine-grained design of the DNN architecture. We next study the branching mechanism in different DNN architectures, and analyze the per-layer latency and energy consumption prediction models as well, which provide the basis for the multi-objective optimization. An efficient DNN computation partitioning approach is another core component for a distributed processing platform. Finally, we propose a DNN computation partitioning solution based on reinforcement learning in order to address the problem of the explosive growth in the partitioning decision space.

Experiments have been conducted within both "vertical" and "horizontal" computing scenarios for mobile Web AR services in 5G networks. The prototype demonstrates an improvement of DNN inference latency by about 72.17% and 47.02%, and 0.875× system throughput improvement on average, also 66.91% mobile energy saving for different DNN architectures (i.e., AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1) with the given partitioning decisions.

The main contribution lies in the following aspects:

- *Prove the availability of collaborative computing mechanism in mobile Web AR services.* Where can the DNN computations be completed? We detail the first collaborative computing scenarios for mobile Web AR in the 5G era, which is the basis for the fine-grained DNN computation partitioning. But, more generally, this can also contribute to other collaborative computing problems

especially in distributed systems over 5G networks.

- *Design of the fine-grained DNN architectures.* (1) To improve the inference efficiency, we re-design the branching mechanism for DNN with the lowest inference cost (i.e., only exits are added to each branch). Unlike the status quo solutions, this fine-grained branching approach will also effectively reduce the redundant DNN computations. (2) On the basis of factor analysis of the device computing capability and the DNN layers' configurable parameters, we then present per-layer inference latency and energy consumption prediction models that give basic insights into the on-demand real-time neural network computation partitioning.
- *DNN partitioning algorithm.* (1) We propose the Intent-oriented Offloading algorithm (IoRLO) based on reinforcement learning, in which Deep Deterministic Policy Gradient (DDPG) is used to learn the optimal partitioning decision, in order to address the challenge of the explosive growth of the decision space. The core mechanism of IoRLO can also provide valuable experience for other decision-making problems. (2) Besides, we formulate a multi-objective optimization problem to meet the user's requirements for inference latency and energy consumption simultaneously in the DNN computation partitioning process. Unlike the status quo solutions, our approach is more flexible based on the fine-grained DNN computation partitioning mechanism.
- *Mobile Web AR application.* We examine the performance of our proposed partitioning solution for Distributed DNN (DDNN) for mobile Web AR applications in the "vertical" and "horizontal" collaborative computing scenarios over the actually deployed 5G networks[1].

The remainder of this paper is organized as follows. Section 2 presents the two collaborative scenarios for mobile Web AR in 5G networks, followed by the overview of the partitioning decision-making system. Section 3 gives the per-layer characteristics prediction models. An analysis of IoRLO is given in Section 4. Section 5 presents the experimental analysis. Section 6 reviews the related literature. Section 7 concludes the paper.

2 Overview of the Architecture

The collaborative mechanism provides opportunities for flexible service provisioning. For clarity, it is important to detail the distributed computing scenario designed for mobile Web AR over 5G networks before the analyzation of the DNN computation partitioning method.

Typically, an AR system pipeline consists of three components: object recognition, object tracking, and annotation rendering [3]. When the AR subscriber targets the camera at the object, the AR system first needs to know what the user is focusing on (i.e., environment perception). That is, object recognition, which provides a key for subscribers to enter the mixed-reality world, and therefore, is one of the most important components in AR systems. The traditional object recognition methods need to first extract the local feature points [11], [12], [13] from the video frames captured by the camera in real time, and then complete the recognition process through object retrieval [14]. Remarkably, the performance of this traditional recognition mechanism is greatly limited by the ability of local feature extraction and object retrieval techniques. With the boom of artificial intelligence, deep learning technologies have been widely used in the field of computer vision (such as object detection, recognition, and segmentation). By adopting the DNN technologies into the augmented reality application, it will be able to greatly improve the accuracy and generalization ability of object recognition (i.e., environment perception), thereby improving the AR application performance.

The rapid development of 5G network technologies and large-scale commercial deployment are major factors that mobile Web AR can be achieved [15]. Specifically, AR is a kind of delay-sensitive and computationally-intensive services. However, mobile Web platforms (especially built-in Web browsers) often require additional computing resources to complete the AR processes due to the limited computing capability [16]. From the communication perspective, 5G networks promise even gigabyte-level bandwidth and millisecond-level network delay, which will greatly optimize the data transmission in the network and thus provide the basis for collaborative computing. From the computing perspective, both the ubiquitous edge servers and surrounding mobile devices can provide AR subscribers with additional computing resources. In this paper, we explore the orchestration of the networking and computing resources in 5G networks for adopting the distributed DNN in mobile Web AR as a proof of concept.

2.1 Collaborative Scenarios in 5G Networks

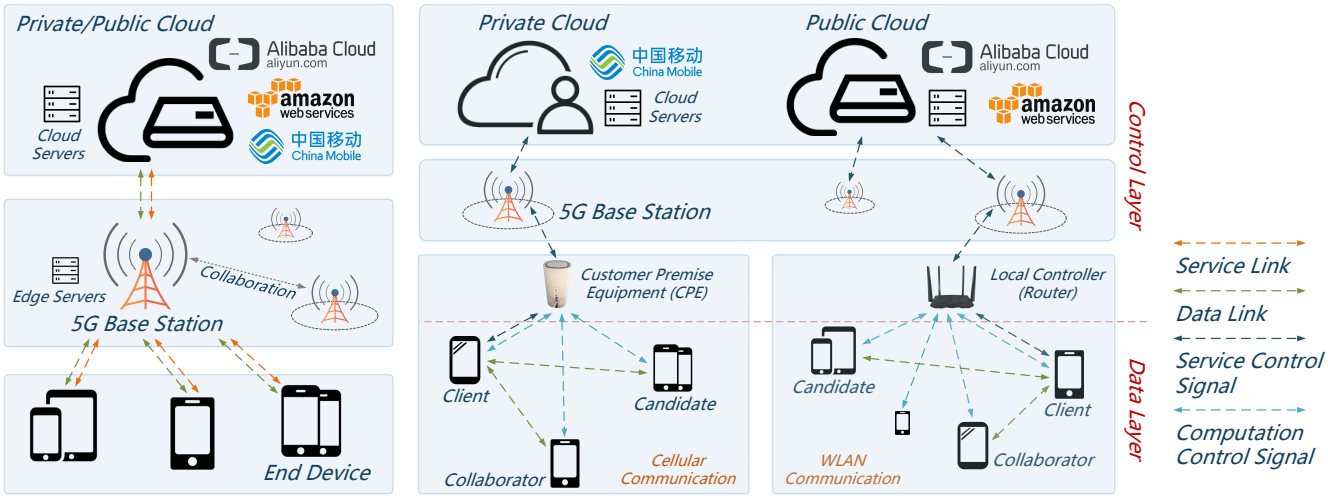
In addition to the skip-type improvement of networking performance, the emerging 5G networks also introduce a variety of promising features, such as the pervasive edge server deployment and D2D communication support. However, the evolution of both computing and communication paradigms bring Web-based services new opportunities as well as challenges. To this end, we propose the MEC-based and D2D-based computing solutions directed at the problem of collaborative service provisioning. More generally, the two collaborative modes can also contribute to other types of services with heavy computation and communication demands, especially in the 5G era.

As shown in Figure 3a, the computing platform in the "vertical" scenario mainly consists of the end device, network edge, and cloud. All the computing and network resources along the path between the end device (i.e., the data source) and the cloud data center can be defined as an "edge" [5]. For simplicity, we refer the micro data center that is deployed on the 5G base station as the network edge.

The D2D-based collaborative computing mode is another important and promising one, attributable to the support of D2D communication technology in 5G networks and the increasing popularity of intelligent devices [17]. The D2D-based mobile Web AR implementations have prospective value for a wide range of applications, such as entertainment (e.g., multiplayer games [18]) and training [19]. Figure 3b shows this collaborative computing scenario.

Based on the discussions above, the question then arises as to how to balance multiple interests by leveraging the dis-

1. The 5G trial network was supported by China Mobile Communications Group Beijing Co., Ltd. and Huawei Technologies Co., Ltd.



(a) MEC-based Collaboration (b) D2D-based Collaboration

Fig. 3. Two collaborative computing scenarios for mobile Web AR in the 5G era.

tributed and heterogeneous resources. Obviously, a flexible computation scheduling approach is thus necessary.

2.2 Why Elastic Computation Offloading is Needed

The conventional offloading paradigms are proposed for mobile cloud computing to solve the problem of Quality of Service (QoS) degradation caused by insufficient computing resources of the user device. However, these efforts, such as MAUI [20] and CloneCloud [21], were all designed to offload as many computations as possible to the cloud, for the sake of improved performance. Service providers therefore had to pay hefty deployment costs to support the operation of the business.

Mobile edge computing as a revolutionary paradigm has been widely recognized. But the offloading decision-making process also becomes more difficult for the following reasons: (1) Complex computing scenario. Computations can be completed not only in the cloud or locally, but also can be assigned to various distributed and heterogeneous computing platforms. (2) Dynamic network situation. The communication situation will also be more complicated and changeable due to the increase in networking equipments. (3) Diverse service requirements. Both the user experience and service deployment costs need to be considered.

Fortunately, the computing capability of mobile devices is now increasing rapidly: more and more complex computations can be completed locally. With the breakthrough of wireless transmission in 5G networks, the connection between network edge devices (including these user end devices) will be more closer. An elastic computation offloading mechanism relies on efficient network communication capability, which can realize dynamic decision-making in changing application environment, and is expected to balance the interests of both the user and service provider. Specifically, under the premise that QoS (it depends on the service requirements, and here is just a generalized concept) is satisfied, by coordinating computations across heterogeneous platforms, the crowdsourced computing capability will therefore effectively alleviate the computational pressure on the central site in the distributed manner.

3 Fine-grained Deep Neural Networks

Features learned at the early stage of a DNN are sufficient to provide credible result for simple samples. By placing early exits at the appropriate position, this DNN architecture can obviously reduce the inference time and simultaneously reduce the computational cost [10]. This section focuses on the improvement of the branch-based DNN architecture.

3.1 Design of the Branch-based DNN Architecture

Aiming at the fine-grained computation partitioning and scheduling, we first explore the DNN branching mechanism as follows. Figure 2 (see Supplementary Material) illustrates the architecture improvement of DNN; here we use the AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1 as examples. Early exits have the potential to provide credible predictions and the number of input samples exiting from each branch is illustrated in Table 1 (see Supplementary Material). Similarly, we use entropy (i.e., $\text{entropy}(y) = \sum_{c \in C} y_c \log y_c$) to measure the confidence of each side branch [10]. When the classification result is sufficient (i.e., $\text{entropy}(y) <$ Thrshld) for prediction, the output will then be returned immediately as the final result without further inference processing. Here y refers to the probabilities of all possible class labels C, and Thrshld indicates the predefined threshold of side branches.

This lightweight branching and processing mechanism for DDNN can effectively improve the inference efficiency and can also optimize the system load, but it is still affected by the specific branch design mechanisms. Because this is beyond the scope of this paper, here we only give a brief and necessary discussion about it; and only use the dense DNN branch structure with the FC layer for demonstration.

- *The density of DNN branches (i.e., how the early exit points are determined)*: The introduction of DNN early exit branches will undoubtedly cause additional inference processes in the case where the prediction results are not credible enough, especially when the fine-grained DNN branch design is intensive, the complex input

samples will continuously activate multiple branch inference process, which is ineffective for prediction and will lead to negative effects. While in practical applications, the determination of early exit points should depend on multiple factors. For example, when all the input samples are more complex, by avoiding the early stage DNN branches can reduce the unnecessary computations; while when service providers have lower requirement for prediction accuracy, those early stage DNN branches will help improve the system inference efficiency. Moreover, in a distributed collaborative computing scenario, the DNN service computational complexity can also be reduced by adjusting the inference process (i.e., execution mechanism). Assume that the DNN computations are partitioned into several parts as shown in Figure 4, here only certain side branches will be activated. Specifically, before the intermediate results are transmitted between two computing platforms, the subsequent exit branch then will be activated, if the features learned are sufficient for prediction, the DNN inference process will be terminated and return the results to the client immediately.

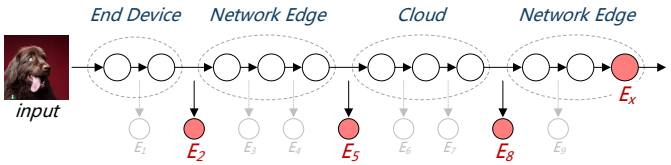


Fig. 4. Execution mechanism of branch-based DNN. When the intermediate result is transmitted between different platforms, will the branch exits be activated first to determine whether it is necessary to perform the subsequent DNN computations.

- *The complexity of each branch*: On the other side, the complexity of the introduced branch will also affect the inference efficiency; Complex DNN branches often lead to increased inference response latency, energy consumption, and storage occupation. For example, the branches in BranchyNet adopts the structure of $n\times$ Convolutional layer + Fully-Connected layer, undoubtedly the introduction of convolution operation will increase the complexity of the DNN branch; while our proposed branching mechanism requires only one Fully-Connected layer for each branch, which is necessary to be used for early exit. By comparison, this cost is still acceptable when compared with the other complex branching mechanisms. In practice, the complexity of the branch structure can be customized by the developer based on specific application requirements.

Theoretically, this collaborative method can achieve a balance of DNN inference accuracy and efficiency, that is, by dynamically adjusting the prediction accuracy threshold, so as to meet the different inference efficiency requirements. Partitioning schemes discussed in this paper are for the DNN computations scheduling, the prediction accuracy of the system is not directly related to it, but depends on the accuracy threshold predefined by the service provider; this partitioning mechanism can be directly applied to the current mature DNN backbone networks without additional modification (if early exit is not considered). On the other hand, compared with conventional methods, different branch-based DNN training modes will also have different effects on the DNN prediction accuracy. Specifically, if the branch parameters are trained directly based on the pre-trained DNN model for early exit, the parameters of the DNN backbone network remain unchanged, and the prediction accuracy of the system depends on the designate of the threshold (i.e., the method we used in this paper); while if all the parameters of the branch neural network are retrained, these early exit points may play the same role as the softmax in GoogLeNet [22], which can achieve more efficient gradient transmission, thereby affecting the prediction accuracy of the entire DNN model, but this is not within the scope of our discussion.

3.2 Per-Layer Prediction Models

For the elastic computation partitioning, we give two prediction models on DNN inference latency and energy consumption by analyzing the per-layer input and/or output size (although different DNN layers have different floating-point operations, their inference efficiency is not proportional to this value [23]) and capabilities of the computing platforms. The performance measurement is conducted with different neural network architectures at layer level using various computing platforms (due to the limited data access on iOS, we only used mobile devices with Android OS for testing). To avoid the impact of architectural differences between computing platforms on the DNN performance of feedforward, we introduce a "standard" approach using an edge server (IBM System x3650 M4, Intel Xeon E5-2600 v2 @ 2.0 GHz) as a measurement benchmark to analyze the device computing capability [24]. Moreover, only the mobile device energy consumption is measured by collecting the idle and active status using Power Monitor AAA10F.

Specifically, our obtained DNN layer characteristics models are detailed below: (1) for convolution, pooling, and fully-connected layers, both the inference latency and energy consumption prediction model can be expressed as $prediction = d \times (\alpha \times input + \beta \times output + \gamma)$; (2) the prediction model of ReLU and normalization layers only considers the computing capability and input feature size, that is, $prediction = d \times (\alpha \times input + \gamma)$, since these layers have fewer configurable parameters.

The parameters for the DNN layer characteristics prediction models are listed in Table 1. Besides the linear regression-based prediction models analysis, we also consider logistic-based and polynomial-based approaches. But the linear-based approach perform better than others, with lower prediction errors for both DNN inference latency and energy consumption, that is, on average 13.25% and 49.35% lower compared to logistic-based approach, and 83.13% and 85.28% lower compared to polynomial-based approach.

Although the system status monitoring on the mobile device may increase the energy consumption, it is negligible since the energy required is small and the monitoring operation is only conducted periodically. Moreover, the fitting processing of the proposed prediction models is completed on the edge/cloud server automatically, which can be updated effectively by collecting the feedforward performance on the different computing platforms in real time.

Based on the analysis and discussion above, the proposed per-layer latency and energy consumption prediction

TABLE 1
Parameters for the DNN Characteristics Prediction Models

Layer Type	Inference Latency Models			Energy Consumption Models		
	α_l	β_l	γ_l	α_e	β_e	γ_e
Conv	6.240e−5	1.074e−4	−1.938e+0	9.240e−7	1.874e−6	3.810e−2
ReLU	1.534e−5	–	4.844e−1	1.435e−6	–	2.881e−1
Pooling	1.136e−5	1.313e−6	−1.695e+0	1.410e−6	1.312e−7	3.572e−1
Norm	5.182e−5	–	6.497e−1	5.187e−6	–	5.991e−1
FC	9.163e−5	3.990e−4	1.172e+0	9.213e−6	4.012e−5	1.125e+0

* We use l and e as subscripts to identify α, β, and γ in DNN inference latency and energy consumption prediction models, and α, β, and/or γ are the parameters used by the above prediction models as detailed in subsection 3.2.

models provide basic insights into how to achieve fine-grained DNN computation partitioning. These models also make their predictions based on the configurations without execution in advance, and therefore can be directly applied to different DNN architectures as a generalized approach.

4 The Elastic Partitioning Method

Based on the discussion above, the DNN computation partitioning method should be responsible for: (1) adaptive decision-making in accordance with the different capabilities of the computing platforms and mobile network performance; and (2) striking a balance between user experience and system deployment cost.

4.1 Problem Formulation

In this part, we consider the two typical computing scenarios in the 5G era as mentioned before (see subsection 2.1).

Our object is a win–win situation, that is, by leveraging the newly emerged computing and communication technologies in 5G networks, the demands from the two conflicting parties (i.e., AR service subscribers and provider), viz. user experience maintenance and deployment cost saving, are expected to be satisfied simultaneously. The analyses of the system for these two scenarios are given below.

4.1.1 "Vertical" Computing Scenario

As shown in Figure 5, the overall system processing model consists of two parts, the performance monitoring module and the service provisioning module.

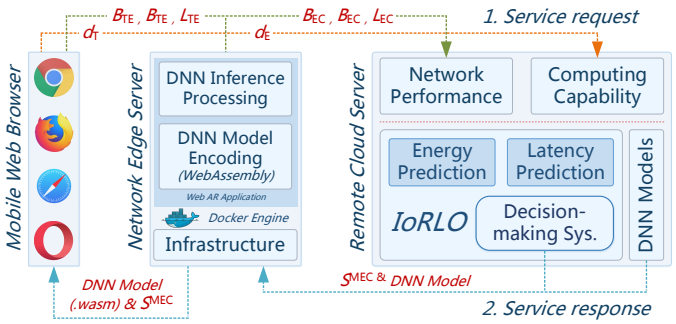


Fig. 5. MEC-based collaboration pipeline for mobile Web AR services.

Performance Monitoring Module. An additional thread is maintained for periodically monitoring the performance of the network and reporting the computing capability. Specifically, we denote by $B_{\mathrm{TE}}^{\mathrm{U/D}}$ the uplink and downlink bandwidths (mbps) between the Terminal and Edge server, and by $B_{\mathrm{EC}}^{\mathrm{U/D}}$ the bandwidths between the Edge and Cloud servers. Similarly, we denote by $L = \{L_{\mathrm{TE}}, L_{\mathrm{EC}}\}$ the end-to-end latency (ms) between these three computing platforms. Only if the detected data exceeds the pre-defined thresholds will it be updated to the cloud. The computing capabilities (GHz) of the mobile terminal device d_{T} and network edge server d_{E} are also collected during the service request phase.

Service-oriented Processing Pipeline. After receiving a service request, the cloud performs the computation partitioning using the proposed IoRLO algorithm based on the collected network performance, capabilities of the computing platforms (i.e., $d = \{d_{\mathrm{T}}, d_{\mathrm{E}}, d_{\mathrm{C}}\}$, here we denote by d_{C} the computing capability of the cloud server), latency and energy consumption prediction models, DNN architecture, and the proposed IoRLO algorithm. Then the target DNN model and the computation partitioning decision will be returned to the user and edge server. And we denote by $G = \{G_1, G_2, \ldots, G_n\}$ the DNN blocks. For example, the number of blocks in the branch-based AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1 architectures is 10, 15, 17, and 6 as shown in Figure 1 (see Supplementary Material). The granularity of neural network segmentation is determined by the service provider during the DNN model design phase. Moreover, we denote by G_i^{in} and G_i^{out} the size (Kilobyte) of input and output of the i-th block, respectively, and denote by $S = \{S_1, S_2, \ldots, S_n\}$ the obtained partitioning decision in the MEC-based computing scenario, where $S_i \in \{1, 2, 3\}$. Explicitly, $S_i = 1$ indicates that the i-th DNN block is assigned to execute on the mobile Web browser, $S_i = 2$ means it will be completed on the edge server, otherwise, it will be processed on the cloud, that is, $S_i = 3$. Worth mentioning is that to perform inference on the mobile Web platform, the DNN models will be encoded in the WebAssembly[2] format on the edge server in advance.

Based on the proposed per-layer characteristics prediction models, we can therefore derive the inference latency of the i-th DNN block on computing platform X as $T_{i-\mathrm{cp}}^{X}$, $X \in \{\mathrm{T}, \mathrm{E}, \mathrm{C}\}$, $i \in \{1, 2, \ldots, \mathrm{n}\}$. Similarly, the per-block inference energy consumption is defined as $E_{i-\mathrm{cp}}^{\mathrm{T}}$, here we only focus on the overhead of mobile device.

Considering the layer-by-layer DNN inference mechanism, the i-th neural network block can only be executed after the computing platform receives its input (i.e., output of the previous block or the raw image for the first DNN block) completely; and only after the current block is completed will its output be sent to the next block. The detailed modeling of the cost of inference latency and energy consumption in the DNN computation partitioning is then illustrated in what follows.

Cost of Local Computing. When the i-th neural network block is assigned to execute on the mobile device, the cost of latency can be generally divided into two parts, the part for receiving the input T_{i-cmr}^{T} and the part for inference

2. WebAssembly [25] is designed as a computational acceleration approach by encoding procedures (e.g., C, C++) into a size- and load-time-efficient binary format, which can be executed on the Web directly.

processing T^{T}_{i-cp}. For a given partitioning decision, the total cost of latency for inference on the mobile device is

$$T^{\mathrm{T}}_{\mathrm{total}} = \sum_{i=1,S_i=1}^{\mathrm{n}} (T^{\mathrm{T}}_{i-cp} + T^{\mathrm{T}}_{i-cmr}). \quad (1)$$

Specifically, only the previous result needs to be obtained from other computing platforms, then it will cause the transmission overhead. Moreover, we assume that the raw image has already been obtained, and thus the transmission latency will also be neglected in the case $S_1 = 1$.

In addition to the energy consumed by carrying out DNN inference on the mobile Web browser, the data transmission will also consume mobile energy E^{T}_{i-cm}, which includes not only receiving input but also sending the output to other platforms. Specifically, if the input comes from edge server, the T^{T}_{i-cmr} will be $G^{\mathrm{in}}_i/B^{\mathrm{D}}_{\mathrm{TE}} + L_{\mathrm{TE}}$, and if it is from remote cloud, the data receiving latency needs to increase $G^{\mathrm{in}}_i/B^{\mathrm{D}}_{\mathrm{EC}} + L_{\mathrm{EC}}$; similarly, the data sending latency to the edge server and remote cloud is $G^{\mathrm{out}}_i/B^{\mathrm{U}}_{\mathrm{TE}} + L_{\mathrm{TE}}$ and $G^{\mathrm{out}}_i/B^{\mathrm{U}}_{\mathrm{TE}} + G^{\mathrm{out}}_i/B^{\mathrm{U}}_{\mathrm{EC}} + L_{\mathrm{TE}} + L_{\mathrm{EC}}$, respectively.

Here we only consider the energy consumption of mobile terminal device $E^{\mathrm{T}}_{\mathrm{total}}$, with

$$E^{\mathrm{T}}_{\mathrm{total}} = \sum_{i=1,S_i=1}^{\mathrm{n}} (E^{\mathrm{T}}_{i-cp} + E^{\mathrm{T}}_{i-cm}). \quad (2)$$

Moreover, we denote by $r_{5\mathrm{G}}$ the available data rate, and by $P^{\mathrm{U/D}}_{5\mathrm{G}} = \alpha^{\mathrm{U/D}}_{5\mathrm{G}} \cdot r^{\mathrm{U/D}}_{5\mathrm{G}} + \beta_{5\mathrm{G}}$ (the parameters $\alpha^{\mathrm{U}}_{5\mathrm{G}}$ and $\alpha^{\mathrm{D}}_{5\mathrm{G}}$ are set to be 65 mW/Mbps and 6.5 mW/Mbps, respectively; and $\beta_{5\mathrm{G}}$ is set to be 11475.97 mW) the uplink and downlink transmit power in 5G networks [26]. Therefore the mobile energy consumed during the data transmission is

$$E^{\mathrm{T}}_{i-cm} = T^{\mathrm{T}}_{i-cmr} \cdot P^{\mathrm{D}}_{5\mathrm{G}} + T^{\mathrm{T}}_{i-cmt} \cdot P^{\mathrm{U}}_{5\mathrm{G}}$$

Cost of On-Edge Computing. Similarly, when the i-th DNN block is assigned to execute on the edge server, the cost of processing latency is given by

$$T^{\mathrm{E}}_{\mathrm{total}} = \sum_{i=1,S_i=2}^{\mathrm{n}} (T^{\mathrm{E}}_{i-cp} + T^{\mathrm{E}}_{i-cmr}). \quad (3)$$

Also, the input (i.e., the output of the previous block) source of the i-th DNN block can be classified into three categories, that is, mobile device, network edge, and cloud server. When the data need to be transfered from mobile device, then the transmission latency is $G^{\mathrm{in}}_i/B^{\mathrm{U}}_{\mathrm{TE}} + L_{\mathrm{TE}}$; and the data receiving latency from the remote cloud is thus $G^{\mathrm{in}}_i/B^{\mathrm{D}}_{\mathrm{EC}} + L_{\mathrm{EC}}$.

When the first DNN block is specified to be executed by the edge server, the raw image needs to be transmitted from the mobile device as the input to this block. Therefore, the mobile energy consumption caused by image transmission is given by

$$E^{\mathrm{E}} = E^{\mathrm{E}}_{1-\mathrm{cmr}} = (\frac{G^{\mathrm{in}}_1}{B^{\mathrm{U}}_{\mathrm{TE}}} + L_{\mathrm{TE}}) \cdot P^{\mathrm{U}}_{5\mathrm{G}}. \quad (4)$$

Cost of On-Cloud Computing. In the case where the computing task is assigned to the cloud for execution, we define the cost of inference latency as follows:

$$T^{\mathrm{C}}_{\mathrm{total}} = \sum_{i=1,S_i=3}^{\mathrm{n}} (T^{\mathrm{C}}_{i-cp} + T^{\mathrm{C}}_{i-cmr}). \quad (5)$$

Similarly, the per-block input transmission latency T^{C}_{i-cmr} from the mobile device and edge server is $G^{\mathrm{in}}_i/B^{\mathrm{U}}_{\mathrm{TE}} + G^{\mathrm{in}}_i/B^{\mathrm{U}}_{\mathrm{EC}} + L_{\mathrm{TE}} + L_{\mathrm{EC}}$ and $G^{\mathrm{in}}_i/B^{\mathrm{U}}_{\mathrm{EC}} + L_{\mathrm{EC}}$, respectively.

If $S_1 = 3$, the raw image needs to be transmitted from the mobile device to the remote cloud, therefore the energy consumption of the mobile device caused by the data transmission is $E^{\mathrm{C}} = E^{\mathrm{C}}_{1-\mathrm{cmr}}$, with

$$E^{\mathrm{C}}_{1-\mathrm{cmr}} = E^{\mathrm{E}}_{1-\mathrm{cmr}} + (\frac{G^{\mathrm{in}}_1}{B^{\mathrm{U}}_{\mathrm{EC}}} + \frac{L_{\mathrm{EC}}}{2}) \cdot P^{\mathrm{U}}_{5\mathrm{G}}. \quad (6)$$

Cost of Result Receiving. In addition, the inference result needs to be finally returned to the user; and the result transmission latency T_{result} depends on where the n-th DNN block is performed. In detail, the latency of result transmission from edge server and remote cloud is $G^{\mathrm{out}}_{\mathrm{n}}/B^{\mathrm{D}}_{\mathrm{TE}} + L_{\mathrm{TE}}$ and $G^{\mathrm{out}}_{\mathrm{n}}/B^{\mathrm{D}}_{\mathrm{TE}} + G^{\mathrm{out}}_{\mathrm{n}}/B^{\mathrm{D}}_{\mathrm{EC}} + L_{\mathrm{TE}} + L_{\mathrm{EC}}$, respectively. Similarly, the energy consumption of mobile device for service result receiving is defined as $E_{\mathrm{result}} = T_{\mathrm{result}} \cdot P^{\mathrm{D}}_{5\mathrm{G}}$.

In summary, for a given DNN computation partitioning decision, the total cost of the inference latency T and mobile energy consumption E is defined as the sum of the above four types of costs. It is worth noting that the object of our partitioning mechanism is a win–win situation, besides the aforementioned user experience-related requirements, we also considered the deployment overhead, especially, computing cost, from the service provider's perspective. For simplicity, we take the percentage of computations completed at the network edge and the remote cloud platforms as the mobile Web AR service deployment overhead, that is, $D_{\mathrm{cp}} = \sum_{i=1,S_i\neq 1}^{\mathrm{n}} S_i/\mathrm{n}$. More detailed deployment overhead optimization problems will be discussed in our future works, for example, taking into account the computing capability and rental differences between the network edge and the remote cloud servers.

Based on the above discussion, we formulate the optimization problem that aims at minimizing the DNN inference latency, mobile energy consumption, and service deployment overhead by optimizing the computation partitioning decision. Specifically, our formulated DNN computation partitioning problem is given by:

$$\mathcal{P}: \ min \ (\underbrace{T + \eta^{\mathrm{MEC}}_1 \cdot E}_{\text{Mobile Web User}} + \underbrace{\eta^{\mathrm{MEC}}_2 \cdot D_{cp}}_{\text{Service Provider}}).$$

Note that the above weighted objective function is able to balance the DNN inference latency and mobile energy consumption (for service subscriber) and deployment overhead (for service provider) dynamically by adjusting the predefined weighting factor η^{MEC}_1 and η^{MEC}_2.

4.1.2 "Horizontal" Computing Scenario

The system processing model for D2D-based computing scenario is illustrated in Figure 6, which significantly differs from the aforementioned "vertical" computing scenario but also includes two parts.

Device Monitoring Module. This part is responsible for the monitoring of (1) the availability of all candidate collaborators $H = \{H_1, H_2, \ldots, H_m\}$; (2) the computing capability of both end-user device d_{U} and other collaborators $d_{\mathrm{H}} = \{d_{\mathrm{H}_1}, d_{\mathrm{H}_2}, \ldots, d_{\mathrm{H}_{\mathrm{m}}}\}$; and (3) the communication

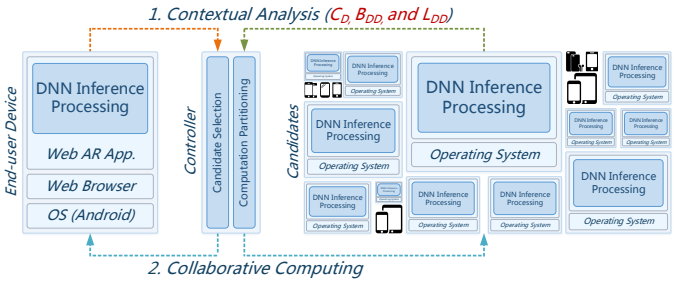


Fig. 6. D2D-based collaboration pipeline for mobile Web AR services.

characteristics, that is, the end-to-end data transmission bandwidth $B_{\mathrm{H}} = \{B_{\mathrm{H}_1}, B_{\mathrm{H}_2}, \ldots, B_{\mathrm{H}_\mathrm{m}}\}$ and the network latency $L_{\mathrm{H}} = \{L_{\mathrm{H}_1}, L_{\mathrm{H}_2}, \ldots, L_{\mathrm{H}_\mathrm{m}}\}$ between the end-user and other collaborators. Here we denote by m the number of available collaborators in a D2D network. Considering the fluctuations in the network performance and load on the computing platform, the local controller which is deployed over the access point will periodically select the "best" collaborator, that is, $H_x = \arg\max(d_{\mathrm{H}_i}/\max(d_{\mathrm{H}}) + B_{\mathrm{H}_i}/\max(B_{\mathrm{H}}) + 1/L_{\mathrm{H}_i})$, based on the previously monitored context information.

Generally, the block-based DNN computations are partitioned among two platforms (i.e., the user device and a specific collaborator with computing capability d_{H_x}). But other "candidates" may be involved in the following cases:

- Overload of the collaborator degrades the service performance and thus needs to be replaced by others.
- The collaborator leaves the current D2D communication network proactively.
- The collaborator disconnects from the controller or user device due to an unstable connection.

Service-oriented Processing Pipeline. The local controller will perform the DNN computation partitioning based on the collected network performance and the computing capabilities of the devices after deciding on the collaborator. The specified neural network model and the partitioning decision S, $S_i \in \{0, 1\}$, will be transmitted to the end-user device and the collaborator, simultaneously. Specifically, $S_i = 0$ means that the i-th DNN block is assigned to execute on the end-user device, otherwise, it will be completed on the selected collaborator.

Based on the discussion above, we can therefore derive the DNN inference latency of the i-th neural network block on end-user device and collaborator as $T^{\mathrm{U}}_{i-\mathrm{cp}}$ and $T^{\mathrm{H}}_{i-\mathrm{cp}}$, $i \in \{1, 2, \ldots, \mathrm{n}\}$, respectively. Also we only focus on the user's energy consumption, which is denoted by $E^{\mathrm{U}}_{i-\mathrm{cp}}$. The service overhead is described in detail below.

Cost of Local Computing. Similarly, the processing latency consists of input receiving T^{U}_{i-cmr} and inference processing. For a given partitioning decision , the total cost of the latency on the end-user device is given by

$$T^{\mathrm{U}}_{\mathrm{total}} = \sum_{i=1, S_i=0}^{n} (T^{\mathrm{U}}_{i-cp} + T^{\mathrm{U}}_{i-cmr}). \tag{7}$$

Specifically, input transmission only occurs if the previous DNN block is executed on the collaborator platform. The transmission latency, especially for data receiving, is therefore given by $T^{\mathrm{U}}_{i-cmr} = G^{\mathrm{in}}_i / B_{\mathrm{H}_x} + L_{\mathrm{H}_x}$. For the first DNN block or the previous block that is completed locally, there will be no data transfer. Also, when the next DNN block is assigned to the collaborator, that is, $S_{i+1} = 1$, then the output sending will be activated, and thus, the data transmission latency is defined as $T^{\mathrm{U}}_{i-cmt} = G^{\mathrm{out}}_i / B_{\mathrm{H}_x} + L_{\mathrm{H}_x}$.

Similarly, the mobile energy consumption of the end-user device consists of three parts, that is, DNN inference, input receiving E^{U}_{i-cmr} and output sending E^{U}_{i-cmt}.

$$E^{\mathrm{U}}_{\mathrm{total}} = \sum_{i=1, S_i=0}^{\mathrm{n}} (E^{\mathrm{U}}_{i-cp} + E^{\mathrm{U}}_{i-cmr} + E^{\mathrm{U}}_{i-cmt}) \tag{8}$$

Specifically, the mobile energy overhead of communication depends on the transmit power and the transmission duration, with $E^{\mathrm{U}}_{i-cmr/t} = T^{\mathrm{U}}_{i-cmr/t} \cdot P_{\mathrm{D2D}}$. Here, we denote by r_{D2D} the available data rate between the two devices and by $P_{\mathrm{D2D}} = \alpha_{\mathrm{D2D}} \cdot r_{\mathrm{D2D}} + \beta_{\mathrm{D2D}}$ the transmit power in a D2D network using Wi-Fi Direct link. The parameters α_{D2D} and β_{D2D} are set to be 283.17 mW/Mbps and 132.86 mW, respectively [26].

Cost of Collaborator Computing. When the i-th DNN block is assigned to execute on the selected collaborator, the processing latency is given by

$$T^{\mathrm{H}}_{\mathrm{total}} = \sum_{i=1, S_i=1}^{n} (T^{\mathrm{H}}_{i-cp} + T^{\mathrm{H}}_{i-cmr}). \tag{9}$$

The input source of the current DNN block may be end-user device (including the raw image and the output of previous block) or collaborator itself. Only for the first case, the input data needs to be transmitted to the current collaborator before continuing the DNN inference. The data transmission latency for the input receiving is therefore defined as $T^{\mathrm{H}}_{i-cmr} = G^{\mathrm{in}}_i / B_{\mathrm{H}_x} + L_{\mathrm{H}_x}$. Note that when the first DNN block is assigned to be executed by the collaborator, the raw image needs to be transmitted to the collaborator, which will also cause the mobile energy consumption:

$$E^{\mathrm{H}} = E^{\mathrm{H}}_{1-\mathrm{cmr}} = (\frac{G^{\mathrm{in}}_1}{B_{\mathrm{H}_x}} + L_{\mathrm{H}_x}) \cdot P_{\mathrm{D2D}}. \tag{10}$$

Additionally, the transmission of the inference result from collaborator to the end-user device will also incur a latency cost $T_{\mathrm{result}} = G^{\mathrm{out}}_{\mathrm{n}} / B_{\mathrm{H}_x} + L_{\mathrm{H}_x}$ and mobile energy consumption $E_{\mathrm{result}} = T_{\mathrm{result}} \cdot P_{\mathrm{D2D}}$. In summary, the total cost of the inference latency and energy consumption in this collaborative computing scenario is the sum of the costs incurred by the mobile user and the collaborator.

Based on the above discussion, our formulated computation partitioning problem can be defined as follows:

$$\mathcal{P}: \ min\ (\ T + \eta^{\mathrm{D2D}} \cdot E\ ).$$

4.2 Intent-oriented Offloading Algorithm (IoRLO)

Considering the above discussion, what we focused on are two different collaborative computing scenarios in 5G networks. But in essence, what they discussed is to reasonably allocate DNN computations to different computing platforms with the help of a partition system to meet multiple requirements. The object of this partitioning system is to reduce the service deployment overhead while simultaneously satisfying the user experience requirement.

In general, assuming the number of computing platforms is a and the number of DNN blocks is n', the number of potential partitioning decisions therefore will be $a^{n'}$. And as the DNN architecture continues to deepen, the decision-making process will become more complex. To this end, an offloading decision-making approach based on deep reinforcement learning is proposed, which leverages the DDPG for addressing the challenges of this "dimension disaster".

In this part, we analyze the proposed IoRLO algorithm, which is designed as a computation partitioning solution for distributed DNN in the 5G era. This decision maker includes the following three key features:

- *Efficiency*. The experience replay mechanism makes it easy to use the existing "experience" to accelerate the training process. Also, using the online learning method enables the decision maker to cope with changing environments, and improves the decision-making ability attributed to the "rich" learning experiences.
- *Flexibility*. Various factors, such as custom requirements from service provider, network performance, and computing capability, are all considered for the decision making, so the IoRLO contributes to the elasticity of the DNN partitioning in practical application.
- *Reusability*. Although the IoRLO is designed for mobile Web AR service, the core mechanism also motivates the investigations of partitioning problems in other fields.

The core phases of IoRLO are detailed below.

4.2.1 Overview of the IoRLO Algorithm

The goal of IoRLO is to devise a computation partitioning policy π that can generate an optimal partitioning decision (i.e., action, which indicates the computing platform each DNN block will be assigned to) $a_t = \{a_{t,1}, a_{t,2}, \ldots, a_{t,n}\}$, $a_t \in A$, based on the received agent state information $s_t = \{s_{t,1}, s_{t,2}, \ldots, s_{t,n}\}$, $s_t \in S$ at the t-th time frame, which represents the DNN inference cost, including the DNN block inference latency, energy consumption, and service deployment overhead (in the MEC-based "vertical" collaborative computing scenario), here n is the number of blocks in the DNN model.

As illustrated in Figure 7, the IoRLO algorithm mainly consists of two components, Actor and Critic. The partitioning action is generated by the Actor after receiving the state information s_t. When the action acts on the environment (here the environment refers to the 5G communication networks), the reward r_t will be fed back and then the agent enters the state s_{t+1}. Based on the previous discussion, the reward r_t can be represented as $-(T^{\mathrm{MEC}} + \eta_1^{\mathrm{MEC}} \cdot E^{\mathrm{MEC}} + \eta_2^{\mathrm{MEC}} \cdot D_{cp})$ in the "vertical" computing scenario; and $-(T^{\mathrm{D2D}} + \eta^{\mathrm{D2D}} \cdot E^{\mathrm{D2D}})$ in the "horizontal" scenario. Because the two scenarios discussed in Section 4.1 are all designed to reduce the DNN inference cost, therefore in the IoRLO algorithm, the reward needs to be set to the opposite of the cost value as the agent always tends to the actions with higher reward in reinforcement learning. The tuple (s_t, a_t, r_t, s_{t+1}) will be stored in memory for the learning of Actor and Critic, that is, θ^{μ} and θ^{Q}, respectively. Then it comes to the partitioning policy $\pi_{\theta_{t+1}^{\mu}}$.

It should be noted that r_t (i.e., $r_{ss'}^{a}$) refers to the immediate environment reward for performing the action a to state s' under the state s, which is defined as $r_{ss'}^{a} = -cost_t$. According to the Markov Decision Process (MDP), the cumulative reward at state s_t is given by $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma \in [0, 1]$ is the discount factor. Obviously, our goal is to maximize the expected cumulative reward through the learned DNN computation partitioning policy $\pi(a|s) = \arg\max E(R_t)$.

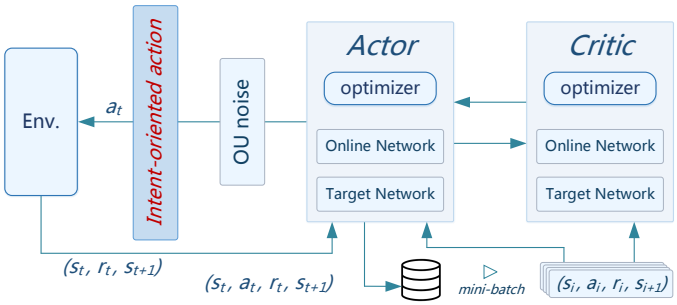


Fig. 7. Processing pipeline for DDPG-based IoRLO.

4.2.2 Intent-oriented Action Generation

The action space noise is introduced to the output of the Actor network which adopts the Sigmoid function, that is, $a = \mathrm{Sigmoid}(z_a)$, as the activation function to balance exploration and exploitation. Note that in our computing scenarios, the value of each sub-action $a_{t,i}$ is discrete, that is, where the specific DNN block will be performed. Therefore, we introduced a piecewise function in the original DDPG model before the action acts on the environment as shown in Figure 7, thus realizing the discretization of the continuous actions in DDPG. Specifically, it is generated as follows:

$$a_{t,i}^{\mathrm{MEC}} = \begin{cases} 1 & 0 < a < \varepsilon \quad \text{(mobile Web browser)} \\ 2 & \varepsilon \le a < \varphi \quad \text{(network edge server)} \\ 3 & \varphi \le a < 1 \quad \text{(remote cloud server)} \end{cases}.$$

The pre-defined thresholds ε and φ here refer to the intent of the actions distribution from service provider. The greater the value of ε, the more computations are intended to be completed on the mobile Web browser (i.e., the savings in deployment overhead are more important). Otherwise, the service provider is more focused on the user service experience (i.e., places more DNN computations on the network edge or remote cloud servers), which will in contrast result in an increase in the deployment cost. The value of φ refers to the intent to assign the DNN computation to either edge or cloud servers.

Similarly, based on the intention of actions distribution, the value of each sub-action $a_{t,i}$ in the D2D-based computing scenario is given by

$$a_{t,i}^{\mathrm{D2D}} = \begin{cases} 1 & 0 < a < \zeta \quad \text{(service subscriber)} \\ 0 & \zeta \le a < 1 \quad \text{(specified collaborator)} \end{cases}.$$

Based on the existing experience, the Actor and Critic update the network parameters during each learning phase. The experience replay mechanism significantly reduces the correlation in the training samples and thus quickens the convergence. Specifically, DDPG leverages neural networks as the function approximator; and here we only use one hidden layer with 50 and 30 neurons for Actor and Critic

networks. Overall, the IoRLO uses the same structure and training method as DDPG [27], but introduces an intent-oriented action generation module for DNN block assignment according to the specific service requirements. In the practical applications, we can also directly use the pre-trained Actor network to generate corresponding actions (that is, partitioning strategies). However, it is necessary to design and train the Actor and Critic networks in DDPG according to different DNN models in advance.

5 Performance Evaluation

In this section, we first detail the method and settings (see subsection 5.1), followed by the evaluation of the proposed intent-oriented DNN computation partitioning algorithm IoRLO (see Section 1 in Supplementary Material), then analyze the results for a collaborative mobile Web AR application in an actually deployed 5G trial network (see subsection 5.2), which has been supported by China Mobile Communications Group Beijing Co., Ltd. and Huawei Technologies Co., Ltd. What we considered in the experiments are the response latency, energy consumption, and throughput of the system, the factors most valued by service subscribers and providers.

5.1 Method and Settings

For the sake of clarity, in this part, we detail the benchmark of the computation offloading approaches, the DNN architectures, as well as the datasets, and also the performance of the 5G trial network in our experiment.

5.1.1 Experimental Environment

We illustrate the experimental communication network environment in Figure 8. End devices connect to the Internet via Customer Premise Equipment (CPE), and edge servers are deployed at the 5G base station to provide AR services.

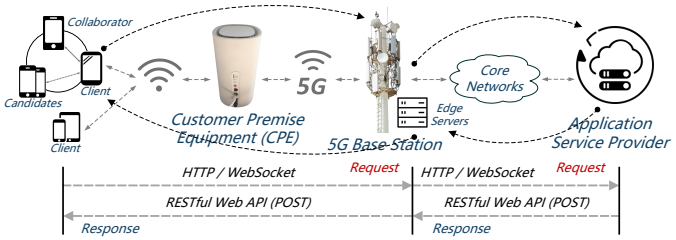


Fig. 8. Experimental 5G network environment.

In practice, the container setting of edge and remote clouds and the service provisioning in our system are similar to traditional Web application services. Specifically, both the edge server and cloud server can use Flask or Apache Tomcat technology to provide collaborative service APIs (including the Get and Post requests) for service collaboration and data transmission over the cloud, the edge, and end devices. For the convenience of DNN inference, we adopt the Flask (which has better support for the DNN models) to provide computing and data request services. Note that considering the stability and continuity requirements of the data transmission, we can also adopt WebSocket/Socket in the testbed as the transmission protocol between different devices to verify the effectiveness of the proposed fine-grained elastic partitioning algorithm.

In addition, we present the specification of the 5G trial network performance and the details of the devices used in the experiment (see Table 2). Since the network is still in the experimental stage, the performance is occasionally unstable, but it performs well on average. Moreover, the CPU frequency of all Android devices can be artificially controlled by obtaining root permission.

TABLE 2
Specification of 5G Trial Network and Details of the Devices

Bandwidth (mbps) Uplink / Downlink	Min.	Max.	Avg.	Loss Tolerance
Device – to – Network Edge	74.4 / 332.5	77.9 / 440.3	76.1 / 382.4	0
Network Edge – to – Cloud	72.2 / 466.3	73.9 / 600.7	73.3 / 542.3	0.045%
Device – to – Device	71.1	97.3	87.6	0.012%
End-to-End Latency (ms)	**Min.**	**Max.**	**Avg.**	**Jitter**
Device – to – Network Edge	1.61	104.25	8.76	102.64
Network Edge – to – Cloud	15.37	905.16	27.04	889.79
Device – to – Device	2.72	304.16	23.14	301.44

[1] Cloud: Ubuntu 16.04, Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.0 GHz, 128 GB RAM.
[2] Network Edge: Ubuntu 16.04, Intel Xeon E5-2600 v2 @ 2.0 GHz, 64 GB RAM.
[3] End Device: Huawei Mate 10, Android 8.0, HiSilicon Kirin 970, 4 GB RAM (Chrome).
[4] Collaborator: Xiaomi Mi 8, Android 8.1, Qualcomm Snapdragon 845, 6 GB RAM (Chrome).

5.1.2 Benchmark of DNN Architectures

For demonstration purposes, we chose four representative neural network architectures (i.e., AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1) with different computational complexity. These DNN architectures not only perform well in the field of object recognition but also serve as the backbones for other computer vision solutions such as object detection (Faster R-CNN, Mask R-CNN, YOLO, and SSD) and semantic segmentation (FCNs and DeepLab).

As discussed in subsection 3.1, all the aforementioned neural network architectures have been re-designed with multiple early exit branches and then re-trained on the datasets CIFAR-10 and CIFAR-100. Re-training large DNN architectures (e.g., ResNet-152) with dense side branches or using a complex dataset (e.g., ImageNet) is obviously time-consuming, especially with limited GPU resources. Therefore, for this research, we only adopted simple DNN architectures and datasets for demonstration purposes.

5.1.3 Benchmark of Offloading Approaches

The proposed partitioning algorithm IoRLO is designed to offload DNN computations among the heterogeneous platforms so as to balance multiple interests. For comparison, we also adopted another two status quo computation offloading approaches in our experiment. Neurosurgeon [8] is a data-centric approach, which provides an automatic DNN partitioning solution between the user and cloud based on the neural network architecture characteristics for the best response latency or energy savings, denoted by Neuro-L and Neuro-E, respectively. In contrast, MAUI [20] is a control-centric offloading approach proposed in 2010 but still famous for its function granularity (i.e., code level) offloading decision making.

5.2 Application Performance Analysis

We implemented an AR-based instance retrieval and recommendation application (see Figure 9) on a mobile Web

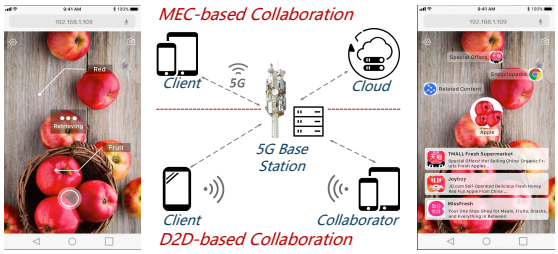


Fig. 9. Mobile Web AR application.

TABLE 3
DNN Computation Distributions in Two Collaborative Scenarios

DNN architectures	DNN Computation Distribution (%) (MEC-based Collaboration) Web Browser / Network Edge / Cloud			DNN Computation Distribution (%) (D2D-based Collaboration) Client / Collaborator			
	Neuro	MAUI	Ours	Self	Co-H	Ours	Co-L
AlexNet	0 / 0 / 100	0 / 0 / 100	9.08 / 16.68 / 74.24	100 / 0	0 / 100	15.07 / 84.93	0 / 100
VGGNet-16	0 / 0 / 100	0 / 0 / 100	2.41 / 84.30 / 13.29	100 / 0	0 / 100	3.49 / 96.51	0 / 100
ResNet-32	0 / 0 / 100	0 / 0 / 100	30.85 / 31.98 / 37.17	100 / 0	0 / 100	23.51 / 76.49	0 / 100
MobileNet-V1	0 / 0 / 100	0 / 0 / 100	47.25 / 35.22 / 17.53	100 / 0	0 / 100	47.26 / 52.74	0 / 100

* Collaboration with low-performance devices cannot improve inference latency and energy consumption in the D2D-based scenario, therefore, our approach only considers collaboration with Co-H.

browser for advertising, as an example. To mitigate the impact of annotation rendering on the performance of the mobile Web AR system, we only selected simple 2D virtual contents to present to users, which will not consume too much additional resources. Users can access this mobile Web AR service via a pre-defined URL, the specific DNN model will then be downloaded asynchronously. When the user targets an instance, such as apples in our experiment, the relevant "augmented" virtual information will be displayed on the user's equipment. Moreover, this link can also be embedded in many places to provide pervasive AR service, such as Facebook, Twitter, and WeChat.

The theme of this paper is to provide the DNN-based mobile Web AR services with multiple requirements from subscribers and providers by leveraging the collaborative mechanism in the 5G era. Here we first present the performance of Neurosurgeon and MAUI for comparison. Specifically, Neuro-L and Neuro-E aim to provide the best service response latency and mobile energy savings. As data transmission will only have a weak impact on application performance, all the DNN computations will be assigned to the cloud or collaborator for accelerating the DNN inference and saving energy in 5G networks. When the collaborator cannot provide sufficient computing capability, the self-contained approach will be better but will still offload computations to the collaborator for energy saving. MAUI chooses to send time-consuming functions to the cloud, but it also degenerates into Neuro-L. Therefore, we will consider the cloud-only (Cloud), edge-only (Edge), self-contained (Self), and collaboration (Co) in the experiment.

In contrast, our offloading algorithm IoRLO works well in 5G networks. Specifically, computation-intensive parts are more likely to be offloaded to the cloud or network edge to accelerate the inference. The others, which will not cause a great impact on the overall service latency, in contrast, will be placed on the mobile Web browser, which therefore can benefit to the service deployment overhead.

In experiments, the specific offloading decision is selected randomly from the potential partitioning decisions (i.e., the colored space) based on the specific requirements, either DNN inference latency, mobile energy consumption, and/or deployment overhead, and the DNN computation distributions in the two collaborative scenarios are illustrated in Table 3. The "actor" in IoRLO consists of one hidden layer with 50 neurons, which only needs 0.49 ms on average to generate an action. Moreover, the experiments were conducted 100 times, and all the experimental results in the papers are the average values. Note that for a practical application, the users' experience can be fed back to the cloud for further learning.

5.2.1 Service Response Latency

In this part, we present the performance of IoRLO against the other three offloading approaches in term of response latency. Specifically, the introduction of cloud and network edge computing resources in the MEC-based collaborative scenario significantly improves the service response latency by about 72.17% on average (i.e., 86.77%, 88.64%, 64.68%, and 48.60% for AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1, respectively) compared with the self-contained approaches as illustrated in Figure 10.

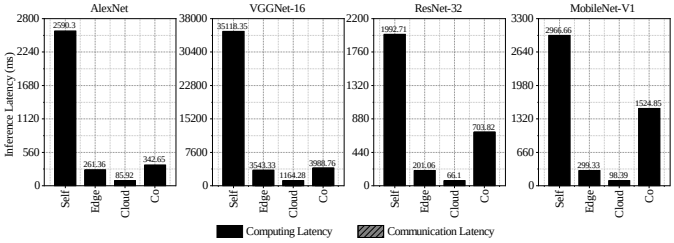


Fig. 10. DNN inference latency in MEC-based collaborative scenario.

Although there is still a performance (latency) gap with the edge-only or cloud-only approach (e.g., 261.36 ms, 85.92 ms, and 342.65 ms for AlexNet with edge-only, cloud-only, and our collaborative solutions, respectively), by assigning part of computations to the user side, it can also improve the system throughput (i.e., reduce the computational pressure of the server), compared with the other two approaches, but these results are directly related to the DNN computation partitioning decision. The improvement in throughput indicates that service providers can process more requests using the same computing resources, thereby reducing the service deployment overhead. Considering that the edge server undertakes a large number of computations, from the entire system perspective, the remote cloud can therefore serve more requests, but the system performance has not been improved as this partitioning decision causes the throughput bottleneck of this edge domain. But overall, under the premise that QoS is satisfied, this collaborative approach obviously reduces the deployment overhead, thus striking a balance between various requirements.

Remarkably, the process throughput on the network edge and remote cloud are different, and the system overall throughput is bounded by the smallest one for the collaborative scenario. The edge and cloud servers adopt buffer queues to cache the intermediate results of DNN for processing the accumulated service queries.

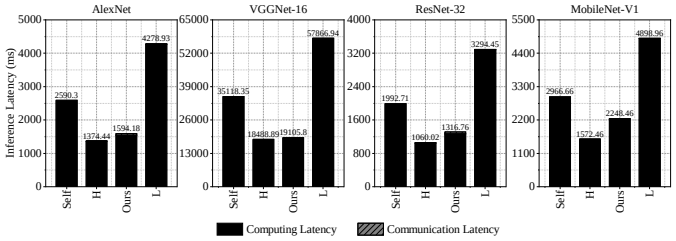


Fig. 11. DNN inference latency in D2D-based collaborative scenario.

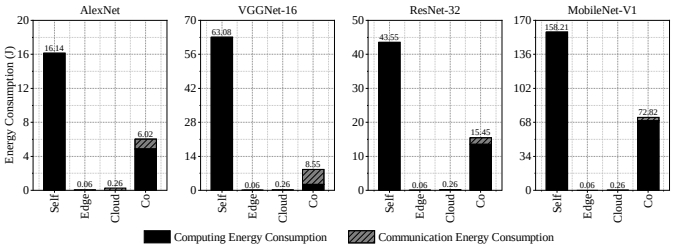


Fig. 12. Energy consumption in MEC-based collaborative scenario.

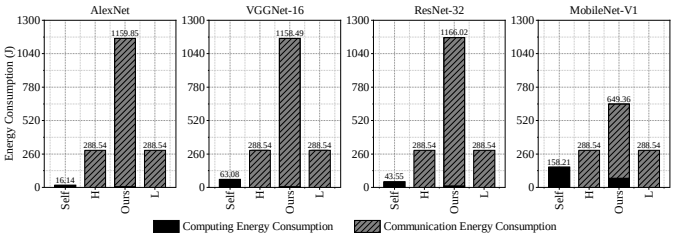


Fig. 13. Energy consumption in D2D-based collaborative scenario.

But for the D2D-based collaborative scenario, in which all the computations are completed on the mobile devices, viz. the service subscriber and collaborator, the computing capability of the collaborator is set to 100% (Co-H) and 60% (Co-L) available respectively; the capability of the collaborator will exert a great influence on the service experience. When the computing capability of the collaborator is limited, this will apparently result in service response latency degradation. As illustrated in Figure 11, the powerful collaborator brings about 47.02% service response latency improvement on average (46.94%, 47.35%, 46.81%, 47.00% for the AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1) compared with the self-contained approach. Similarly, by placing part of computations to more powerful collaborator, our proposed approach can also achieve 38.46%, 45.60%, 33.92%, 24.21% latency improvement.

Obviously, the computing capability of different devices is different. Assigning part of DNN inference computations to devices with weaker computing capability will undoubtedly increase the processing time, which will affect the subsequent DNN computations on other devices, and thus increases the overall waiting delay. Fundamentally, both our proposed DNN fine-grained partitioning-inference scheme and DDNN [28] are designed for collaborative serial DNN inference among multiple devices, which will inevitably cause the problem of synchronization between different devices. But in our DNN partitioning solution, we have considered the impact of the waiting delay that may be caused by the different devices on the service performance. In this part, the user device, edge server, cloud server, and nearby collaborator devices all have different computing capabilities, but as we discussed above, the DNN partitioning decisions have been able to distribute complex DNN computations to high-performance devices, and simple computations that are assigned to low-performance devices will not have a serious impact on the system.

5.2.2 Mobile Energy Saving

For the purpose of energy saving, the status quo approach Neuro-E will offload all the DNN computations to the cloud or collaborator, as discussed earlier, the end-user equipment therefore only needs to shoulder the energy consumption required for communication (i.e., input transmission and result receiving), that is, 0.06 Joule and 0.26 Joule for edge-only and cloud-only approaches, respectively. The mobile energy consumption in MEC-based and D2D-based collaborative scenarios are illustrated in Figure 12 and Figure 13.

In contrast, with our proposed offloading algorithm, users need to pay some energy for DNN inference on the mobile Web browser. But it can still bring considerable energy savings, about 66.91% on average compared to the self-contained approach for MEC-based scenario (i.e., 62.70%, 86.45%, 64.52%, and 53.97% energy savings, for the AlexNet, VGGNet-16, ResNet-32, and MobileNet-V1, respectively).

However, for service provision in the D2D-based scenario, our proposed approach will consume more energy, which is used for data transmission between mobile devices. Why does it still need collaboration for DNN inference? It depends on the specific purpose of the user, such as DNN inference acceleration or mobile energy saving.

Another observation is that the more frequent the data (intermediate computing results) transmission, the higher the energy consumption, and vice versa. In general, the Co-H method, that is, offloading all DNN computations to the collaborator device with higher computing capability, can obviously achieve less inference latency and energy consumption, which is more of an ideal situation. Because the performance of mobile devices is continuous dynamically changing, collaborators with high performance do not always exist. If all computations are offloaded to the collaborator with lower computing capabilities, that is, the Co-L method, it will lead to unacceptable service response latency. While our proposed collaborative computing solution can reasonably orchestrate the computing resources of local and collaborator devices to achieve better DNN inference latency, but at the same time user device needs to pay additional communication energy consumption. Therefore, the choice of computing mode in the D2D-based collaboration scenario depends on which service performance metric the user pays more attention to. For example, when the user device has sufficient energy, our approach can thus achieve better service response latency; however, when the user is more concerned about the energy consumption (or the user device faces the energy shortage problem), adopting the Co-L method can effectively extend the service time, but the QoS will be compromised.

6 Related Work

Many efforts have focused on the acceleration of DNN inference, (1) model compression; (2) architecture optimization; and (3) dedicated accelerator development (e.g., GPU, FPGA, ASIC); However, adopting cloud-centric approaches undoubtedly incur a high response latency in dynamic and noisy mobile networks, and thus result in user experience degradation. To eliminate the negative effects of data transmission, the DNN computations can also be placed on mobile devices to achieve off-line inference benefiting from the design of lightweight DNN architectures [29], [30], [31], but this approach cannot strike a balance between accuracy and efficiency. The computation offloading mechanism, therefore, attracted our attention.

However, current cloud-based offloading solutions still face challenges, (1) coarse-grained partitioning [20], [32], which cannot fully explore the DNN structural characteristics and balance the accuracy and efficiency (e.g., BranchyNet [10]). Terminating the inference process too early will cause insufficient accuracy, but terminating too late will introduce additional redundant computations; (2) lack of comprehensive consideration of user experience and service deployment overhead (e.g., Neurosurgeon [8] and Edgent [33]); (3) static and aptotic scheduling [21], [28], [34] cannot cope with the dynamically changing environment.

Fundamentally, both the DNN computation offloading discussed in this paper and the service offloading on mobile computing environment are aimed at optimizing service performance by orchestrating distributed resources. However, the current service offloading mechanisms are mainly focused on the collaboration between the mobile device and network edge/remote cloud [35], [36], [37]; Because of network performance limitations, these offloading collaborative scenarios are relatively simple. While in 5G networks, the MEC and D2D technologies make the computing environment more complex, but there is still a lack of research on collaborative computing optimization for latency-sensitive DNN-based services.

The emerging 5G networks together with a variety of promising features provide us with opportunities for the collaborative computing of distributed DNNs. But all the current approaches will degenerate into the cloud-based mode [8], [20], therefore, although the user experience is satisfied, the service provider will undoubtedly have to pay a heavy expenses. Unlike previous efforts, we have investigated the collaboration of the cloud, network edge, and mobile devices, thus bringing about a distributed DNN, (1) the collaborative mode satisfies the application performance requirement and saves the deployment overhead as well; (2) layer granularity offloading supports fine-grained computation partitioning for distributed collaboration; and (3) by considering multiple factors, it therefore achieves adaptive computation scheduling.

7 Conclusion

In this paper, we have presented the fine-grained elastic computation partitioning mechanism for distributed DNN in 5G networks. This collaborative solution provides a promising approach to balance the interests of both service subscribers and provider. We elaborated two collaborative computing scenarios for the 5G era by leveraging the MEC and D2D technologies, then presented the computing system. Aimed at layer granularity computation scheduling, we investigated per-layer inference latency and energy consumption prediction models for DNN, followed by the DNN partitioning scheme, which provides a generalized approach. The experiments were conducted in an actually deployed 5G trial network based on our developed Web-based mobile AR application, and the results show the superiority of this collaborative approach.

Acknowledgments

This work was supported in part by the Funds for International Cooperation and Exchange of NSFC under Grant 61720106007, in part by the National Key R&D Program of China under Grant 2018YFE0205503, and in part by the 111 Project under Grant B18008.

References

[1] M. Billinghurst, A. Clark, and G. Lee, "A Survey of Augmented Reality," *Foundations and Trends in Human–Computer Interaction*.
[2] X. Qiao, P. Ren, S. Dustdar, and J. Chen, "A New Era for Web AR with Mobile Edge Computing," *IEEE Internet Computing*, 2018.
[3] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, and J. Chen, "Web AR: A Promising Future for Mobile Augmented Reality–State of the Art, Challenges, and Insights," *Proceedings of the IEEE*, 2019.
[4] Y. Ma, D. Xiang, S. Zheng, D. Tian, and X. Liu, "Moving Deep Learning into Web Browser: How Far Can We Go?" in *ACM 2019 The World Wide Web Conference*.
[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet Things J.*, 2016.
[6] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," *IEEE Commun Mag*, 2017.
[7] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-Device Communication in 5G Cellular Networks: Challenges, Solutions, and Future Directions," *IEEE Commun Mag*, 2014.
[8] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," *SIGARCH Comput. Archit. News*, 2017.
[9] P. Panda, A. Sengupta, and K. Roy, "Conditional Deep Learning for Energy-Efficient and Enhanced Pattern Recognition," in *IEEE 2016 DATE*.
[10] S. Teerapittayanon, B. McDanel, and H. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks," in *IEEE 2016 ICPR*.
[11] D. G. Lowe, "Distinctive Image Features from Scale-invariant Keypoints," *Springer 2004 IJCV*.
[12] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *Springer 2006 ECCV*.
[13] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE 2011 ICCV*.
[14] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object Retrieval with Large Vocabularies and Fast Spatial Matching," in *IEEE 2007 CVPR*.
[15] X. Qiao, P. Ren, G. Nan, L. Liu, S. Dustdar, and J. Chen, "Mobile Web Augmented Reality in 5G and Beyond: Challenges, Opportunities, and Future Directions," *China Communications*, 2019.
[16] P. Ren, X. Qiao, J. Chen, and S. Dustdar, "Mobile Edge Computing–a Booster for the Practical Provisioning Approach of Web-Based Augmented Reality," in *IEEE/ACM 2018 SEC*.
[17] B. L. R. Stojkoska and K. V. Trivodaliev, "A Review of Internet of Things for Smart Home: Challenges and Solutions," *Journal of Cleaner Production*, 2017.
[18] M. Jia and W. Liang, "Delay-Sensitive Multiplayer Augmented Reality Game Planning in Mobile Edge Computing," in *ACM 2018 MSWiM*.
[19] N. Gavish, T. Gutiérrez, S. Webel, J. Rodríguez, M. Peveri, U. Bockholt, and F. Tecchia, "Evaluating Virtual Reality and Augmented Reality Training for Industrial Maintenance and Assembly Tasks," *Interactive Learning Environments*, 2015.

[20] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *ACM 2010 MobiSys*.

[21] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud," in *ACM 2011 EuroSys*.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *IEEE 2015 CVPR*.

[23] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications," in *IEEE 2018 ECCV*.

[24] P. Ren, X. Qiao, Y. Huang, L. Liu, S. Dustdar, and J. Chen, "Edge-Assisted Distributed DNN Collaborative Computing Approach for Mobile Web Augmented Reality in 5G Networks," *IEEE Network*, 2020.

[25] A. Haas, A. Rossberg, D. L. Schuff, B. L. Titzer, M. Holman, D. Gohman, L. Wagner, A. Zakai, and J. Bastien, "Bringing the Web up to Speed with WebAssembly," in *ACM 2017 PLDI*.

[26] S. Kitanov, B. Popovski, and T. Janevski, "Quality Evaluation of Cloud and Fog Computing Services in 5G Networks," in *Enabling Technologies and Architectures for Next-Generation Networking Capabilities*.

[27] 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," *arXiv preprint arXiv:1509.02971*, 2015.

[28] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices," in *IEEE 2017 ICDCS*.

[29] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized Convolutional Neural Networks for Mobile Devices," in *IEEE 2016 CVPR*.

[30] N. Lane, S. Bhattacharya, A. Mathur, C. Forlivesi, and F. Kawsar, "DXTK: Enabling Resource-efficient Deep Learning on Mobile and Embedded Devices with the DeepX Toolkit," in *Springer 2016 MobiCASE*.

[31] B. McDanel, S. Teerapittayanon, and H. Kung, "Embedded Binarized Neural Networks," in *ACM 2017 EWSN*.

[32] M. S. Gordon, D. A. Jamshidi, S. A. Mahlke, Z. M. Mao, and X. Chen, "COMET: Code Offload by Migrating Execution Transparently," in *USENIX 2012 OSDI*.

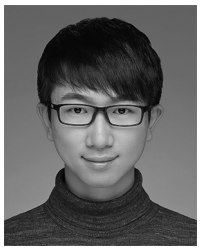
[33] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing," *IEEE Trans. Wirel. Commun.*, 2019.

[34] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: Enabling Interactive Perception Applications on Mobile Devices," in *ACM 2011 MobiSys*.

[35] A. Samanta and Z. Chang, "Adaptive Service Offloading for Revenue Maximization in Mobile Edge Computing with Delay-Constraint," *IEEE Internet Things J.*, 2019.

[36] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A Cloud-MEC Collaborative Task Offloading Scheme with Service Orchestration," *IEEE Internet Things J.*, 2019.

[37] D. Van Le and C.-K. Tham, "Quality of Service Aware Computation Offloading in an Ad-Hoc Mobile Cloud," *IEEE Trans. Veh. Technol.*, 2018.



Pei Ren is currently working toward the Ph.D. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. He is currently a Visiting Scholar with the School of Computer Science, Georgia Institute of Technology, USA, funded by the China Scholarship Council. His current research interests include the machine learning, augmented reality, edge computing, and 5G networks.

Xiuquan Qiao is currently a Professor with the Beijing University of Posts and Telecommunications, China, where he is also the Deputy Director of the State Key Laboratory of Networking and Switching Technology, Network Service Foundation Research Center of State. His current research interests include the services computing, computer vision, augmented reality, and 5G networks. Dr. Qiao was a recipient of the Beijing Nova Program in 2008 and the First Prize of the 13th Beijing Youth Outstanding Science and Technology Paper Award in 2016.

Yakun Huang is currently working toward the Ph.D. degree at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include computer vision, distributed deep learning, machine learning, augmented reality, edge computing.

Ling Liu (Fellow, IEEE) is currently a Professor at the School of Computer Science, Georgia Institute of Technology, USA. She directs the research programs at the Distributed Data Intensive Systems Lab, examining various aspects of large-scale big data systems and analytics, including performance, availability, security, privacy, and trust. Dr. Liu was a recipient of the IEEE Computer Society Technical Achievement Award in 2012. She served as the Editor-in-Chief for the IEEE Transactions on Service Computing from 2013 to 2016. She is currently the Editor-in-Chief of the ACM Transactions on Internet Technology.

Calton Pu (Fellow, IEEE) received the Ph.D. degree from the University of Washington, in 1986 and served on the faculty of Columbia University and Oregon Graduate Institute. Currently, he is holding the position of professor and John P. Imlay, Jr. Chair in Software in the College of Computing, Georgia Institute of Technology His recent research has focused on big data in Internet of things, automated N-tier application deployment and denial of information.

Schahram Dustdar (Fellow, IEEE) is currently a Professor of Computer Science with the Distributed Systems Group, TU Wien, Vienna, Austria. Dr. Dustdar was an elected member of the Academy of Europe, where he is the Chairman of the Informatics Section. He was a recipient of the ACM Distinguished Scientist Award in 2009, the IBM Faculty Award in 2012. He is also an Associate Editor of the IEEE Transactions on Services Computing, the IEEE Transactions on Cloud Computing, the ACM Transactions on the Web, and the ACM Transactions on Internet Technology.