Toward Decentralized and Collaborative Deep Learning Inference for Intelligent IoT Devices

Yakun Huang, Xiuquan Qiao, Schahram Dustdar, Jianwei Zhang, and Jiulin Li

Abstract

Deep learning technologies are empowering IoT devices with an increasing number of intelligent services. However, the contradiction between resource-constrained IoT devices and intensive computing makes it common to transfer data to the cloud center for executing all DNN inference, or dynamically allocate DNN computations between IoT devices and the cloud center. Existing approaches perform a strong dependence on the cloud center, and require the support of a reliable and stable network. Thus, it may directly cause unreliable or even unavailable service in extreme or unstable environments. We propose DeColla, a decentralized and collaborative deep learning inference system for IoT devices, which completely migrates DNN computations from the cloud center to the IoT device side, relying on the collaborative mechanism to accelerate the DNN inference that is difficult for an individual IoT device to accomplish. DeColla uses a parallel acceleration strategy via a DRL-based adaptive allocation for collaborative inference, which aims to improve inference efficiency and robustness. To illustrate the advantages and robustness of DeColla, we built a testbed and employ DeColla to evaluate MobileNet DNN network trained on the ImageNet dataset, and also recognize the object for a mobile web AR application and conduct extensive experiments to analyze the latency, resource usage, and robustness against existing methods. Numerical results show that DeColla outperforms other methods in terms of latency and resource usage, which can especially reduce at least 2.5 times latency than the hierarchical inference method when the collaboration is interrupted abnormally.

Introduction

With the rapid development of artificial intelligence (AI) chip-making technology, an increasing number of smart end devices, including Internet of Things (IoT) devices, can independently collect and process data in real-time [1, 2]. This eliminates the demand to transfer data over the cellular network to the cloud center, which reduces transmission overhead, avoids leakage of user's privacy, and reduces the load of the cloud center [3]. Also, custom embedded implementation of Deep Neural Networks (DNNs) that is suitable for IoT devices, expands the scenarios of DNNs such as driverless vehicles, security monitoring, and so on. This also illustrates the gradual expansion of AI capabilities from the remote cloud to the IoT device side, such as Huawei's HiAI 3.0, which provides an open platform for smart end devices, and supports multiple end devices to share AI capabilities.

However, executing intensive DNN computing for IoT devices is still challenging to acquire a real-time and satisfactory experience by employing the following three approaches summarized in Table. 1:

The most common way is to offload the DNN tasks to the cloud center (e.g., remote cloud and edge cloud in 5G networks), and execute the whole DNN inference there [4, 5]. This introduces the load of the cloud center and increases the dependency on the availability of the network and high-quality service of the cloud center, which also means that the AI capability of IoT devices entirely relies on the cloud center.

The second approach implements collaborative inference between the IoT device and the cloud center by dynamically partitioning DNNs, which leverages available resources of the IoT device, reduces the load of the cloud center, and somewhat accelerates DNNs inference. Neurosurgeon [6] is a classic device-cloud collaborative DNN inference scheme, which automatically chooses the partition points pursuing the optimal latency and energy consumption. LcDNN [7] and DeepAdapter [8] provide lightweight collaborative frameworks for executing distributed DNN inference between the mobile web and edge server, which also rely on the edge server. DDNN [9] further extends the collaborators, which jointly trains mapped sections of a DNN onto a distributed computing hierarchy over the cloud, the edge, and end devices. Unfortunately, this approach also has a strong dependency on the quality of network communications to exchange intermediate results of DNN inference. Especially in an extreme environment where the network or service of the cloud center is unstable, this solution may be unavailable for a scenario of the autopilot, which may cause a major hazardous accident in no-man's-land.

The third approach is to execute all DNN computations on the IoT device side, which can be classified as custom embedded implements, and distributed collaborative inference over IoT devices, neither of which is dependent on the cloud

Yakun Huang, Xiuquan Qiao are with Beijing University of Posts and Telecommunications; Schahram Dustdar is with the Technische Universität Wien; Jianwei Zhang is with the Capinfo Company Limited; Jiulin Li is with the Beijing National Speed Staking Oval Operation Company Limited.

Digital Object Identifier: 10.1109/MNET.011.2000639

0890-8044/22/$25.00 © 2022 IEEE

Technique Used	Typical Literature	Problem Addressed	Advantages	Disadvantages
Remote Execution	DeepQuery [4]	Cloud-based DNN inference	1) Low-latency; 2) High GPU utilization.	1) High pressure of the cloud server; 2) Requiring high network bandwidth; 3) Privacy concerns.
	MobiEye [5]		1) Deep feature flow; 2) Dynamic scheduling.	
Partitioning -offloading	Neurosurgeon [6]	Distributed DNN inference over cloud, edge, and device	1) DNN computation partitioning across the cloud and mobile edge.	1) Requiring retraining the DNN model; 2) Requiring ingenious design of the structure and branches; 3) Requiring an available environment between the device and the cloud.
	LcDNN [7], DeepAdapter [8]		1) Lightweight branch at device side; 2) High throughput; 3) Reduce the pressure of the cloud.	
	DDNN [9]		1) A distributed computing hierarchy; 2) Joint training method.	
Device-side inference	Compression DNNs [10]	Device-side DNN inference	1) Small DNN model size; 2) Fast inference.	1) Accuracy loss; 2) Design the network with expert knowledge.
	Musical Chair [11], Hadidi [12]		1) Distributes the DNN inference among multiple IoT devices; 2) Dynamically changes the distributed tasks.	1) Hierarchical inference characteristic; 2) High waiting latency; 3) Low robustness.

TABLE 1. Summary of existing approaches for executing DNNs on IoT devices.

center or network resources. Custom embedded implements require extensive expert experience and skills to design DNNs with low accuracy loss (e.g., MobileNet, ShuffleNet, and so on) [10], or various compression technologies (e.g., low-rank factorization, knowledge distillation, quantization, and pruning) to reduce unimportant weights and neurons of trained models to acquire lightweight DNNs [8]. Currently, only a small portion of DNNs such as image recognition DNNs, has a low accuracy loss and is difficult to generalize to other areas. DNN inference solutions for IoT device-side collaboration are important ideas to address this challenge. Musical Chair [11] implements efficient real-time recognition using collaborative IoT devices, which proposes model and data parallelism by layers to distribute the computation of a model over multiple IoT devices. The authors further put forward to conduct single-batch influence in real-time while exploiting several new model-parallelism methods with collaborative IoT devices in [12].

Despite the success of implementing collaborative DNN inference over multiple IoT devices [11, 12], and removing the dependency on the cloud center, there are still two key challenges for employing this approach in real applications, which include the following:

- Traditional collaborative approaches execute DNN layers with obvious sequential characteristics, which may cause a large number of IoT devices to be waiting, thus failing to further accelerate the DNN inference. They partition the DNN into multiple sequential sub-tasks by layers and distribute them across IoT devices for sequential execution. This may lead to a situation where one IoT device is performing a sub-task while the others are waiting, failing to improve the resource utilization of IoT devices. When an anomaly occurs in one of the assigned IoT devices, this DNN layer may require reassigning a new IoT device and broadcasting updated collaboration to each IoT device.
- Traditional collaboration typically uses offline and linear models to predict the execution latency of DNN layers and assigns DNN computations to IoT devices for collaboration according to such simple predictions. However, these methods struggle to achieve optimal resource utilization and latency performance in dynamic environments. These approaches require maintaining an IP routing table on each IoT device for collaborative computing, which increases communication costs and reduces robustness. This happens because IoT devices execute different DNN layers inference separately, performing sequential feature and monitoring the status of IoT devices and inference dependencies in a timely manner.

To address the first challenge, we describe the overall system architecture of the proposed DeColla, including a remote cloud layer, an edge cloud layer, and an IoT device layer in 5G networks, which weakens the role of the cloud center during online collaborative DNN inference. The remote cloud can provide offline training and optimization of precise DNN models, and the edge cloud trains a dynamic task assignment engine for assigning computations, which has no impact on online real-time collaboration at the IoT device layer. We further enhance the efficiency and the robustness of DNN collaborative inference for IoT devices by a novel mechanism for parallel inference in DNN layers. Unlike the traditional hierarchical inference, we implement collaboration of multiple IoT devices at a fine-grained level to each DNN layer, rather than the entire DNN inference. In other words, multiple IoT devices work together to execute each DNN layer to replace traditional methods of a single IoT device to perform one or several DNN layers independently. To address the second challenge, we provide a DeColla engine trained on the edge cloud to provide optimal and adaptive assignment, performing parallel inference in DNN layers for IoT devices. This distinguishes it from the traditional method of assigning DNN sub-tasks using offline predicted latency models. We leverage a message queue to judge whether the parallel inference on the current DNN layer is complete or not. Instead of broadcasting and updating the

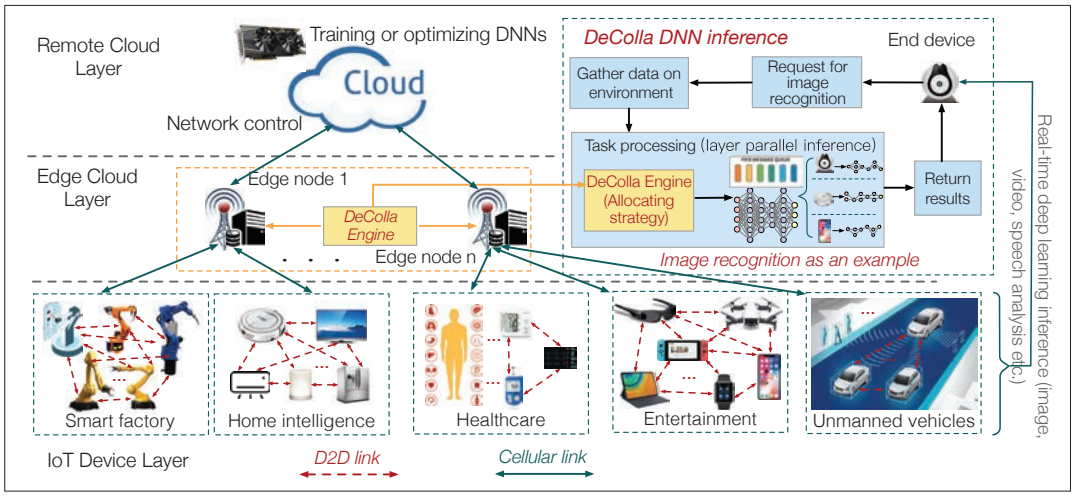

FIGURE 1. Overall system architecture.

message queue on all IoT devices, we just need to initialize and update it on the task requester, reducing the communication cost during the collaboration. Since the message queue records the results of the parallel inference for IoT devices in a DNN layer, DeColla can determine whether a delay or exception has occurred for IoT devices by the results that have been returned. We can see that this collaborative mechanism is more effective in improving the resource utilization of IoT devices, reducing the waiting latency, and improving the robustness when compared with existing methods. We use a case study on image recognition of mobile augmented reality and conduct experiments to indicate that DeColla offers better latency and resource usage than existing approaches. Our contributions can be summarized as:

- We propose DeColla, providing a collaborative mechanism for parallel inference within the DNN layer, and enhancing the efficiency and the robustness of decentralized and collaborative DNN inference.
- We give detailed methods of DeColla to provide a DRL-based adaptive allocation strategy that greatly improves resource utilization of IoT devices, which also uses a message queue to replace the traditional IP routing table, reducing the waiting latency during collaboration.
- We build a testbed and deploy a simplified demo of mobile AR applications. The experimental results demonstrate that DeColla performs better than existing collaboration methods in terms of latency and system robustness.

System Architecture

In Fig. 1 we present the overall system architecture of DeColla, including a remote cloud layer, an edge cloud layer, and local IoT devices layer. To better understand the workflow of DeColla, we use an example of image recognition to describe the architecture details of the system. First, an IoT device with a camera launches a task request for real-time object recognition. The IoT device then collects the contexts needed for collaborative computing such as the network bandwidth, the number of IoT devices, and their real-time status. With this necessary information, the DeColla engine can dynamically and adaptively assign computations of each DNN layer to collaborative IoT devices for parallel inference. In contrast to existing methods, our parallel inference aims at each DNN layer rather than the entire DNN network. During the parallel inference, we use an output message queue to judge whether the current layer has completed all calculations. Once the current DNN layer has been successfully processed, we remove to the next DNN layer and repeat the same processing until the final result is obtained, and return it to the initial IoT device.

We further describe these layers of the system architecture in detail to better understand the idea of the DeColla system.

Remote Cloud Layer: In the DeColla system, a remote cloud layer is often viewed as a computationally resource-rich cloud center that is used to train or optimize DNN models, as accurate DNN models are key to ensuring satisfactory service at the IoT device layer. It is also responsible for network control and the deployment of DNN services between the remote cloud and the edge cloud. Updating DNN services to the edge cloud in real-time can ensure the reliability and accuracy of DNN models.

Edge Cloud Layer: Unlike the device-cloud collaborative inference approach, the edge cloud layer of DeColla is not involved in the DNN inference processing at all, which is mainly responsible for the initialization and optimization of the DeColla engine. Note that the DeColla engine provides a dynamic and adaptive allocation strategy for performing each DNN layer inference collaboratively and in parallel. On the one hand, there is no need to place the training phase of the DeColla engine at the IoT device layer, which avoids the resource and energy consumption of IoT devices. On the other hand, the edge cloud can provide offline training for a more precise allocation strategy of the DeColla engine.

IoT Device Layer: Our DeColla system can be applied to a wide range of scenarios, which are filled with a large number of connected IoT devices and provide a credible environment, such as smart factories, home intelligence, healthcare, and so on. In the IoT device layer, we use the available resources of collaborative IoT devices to accelerate DNN inference. There is no dependency between

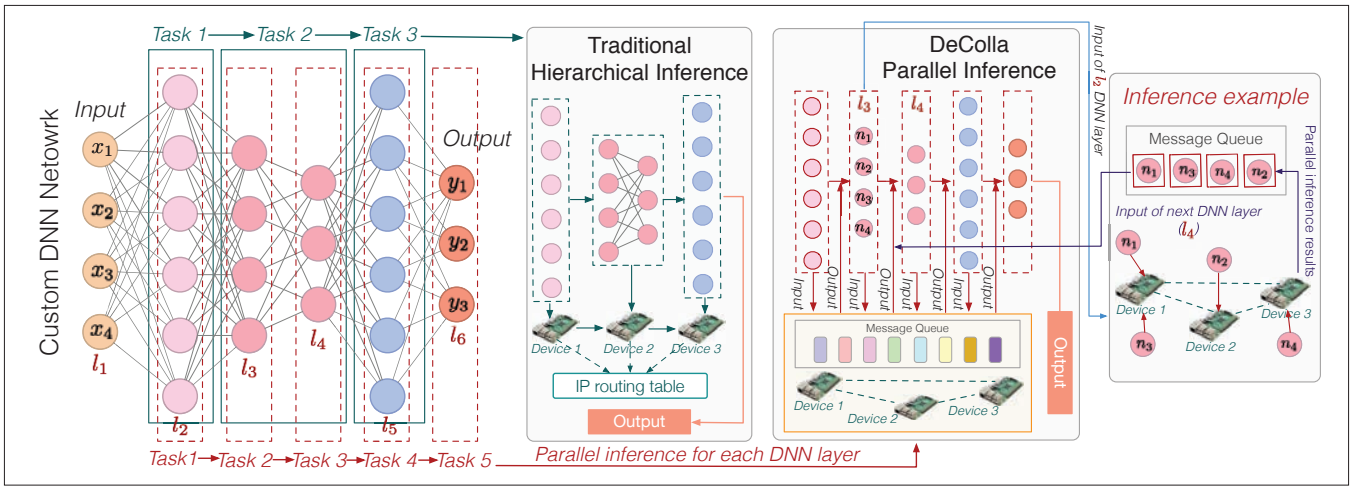

FIGURE 2. Various DNNs execution schemes for acceleration and collaboration.

DNN calculation assigned to each IoT device, which also does not rely on the cloud center and can be executed independently. We periodically update the DeColla engine to the IoT device from the edge cloud when the IoT device is idle, and feedback history behaviors to enhance the DeColla engine's adaptive capabilities. Note that the technical details of communication in the IoT device layer are beyond the scope of this article.

Methods of the DeColla Engine

The DeColla engine plays the role of the brain in implementing decentralized and collaborative inference over IoT devices. The engine provides a dynamic and adaptive allocation strategy for collaboratively executing each DNN layer in parallel (i.e., an optimal allocation that provides decentralized collaborative inference for each DNN layer with optimal processing latency based on dynamic contexts, including the number of IoT devices, available computing resource, and the network condition). We first introduce our parallel DNN layer inference in DeColla, which is more suitable than the existing hierarchical and collaborative inference.

How to Execute Decentralized and Collaborative DNN Inference in Parallel?

We show the difference between traditional hierarchical inference and proposed parallel inference for faster collaboration in Fig. 2. It can be seen that in traditional hierarchical inference, for a four-layer (excluding the input and output layers) DNN network, it can be divided into different sub-tasks and assigned to the allocated IoT devices to execute the partial inference. However, this inference scheme has a typical sequential nature during collaborative inference, whereby the latter device needs to wait for the previous device's inference results as the input. Also, this hierarchical execution scheme reduces the stability and reliability in real scenarios due to the following considerations. First, maintaining an IP routing table across participating devices requiring reallocation and updating changes when one collaborative device occurs abnormity or lost the connection, which causes the increase of the waiting latency. Second, the fluctuation of the network bandwidth may cause inconsistent updating of the IP table, and even lead to paralysis of current distributed collaborative inference. Since there are currently no methods to keep the consistency and integrity of the IP table, it is also the potential reason for the increase of the waiting latency when applied in actual scenarios. Thus, although the IP table itself is small and does not need to consume computing resources, it is easy to be affected by the dynamic environment and running state of collaborative devices.

Unlike traditional hierarchical inference, we propose a novel way to implement parallel inference for accelerating multiple IoT device collaboration within the DNN layer, using a message queue to effectively reduce the waiting latency. When any one of the collaborative devices occurs an abnormity or leaves the current collaboration, the message queuing mechanism can quickly discover and re-allocate the task in time without affecting the other collaborative devices, avoiding a large amount of waiting latency. Instead of offloading one or more DNN layers to a single IoT device, we use multiple IoT devices in parallel for executing each DNN layer. We simply maintain a message queue on the IoT device requester and quickly decide whether the inference of the current DNN layer is completed by judging whether the length of the message queue equals the number of neurons in the current DNN layer. After that, we simply repeat each DNN layer inference until we obtain the result of the last DNN layer. Hence, DeColla can provide a more efficient and robust collaborative mechanism to accelerate distributed DNN inference across multiple devices.

We give an example of parallel inference at layer l_3 in the right part of Fig. 2. It can be seen that we take the output of layer l_2 as input to layer l_3, and assign the computations associated with the neuron in layer l_3 to different IoT devices for executing parallel inference. With this kind of intra-layer parallel inference, there are no computational dependencies among IoT devices. The neuron of the red box in the right part of Fig. 2 indicates that the computation associated with that neuron has been completed. When the length of the message queue equals the number of neurons in the DNN layer, the output can be combined in the order of the input to serve as the input to the next DNN layer. The most critical problem with this parallel inference is how to provide a way to

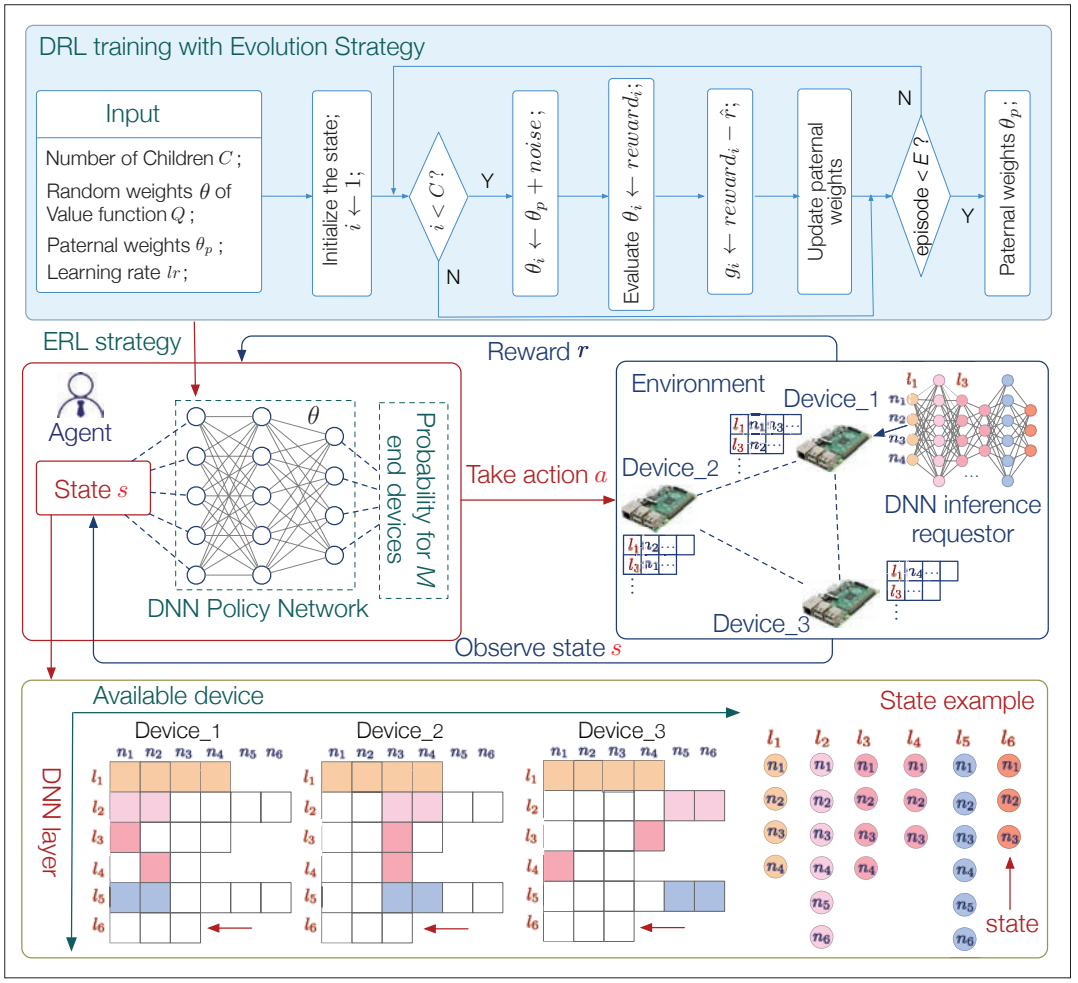

FIGURE 3. Adaptive allocation strategy for executing DNN layers in parallel based on DRL. In the state example at the bottom, we have completed the neuron (which means the sub-task) allocation of l_5, and are considering how to allocate neurons of l_6 layer into collaborative IoT devices. The blank blocks represent the remaining available resources of IoT devices.

dynamically allocate computations of each layer to IoT devices, with the goal of making the overall latency minimal. We next focus on how to provide optimal neuron allocation based on dynamic contexts (e.g., network and IoT devices status).

Adaptive Allocation Strategy for DNN Layers in DeColla

To take full advantage of our parallel inference scheme of each DNN layer for non-dependent DNN inference in DeColla, it is also vital to design an on-line scheduling algorithm to allocate subtasks of each DNN layer for executing collaborative inference across IoT devices. To solve this problem, in Fig. 3 we propose DRL-ES, a real-time dynamic allocation algorithm based on deep reinforcement learning (DRL), which is a machine learning technology that learns which allocation behavior can yield better rewards by observing the state of the environment through an agent. During each time slot t, the agent selects an optimal action a_t that means how to allocate the neurons to IoT devices by observing the current state s_t. Especially, the agent selects the optimal action according to a policy network that is performed by a deep neural network representing probability distribution $\pi(s, a)$ to better fit the state-action value function. We describe the gradient of the optimized objective to maximize the expected reward as:

$$\Delta E_\theta[\Sigma\gamma^t r_t] = E_{\pi_\theta}[\Delta_\theta \log \pi_\theta(s, a) Q^{\pi\theta}(s, a)], \quad (1)$$

where $E_\theta[\Sigma\gamma^t r_t]$ represents the expected cumulative reward and $\gamma \in [0, 1]$ is the discount reward. $Q^{\pi\theta}(s, a)$ is the expected reward when the agent picks action a in state s. Assuming P_j is the required processing time of the neurons set j (i.e., the latency between the receiving neurons set and completing the computing of the neuron set), and T_j is the theoretical processing latency by calculating FLOPs of the neuron set j. Thus, we minimize the average slowdown of processing the neuron set, represented by $S_j = P_j/T_j$, to pursue an optimal allocation for proposed parallel inference.

With such a basic framework, we define the basic elements of the state, the action, and the reward of this DRL problem to better understand the proposed scheduling algorithm as follows:

State Space: At each time slot t, the state S_t can be denoted by $S_t = <l_t, B_t, C_t, n_t>$, where l_t is the current DNN layer and B_t is the network bandwidth. $C = <c_1^t, c_2^t, ..., c_N^t>$ represents the available computing resource of N IoT devices, and n_t denotes the current neurons in l_t. Since the number of IoT devices is always not constant, this indicates that we have to retrain the DRL policy network when a part of IoT devices join or leave

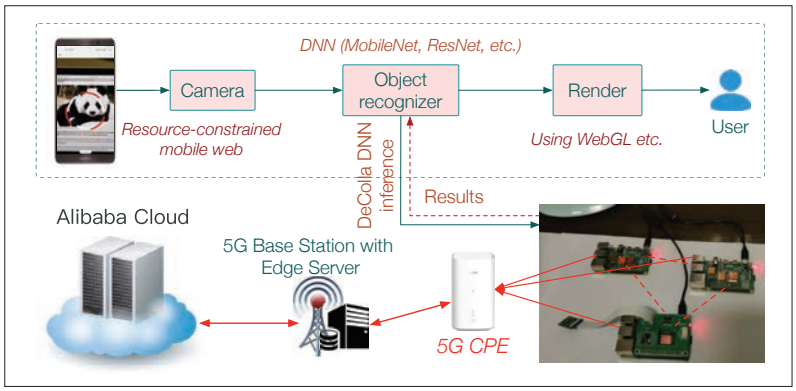

FIGURE 4. System testbed for recognizing case of a mobile web AR application.

the collaboration, which is not practical. Therefore, we fix the number of available IoT devices in the policy network to always be constant value M_c, and when the number of available IoT devices M_a is more than M_c, the highest available resources in the first M_c will be selected as the collaboration IoT devices. If $M_a < M_c$, $M_c - M_a$ excess IoT devices are set as ø, indicating that these IoT devices are not involved in collaboration.

Action Space: Based on the above definitions, the real-time scheduling algorithm requires matching sub-tasks of N neurons in each DNN layer to M_c IoT devices appropriately, using a_t to denote the selected allocation result at time slot t. However, the large-scale neurons of DNN layers cause a large amount of action space of M_c^N, which means each neuron can be assigned to any available IoT device. Also, it is hard to train such a policy network with a dynamic and large number of neurons in the state. Since the amount of computations involved in each neuron is too small and frequent allocation may increase the communication cost and reduce the stability, it is unnecessary to allocate the computing to IoT devices at the level of each neuron. To this end, we always divide the neurons of each DNN layer equally into K pieces to keep the state stable and significantly reduce the action space. In particular, the value of K can be determined by the maximum number of neurons of the DNN network and the number of available IoT devices.

Reward: DeColla's parallel inference for each DNN layer utilizes the computing resource of collaborative IoT devices without data transmission with the cloud center. As a result, more IoT devices involved in parallel inference will accelerate the DNN layer inference and reduce the overall latency. To guide the agent to minimize the average slowdown, we employ the similar definition of the reward $R = \Sigma_{j \in n_t}(-1/T_j)$, which has been illustrated to be effective in [13]. n_t is the current waiting processing neurons at time slot t. We can maximize the cumulative reward to mimic that minimizing the average slowdown when setting the discount factor as $\gamma = 1$. Based on the above definitions, to better train the DNN policy network by reverse propagation and relieve the problem of sparse rewards, we adopt a DRL training approach with a fusion evolutionary strategy [14, 15]. As shown in the upper part of Fig. 3, the specific process is applying evolution strategy to candidate sample populations by increasing the random noise and producing offspring that perform the selection. The more well-adapted offspring have more opportunities to retain and produce new offspring.

A Case Study on Mobile Augmented Reality

Since it is promising to use the cross-platform mobile web to develop various applications, especially popular AR applications recently [3], in Fig. 4 we built a testbed to illustrate the performance of the DeColla system when employing it to recognize the object for a mobile web AR application. As object recognition is the key step in AR applications, which is currently still challenging to execute intensive and heavy DNNs on resource-constrained mobile web, we leverage DeColla to provide efficient collaboration with three IoT devices for accelerating the DNN recognition and relieve the shortcoming of the mobile web. As shown in Fig. 4, users first scan the object (e.g., pandas) via the camera invoked by the web API, and then identify the object by executing a precise MobileNet DNN network trained on the ImageNet dataset, whose model size is about 14 MB. Once the mobile device receives the correct recognition result, the render module performs the 3D animated model for interaction using WebGL and other technologies. Hence, we can employ this use case to evaluate the performance and efficiency of DeColla from the perspective of latency, and resource usage compared with existing methods such as traditional hierarchical inference.

Experiment Settings

As shown in the bottom part of Fig. 4, we use a high-performance server with two GPU cards at the remote cloud layer, which is responsible for training DNNs and providing services for the edge server. A common server with a six-core interprocessor of 2.9 GHz and 16 GB RAM is deployed near the 5G base station, which is a real-world 5G network at Beijing University of Posts and Telecommunications supported by China Unicom. We also use a HUAWEI 5G CPE to connect to the base station to provide stable communication. We use a Huawei honor smartphone as a DNN task requester, and to complete visual rendering. Three RaspBerry 4 Pi's with 4G RAM are used to complete collaborative DNN inference. Since executing DNN inference on the mobile web is mainly done by JavaScript, here we only use a smartphone as a task requester and render to simplify heterogeneous DNN inference, which means all DNN calculations are executed entirely in collaborative IoT devices. To train the DeColla engine and provide an initialized allocation strategy model, we experimentally obtain real measurements of parameter configuration, IoT device available capacity, network bandwidth, and inference latency. We generate simulation data for training the DRL algorithm at different DNN layers to obtain an optimal parallel collaborative inference strategy in various DNN models, network conditions, the available resource of IoT devices, and so on. We analyze DeColla's performance against mobile-only, partition-offloading, and the traditional collaborative method in the latency and the resource usage variance of IoT devices. The mobile-only offloads all the computations to any one IoT device. The partition-offloading approach distributes the computations between any one IoT device and the edge server,

and traditional hierarchical inference, named Hierarchical Colla, distributes DNN layers to multiple IoT devices layer by layer.

Numerical Results and Analysis

Based on the above testbed and experimental settings, in Fig. 5 we discuss the performance of the proposed DRL-ES allocation algorithm from the perspective of convergence and scheduling efficiency. For training the scheduling algorithm, we set the learning rate as lr = 0.002 and the output size of the policy network as M_c =100, respectively. We also set K = 6 to divide neurons of each DNN layer into twice the number of IoT devices. We define the training iterations as 1500 and set the number of children as C = 10 in the evolution strategy. Especially, Figs. 5a and 5b describe the convergence performance of DRL-ES on the average slowdown and total reward compared with other representative approaches.We observe that DRL-ES converges faster than the basic DRL method and has better average slowdown performance as the iteration increases, which also means that employing evolution strategy can provide DeColla a better sub-task allocation of each DNN layer. In general, DRL-based methods show significant advantages compared with non-DRL methods, where the Random method represents the random distribution of computing tasks, the Average method represents the average distribution of computing tasks to each IoT device, and the Least method means that tasks are prioritized to the IoT device with the highest computing resource. The Least method has better average slowdown than the Random and Average methods, but worse than DRL methods. This is because prioritizing tasks assigned to the IoT device with the highest computing resource may directly cause the IoT device to exceed the current computing capacity and cause the waiting latency of computing tasks, while DRL-based methods can provide dynamic and reasonable computing task allocation based on the status of all IoT devices. Also, Fig. 5b illustrates that DRL-ES can converge faster than the basic DRL method, and show better convergence results via the comparative analysis of the total reward. Further, we discuss the runtime performance of DRL-ES scheduling when deployed on IoT devices in Figs. 5c and 5d by using the microsecond as the unit to measure the average runtime performance of 100 task schedules. In particular, we use the CPU Usage Limiter (https://github.com/opsengine/cpulimit) to control available CPU resource to simulate various computing status. The results show that DRL-ES can execute task scheduling under different conditions in 1–2ms, which is efficient enough without increasing additional scheduling costs. This is because we only need to spend time on training DRL models offline, and the inference phase of the running trained model is real-time, even in the complex scenario with large-scale tasks. In summary, our DRL-ES scheduling strategy does provide real-time, available scheduling for the DeColla system. In addition, the DRL-based method has been widely used in intelligent decision-making in different scenarios. This is one of the main reasons that we adaot and extend the DRL method to schedule our DeColla system.

In Fig. 6 we analyze DeColla's performance against the mobile-only, the partition-offloading and hierarchical collaborative methods in the latency and the resource usage variance of IoT devices. We further illustrate that DeColla outperforms the traditional hierarchical method by simulating different running states of collaborative devices such as low available resources and abnormal disconnection. We use Wonder Shaper (https://github.com/magnific0/wondershaper) to control the communication and thus obtain various network bandwidths. We measure the latency by calculating the average latency between the smartphone's request and the receipt of the final result 10 times, excluding the image transfer and result response between the smartphone and the IoT device. The CPU usage of IoT devices is set as 30 percent in Fig. 6a. By comparing DeColla with representative methods in different network bandwidths in Fig. 6a, we observe that:

- As the network bandwidth increases, the latency of Hierarchical Colla decreases and stabilizes, while the partition-offloading method has the best latency performance with the help of the edge server.
- The overall performance of mobile-only is less affected by the network bandwidth because it is only used for one IoT device and does not require data on the interaction with other IoT devices.
- For Hierarchical Colla and our DeColla system, the increase in network bandwidth significantly lifts the data transferring rate among IoT devices, and therefore the increase in network bandwidth effectively improves latency performance.

When the network bandwidth reaches 10 Mb/s, DeColla exhibits lower latency than Hierarchical Colla, suggesting that when the network bandwidth is stable and strong, the collaborative inference approach is the key factor constraining the performance, which also suggests that parallel inference within our method exhibits better perfor-

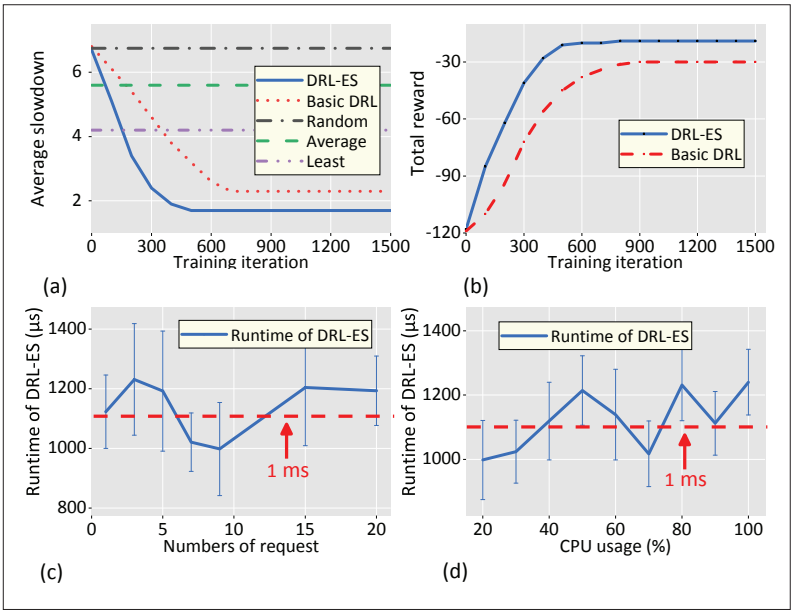

FIGURE 5. Performance of the proposed DRL-ES algorithm: a) Convergence on average slowdown; b) Convergence on total reward; c) Runtime with various request tasks; d) Runtime with various CPU usage.

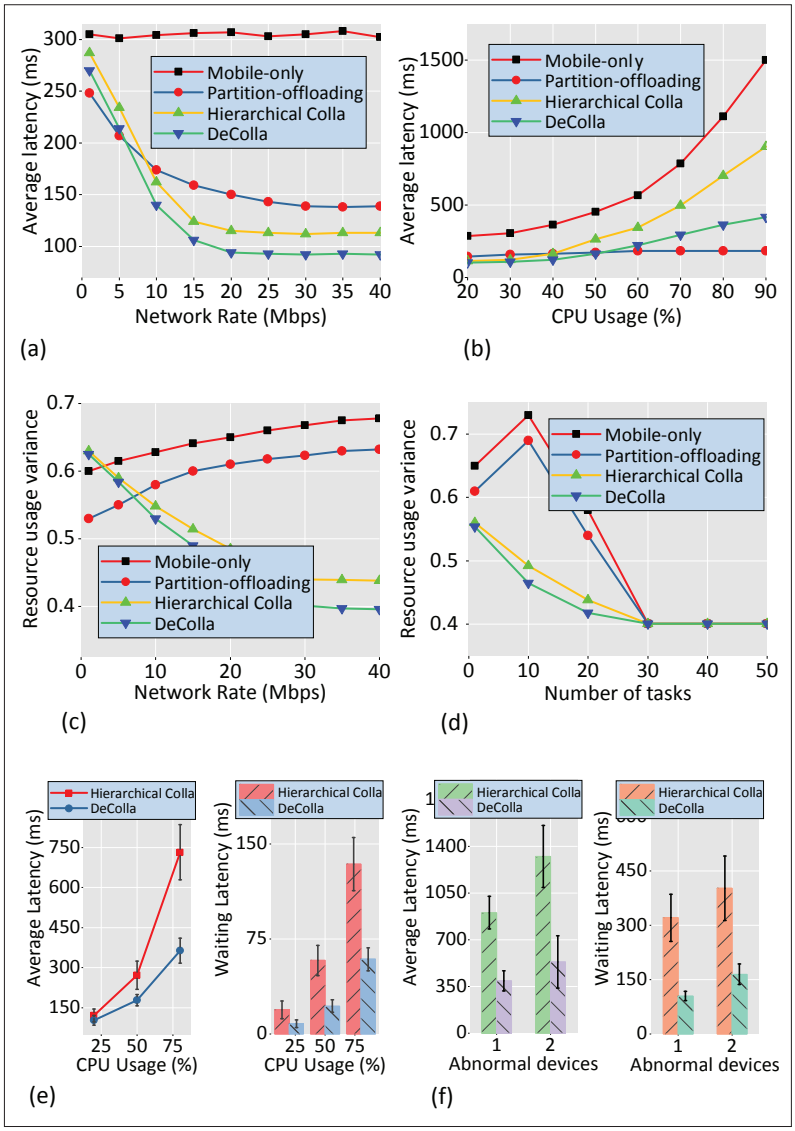

FIGURE 6. Comparing DeColla with representative methods: a) Latency performance with various networks; b) Latency performance with different CPU usages; c) Resource usage variance with different networks; d) Resource usage variance with different tasks; e) Adjusting running state of any collaborative device; f) Abnormity occurring in collaborative devices.

mance. Fig. 6b shows the comparison among different methods in various CPU usages with a stable network bandwidth of 20 Mb/s. We can see that:

- Partition-offloading is least influenced by the available computing resources of the IoT device, and as the available computing resources decrease, most DNN computations will be offloaded to the edge server.
- For the mobile-only method, since all computations are done on one IoT device, the latency to complete DNN collaborative inference increases dramatically as the IoT device's CPU usage increases.
- Compared to Hierarchical Colla, DeColla shows better latency performance when the IoT device's CPU usage is high, illustrating that DeColla's is better than that of Hierarchical Colla. This is because the Hierarchical Colla method not only spends much time on the inference of each DNN layer, increasing waiting latency between DNN layers, but also takes a lot of time for each IoT device to maintain the IP routing table.

Further, in Figs. 6c and 6d we evaluate the normalized resource utilization variance of involved IoT devices under different networks and task scales. Note that the smaller the normalized resource utilization variance, the smaller the fluctuation, representing a balanced task scheduling. We have found that:

- With the continuous increase of network bandwidth, the mobile-only method has a small increase, which shows the computing load of the processing IoT device increases. The partition-offloading method can dynamically adjust the partitioning point according to the network bandwidth, so when the network bandwidth is low, the resource utilization variance performance is low, indicating that most of the computations are executed on the edge server. Once the network bandwidth increases, a lot of computations are offloaded to one IoT device, increasing the resource utilization variance, while DRL-based methods can always provide more reasonable and dynamic allocation according to the network bandwidth and the status of IoT devices. Therefore, as the network bandwidth increases, the resource utilization variance shows a gradually decreasing trend.
- We also evaluate the resource usage performance of DeColla by simulating multiple request tasks and setting the network bandwidth as 20 Mb/s. Since mobile-only and partition-offloading always offload computing tasks to one of the IoT devices, multiple concurrent tasks cause an increase in the variance of resource utilization until one device reaches full load. Once the device reaches full load, tasks will be unloaded to other devices, so there is a certain reduction in resource utilization variance until all IoT devices are fully loaded and stabilized. In contrast, DRL-based methods effectively allocate these concurrent tasks to IoT devices for collaborative execution. Thus, the resource utilization variance appears to gradually decrease until a full load of all IoT devices reaches a stable state. In general, online scheduling using DRL improves the collaboration capabilities and resource utilization of IoT devices and effectively reduces the processing latency at the same time.

In Figs. 6e and 6f, we show the advantages of DeColla's parallel inference in each layer over the traditional hierarchical method and give a deep analysis and discussion. Figure 6e presents the performance between DeColla and the traditional hierarchical scheme by executing the same task 10 times, when controlling the CPU usage of any collaborative device from low to high but still within the available range. The results in the left part of Fig. 6e show that the average processing latency of Hierarchical Colla increases significantly due to the waiting latency of other collaborative devices when increasing the CPU usage of any collaborative device. This also means the performance of using a single device to execute a single DNN layer in Hierarchical Colla is greatly limited by the computational complexity of the current DNN layer and the running state of the device. Because

of the fluctuation of the running state of collaborative devices, it may increase the waiting latency of subsequent collaborative devices and reduce the whole inference efficiency. In contrast, DeColla can greatly improve efficiency, mainly because the fine-grained collaborative inference in each DNN layer has the essential difference when compared with Hierarchical Colla. The right part of Fig. 6e shows that the average waiting latency of different distributed inference when executing the same task 10 times. We use the history log to obtain the latency of the collaborative device waiting for the output of the current running device, and further, analyze the efficiency of these two methods. We also observe that DeColla significantly reduces the waiting latency of the participating device and improves the resource utilization of the collaborative device. Also, in Fig. 6f we experiment on the robustness by randomly disconnecting one or two collaborative devices to simulate the abnormal scenarios that collaborative devices may out of collaboration. The left part of Fig. 6f indicates the average latency of Hierarchical Colla is 2.5 times of DeColla when collaborative inference is interrupted abnormally. In addition to the above-mentioned acceleration of intra-layer parallel inference in Fig. 6e, the main reasons are:

- Once the collaborative device cannot acquire the layer input from other devices in a given response time t_1, it requires to send the reallocation for the rest of DNN layers by the task requester. While DeColla does not receive the results in a given response time t_2, it just reallocates the tasks in the current layer and does not influence other collaborative devices.
- Generally, the response time t_2 of DeColla is smaller than t_1 of Hierarchical Colla because of parallel inference of participating devices in each DNN layer, which reduces the influence and waiting latency of the abnormal device. In summary, DeColla's intra-layer parallel inference speeds up the whole collaboration and also improves the robustness of distributed collaborative inference by efficient task reallocation and adjustment of using message queue, which is more stable and reliable than the traditional Hierarchical Colla method.

Discussion

In this section, we discuss DeColla's improvements for IoT devices, the generalizability, and some limitations. First, our proposed method of decentralized and collaborative inference system based on DNN layer parallel inference and adaptive allocation strategy performs no computational dependences among IoT devices. The approach is widely applicable in different IoT domains and can improve the efficiency and quality of service via decentralized and collaborative computing over IoT devices. Second, in this work, we use image recognition as an example to illustrate the effectiveness of DeColla, which absolutely can be extended to other DNN networks, such as object detection networks and speech recognition networks. More importantly, DeColla represents the distributed and collaborative inference that is considered and designed from the perspective of stability and robustness for real scenarios. Also,

For the purpose of acquiring faster DNN inference, our parallel inference solution in the DNN layer has the characteristics of independent computation and no computation dependency between sub-tasks. It effectively improves the efficiency and service quality of collaborative DNN inference without the assistance of the cloud center.

DeColla primarily provides a novel attempt of decentralized and collaborative thinking for DNN inference. Third, the DeColla system focuses on the optimal allocation in terms of latency and the available resource of the IoT device. However, the IoT device is usually sensitive to energy consumption, and therefore needs to further provide more rational IoT device-side decentralized and collaborative DNN inference. In this work, DeColla's task scheduling and resource allocation mainly consider optimizing the inference latency, which should take mobile energy consumption into account. DeColla is also a kind of Edge-AI framework and aims to the deployment and inference of AI models, which is different from federation architectures mainly used for distributed training of AI models. As we have mentioned in the introduction, the development of AI chips and the commercial deployment of the basic infrastructure of 5G and MEC have provided promising solutions for sharing distributed AI computing capabilities across various end devices.

Conclusions

In this work, we propose a DeColla system based on a decentralized and collaborative DNN inference scheme, which is different from the traditional collaborative mechanism by hierarchical inference. For the purpose of acquiring faster DNN inference, our parallel inference solution in the DNN layer has the characteristics of independent computation and no computation dependency between sub-tasks. It effectively improves the efficiency and service quality of collaborative DNN inference without the assistance of the cloud center. In addition, our proposed DRL-based adaptive allocation strategy provides the most optimal allocation of DNN layer inference for IoT devices according to dynamic contexts. Numerical results show that DeColla can reduce the waiting latency by at least 2.5 times compared with the traditional hierarchical collaborative method when abnormally occurs during the collaborative inference. More importantly, when the collaborative device falls into an abnormal state, experimental results show that DeColla has better robustness and stability than that of hierarchical inference by using the intra-layer parallel inference and message queuing mechanism. In conclusion, the advantages of DeColla are not only to accelerate the collaborative inference of multiple devices but also to improve the stability, robustness, and fluency of service in dynamic collaboration scenarios. As future work, we plan to extend DeColla to more deep learning fields, and apply it to more complex scenarios to obtain more promising results.

Acknowledgment

This research was supported in part by the National Key R&D Program of China under Grant 2019YFF0301500; in part by the National Natural Science Foundation of China (NSFC) under Grant 61671081; in part by the Funds for Interna-

The advantages of DeColla are not only to accelerate the collaborative inference of multiple devices but also to improve the stability, robustness, and fluency of service in dynamic collaboration scenarios. As future work, we plan to extend DeColla to more deep learning fields, and apply it to more complex scenarios to obtain more promising results.

tional Cooperation and Exchange of NSFC under Grant 61720106007; in part by the 111 Project under Grant B18008; in part by the Fundamental Research Funds for the Central Universities under Grant 2018XKJC01; and in part by the BUPT Excellent Ph.D. Students Foundation under Grant CX2019135.

References

[1] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network*, vol. 32, no. 1, 2018, pp. 96–101.
[2] P. Ren *et al.*, "Edge-Assisted Distributed DNN Collaborative Computing Approach for Mobile Web Augmented Reality in 5G Networks," *IEEE Network*, vol. 34, no. 2, 2020, pp. 254–61.
[3] X. Qiao *et al.*, "Web AR: A Promising Future for Mobile Augmented Reality — State of the Art, Challenges, and Insights," *Proc. IEEE*, vol. 107, no. 4, 2019, pp. 651–66.
[4] Z. Fang, D. Hong, and R. K. Gupta, "Serving Deep Neural Networks at the Cloud Edge for Vision Applications on Mobile Platforms," *Proc. 10th ACM Multimedia Systems Conf.*, 2019, pp. 36–47.
[5] J. Mao *et al.*, "Mobieye: An Efficient Cloud-Based Video Detection System for Real-Time Mobile Applications," *Proc. 56th Annual Design Automation Conf. 2019*, 2019, pp. 1–6.
[6] Y. Kang *et al.*, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, 2017, pp. 615–29.
[7] Y. Huang *et al.*, "A Lightweight Collaborative Deep Neural Network for the Mobile Web in Edge Cloud," *IEEE Trans. Mobile Computing*, Jan. 2020.
[8] Y. Huang *et al.*, "Deepadapter: A Collaborative Deep Learning Framework for the Mobile Web Using Context-Aware Network Pruning," *IEEE Conf. Computer Commun. (INFOCOM)*, IEEE, 2020, pp. 834–43.
[9] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed Deep Neural Networks over the Cloud, the Edge and End Devices," *Proc. 2017 IEEE 37th Int'l. Conf. Distributed Computing Systems (ICDCS)*, IEEE, 2017, pp. 328–39.
[10] M. Sandler *et al.*, "Mobilenetv2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, IEEE, 2018, pp. 4510–20.
[11] R. Hadidi *et al.*, "Musical Chair: Efficient Real-Time Recognition Using Collaborative IoT Devices," arXiv preprint arXiv:1802.02138, 2018.
[12] R. Hadidi *et al.*, "Towards Collaborative Inferencing of Deep Neural Networks on Internet of Things Devices," *IEEE Internet of Things J.*, 2020.
[13] H. Mao *et al.*, "Resource Management with Deep Reinforcement Learning," *Proc. 15th ACM Wksp. Hot Topics in Networks*, ACM, 2016, pp. 50–56.
[14] T. Salimans *et al.*, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," arXiv preprint arXiv:1703.03864, 2017.
[15] F. P. Such *et al.*, "Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning," arXiv preprint arXiv:1712.06567, 2017.

Biographies

Yakun Huang received his Ph.D. degree in computer science from Beijing University of Posts and Telecommunications in 2021. His current research interests include video streaming, mobile computing, edge computing, and augmented reality.

Xiuquan Qiao is currently a full professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests include the future Internet, services computing, computer vision, distributed deep learning, augmented reality, virtual reality, and 5G networks.

Schahram Dustdar [F] was an Honorary Professor of Information Systems in the Department of Computing Science, University of Groningen, Groningen, The Netherlands, from 2004 to 2010. He is currently a professor of computer science with the Distributed Systems Group, Technische Universität Wien, Vienna, Austria. He was an elected member of the Academy of Europe, where he is the Chairman of the Informatics Section.

Jianwei Zhang is currently the director of the Institute of Capinfo Company Limited. His research interests include cloud computing, big data, and AI. He is dedicating himself to the research of smart stadium of the national speed skating oval which is funded by the National Key R&D Program of China under grant 2019YFF0301500.

Jiulin Li is currently the deputy chief engineer of Beijing Urban Construction Group Company Limited. He is currently participating in the research of smart stadium of the national speed skating oval which is funded by National Key R&D Program of China under grant 2019YFF0301500 as a senior consultant.