Scenario-Based AI Benchmark Evaluation of Distributed Cloud/Edge Computing Systems

Tianshu Hao, Kai Hwang, *Life Fellow, IEEE*, Jianfeng Zhan, Yuejin Li, and Yong Cao

Abstract—Distributed cloud/edge (DCE) platform has become popular in recent years. This paper proposes a new AI benchmark suite for assessing the performance of DCE platforms in machine learning (ML) and cognitive science applications. The benchmark suite is custom-designed to satisfy scenario-based performance requirements, namely the model training time, inference speed, model accuracy, job response time, quality of service, and system reliability. These metrics are substantiated by intensive experiments with real-life AI workloads. Our work is specially tailored for supporting massive AI multitasking across distributed resources in the networking environment. Our benchmark experiments were conducted on an AI-oriented AIRS cloud built at the Chinese University of Hong Kong, Shenzhen. We have tested a large number of ML/DL programs to narrow down the inclusion of ten representative AI kernel codes in the benchmark suite. Our benchmark results reveal the advantages of using the DCE systems cost-effectively in smart cities, healthcare, community surveillance, and transportation services. Our technical contributions are in the AIRS cloud architecture, benchmark design, testing, and distributed AI computing requirements. Our work will benefit computer system designers and AI application developers on clouds, edge, and mobile devices, that are supported by 5G mobile networks and AIoT resources.

Index Terms—Computer benchmarks, cloud/edge computing, machine learning, and artificial intelligence

1 Introduction

A *distributed cloud/edge* (DCE) computing system is hierarchically structured with workloads widely dispatched over a large network environment [1], [2], [3]. The DCE system resources extend from cloud/datacenters to edge servers and user-end devices. Distributed platform alleviates the problems of limited bandwidth, offloading latency, and data storage often found in centralized platforms [16], [18], [29].

Fig. 1a shows a typical DCE system built with three layers of resources: the cloud clusters, edge servers, and end devices. Using heterogeneous servers and devices makes it difficult to apply traditional benchmarks. We designed the Edge AIBench to include some scenario-based kernel functions in real-life AI applications such as remote healthcare, community surveillance, smart city, and intelligent transportation.

ML/DL models are most trained at the cloud datacenter due to the large datasets involved. However, the pre-trained ML models could run on the edge servers. The AI inference is mostly carried out at edge servers. Data sensing and collection are done at end devices, as illustrated in Fig. 1b. Our key ideas and original contributions are highlighted below:

- We identify the major system requirements of DCE computing platforms. Dynamic resource allocation is suggested to upgrade system performance in massive AI applications.
- A weighted performance model is developed for DCE systems. This model can be applied to any set of AI performance metrics in orthogonal dimensions.
- We propose a scenario-based benchmark suite named *Edge AIBench* based on four representative AI service scenarios often encountered in our daily-life applications.
- We execute the Edge AIBench [13] programs on an AI-oriented AIRS cloud/edge testbed. The innovative features of this system are first-time reported to the general public inters of hardware architecture and benchmarking environments.
- The relative performance of five existing AI benchmark suites is compared. They are complementary in nature to offset the limitations associated with each individual suite.
- We evaluate the DEC system with six key QoS metrics to reveal the effects of model training, inference speed, AI classifier accuracy, task response time, and system reliability.

In the remaining sections, Section 2 presents five existing AI benchmark suites. Section 3 presents the AIRS cloud architecture and features used in our benchmark experiments. Edge AIBench is specified in Section 4 along with AI workload, datasets, timing assumptions, and AI/ML inference engines. Section 5 is devoted to performance metrics

- *Tianshu Hao and Jianfeng Zhan are with the Research Center for Advanced Computer Systems, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100045, China, and also with the University of Chinese Academy of Sciences, Beijing 101408, China. E-mail: {haotianshu, zhanjianfeng}@ict.ac.cn.*
- *Kai Hwang and Yuejin Li are with the Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China. E-mail: hwangkai@cuhk.edu.cn, yuejinli@link.cuhk.edu.cn.*
- *Yong Cao is with the Huazhong University of Science and Technology, Wuhan, Hubei 12443, China. E-mail: yongcao@hust.edu.cn.*

Manuscript received 23 June 2021; revised 29 Apr. 2022; accepted 10 May 2022. Date of publication 23 May 2022; date of current version 10 Feb. 2023.
This work was supported in part by the Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS) under Grant AC01202101114. The project was also jointly supported in part by the Institute of Computing Technology, Chinese Academy of Sciences, Beijing.
(Corresponding author: Kai Hwang.)
Recommended for acceptance by B. Childers.
Digital Object Identifier no. 10.1109/TC.2022.3176803

0018-9340 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

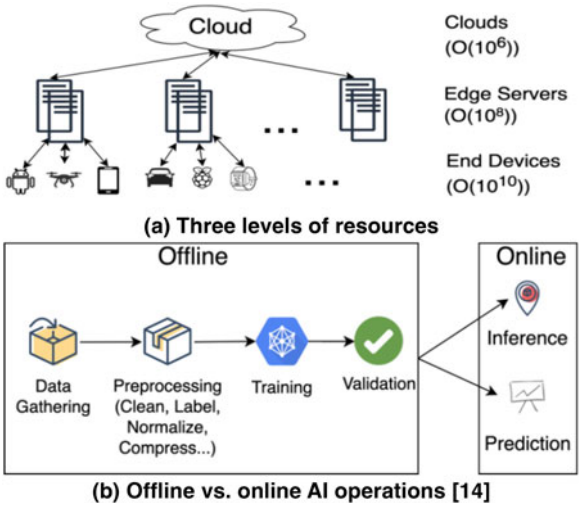

Fig. 1. The architecture and major AI tasks running on a typical distributed cloud/edge (DCE) computing system.

established and the quality of services (QoS) requirements. Sections 6 and 7 present original benchmark results. Finally, we summarize our scientific discoveries and discuss further research. This journal version has 90% new material, very different from our preliminary results reported in [13], [14].

2 AI Benchmarks and Application Scenarios

Considering industrial and academic requirements, AI tasks are encountered in many cloud/edge computing scenarios [11], [32]. Good examples include natural language processing, image understanding, face and speech recognition, etc. Therefore, essential AI applications must be included in our Edge AIBench development.

Fig. 2 shows four representative scenarios we have investigated for evaluating typical AI execution in a cloud/edge computing environment. Table 1 summarizes four existing AI benchmark suites reported in the open literature. We briefly introduce them here to set up the background.

MLPerf [22], [27] was jointly developed by Google, Baidu, Alibaba, and many other organizations. The suite focuses on measuring Machine Learning (ML) performance. *MLPerf* suite considers how fast an AI system can perform inference using a pretrained ML model [27]. The EEMBC develops the *MLMark* to measure the accuracy of embedded inference operations.

The *DeFog* [23] is a fog computing benchmark suite developed by Queen's University and Cisco Systems. It collects four AI and two non-AI tasks. They consider three performance metric types, for fog computing platforms, exclusively. *AIPerf* is an ML benchmark for general-purpose, high-performance cloud platforms [28].

Our Edge AIBench suite was put together by Institute of Computing Technology, Chinese Academy of Sciences, and later adopted by the International Open Benchmark Council (BenchCouncil) [3], [35]. This suite has collected ten representative AI application programs in four real-life AI scenarios running on the DCE platform.

Our Edge AIBench suite differs from the other four AI benchmark suites in three technical aspects.

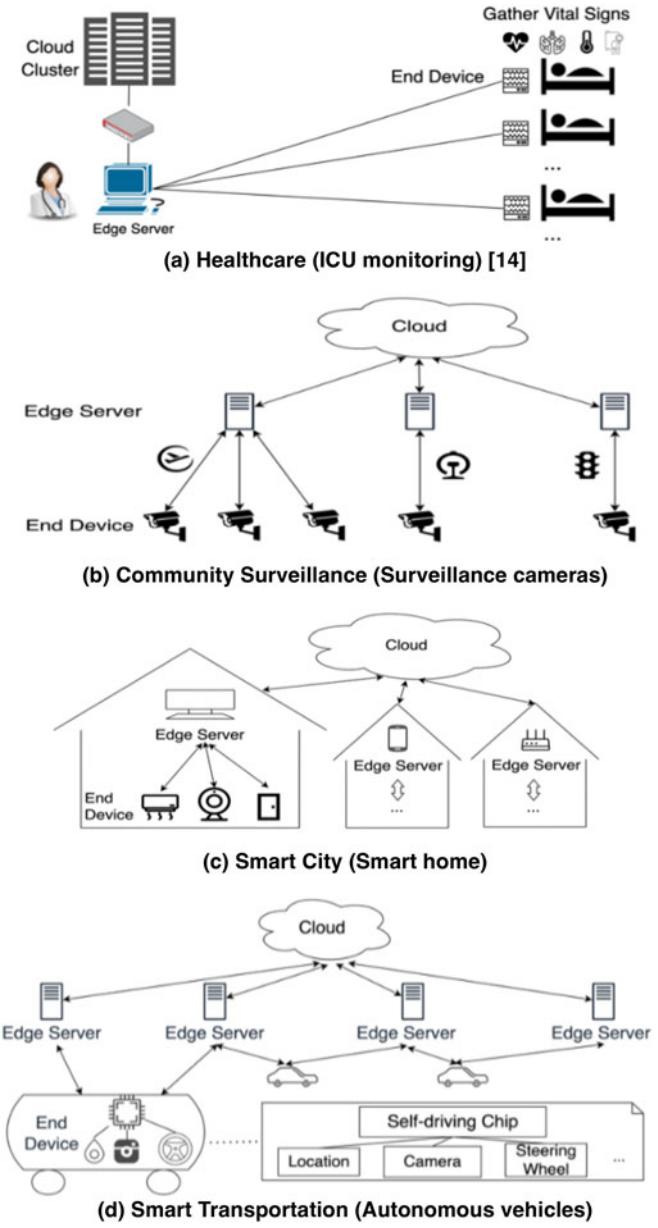

Fig. 2. Four representative AI application scenarios in real-life distributed cloud/edge computing services.

1) We emphasize the evaluation of the hierarchically structured DCE platform. Most other suites are designed for running on centralized clouds or high-performance datacenters.
2) Edge computing for AI is what our benchmark can serve the community better in terms of specifically selected performance metrics to match with major application scenarios. The DeFog benchmark covers mainly the edge services, excluding the performance concerns at the cloud or user-end levels.
3) We aim to balance the workload at three levels to meet the QoS demand by common AI users. Our Edge AIBench is scenario-based. We design the suite for massive AI multitasking on the DCE framework covering cloud, edge, and end devices.

In summary, our Edge AIBench differs from the other four benchmark suites. The *AIPerf* is mainly designed for HPC platforms. *MLMark* appeals more to measure embedded

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Existing AI Benchmark Suites and Their Applicable Platforms

Benchmark Suite	Developers and Key Adopters by 2022	Platforms applied and Key References
MLPerf	Google, Baidu, Alibaba, Facebook, etc	An industrial benchmark for ML [6], [22], [27]
MLMark	Embedded Microprocessor Benchmark (EEMBC)	Speed and accuracy of embedded ML [10]
DeFog	Queen's University and Cisco	A fog computing benchmark for general-purpose Apps [23]
AIPerf	Tsinghua University and Chinese Academy of Science, Institute of Computing Technology	An HPC benchmark for testing AI performance in ML operations [28]
Edge AIBench	Jointly developed and tested by BenchCouncil and the CUHK-Shenzhen	Performance metrics for ML/DL reported here extending from [3], [12], [13]

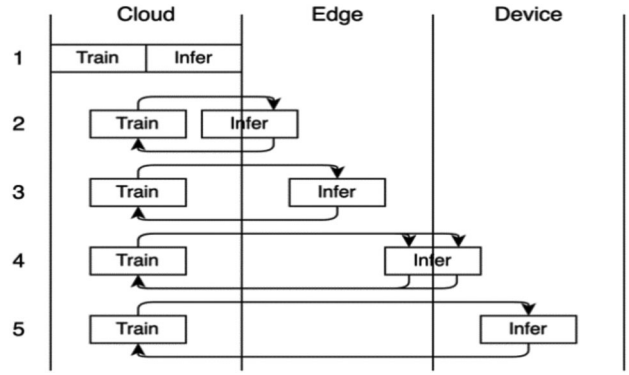

Fig. 3. AI model training in the cloud and inference operations carried out at three cloud/edge/device levels.

applications. The *MLPerf* is designed for industrial settings. Our Edge AIBench is more suitable for studying weighted performance driven by distributed application scenarios.

2.1 Critical Path in AI Computing

DCE computing is deployed on many resources among the cloud, edge, and end devices. The collaborations and communications among all resource nodes affect the overall performance. Fig. 3 shows how the ML training and inference process are carried out at cloud-edge-device resource levels.

The AI performance is influenced by a variety of workloads assigned in three-layer distributed computing architecture [14]. Some end devices or edge servers are limited to performing model training operations. Strong collaborations or workload offloading are needed among the cloud, edge, and end devices.

A comprehensive benchmark suite for cloud/edge computing must consider end-to-end along the critical path. We assume AI model training is mostly done at the cloud level. The ML inference could be done at all three levels, but most likely done at the edge servers as seen in cases 2, 3, and 4 in Fig. 3. Case 5 shows the inference at the end devices such as smart phones or camera surveillance.

2.2 AI Performance and Requirements

Several key performance concerns of the AI benchmark suite are elaborated below for general-purpose distributed cloud, edge, and device computing systems.

a) *High-accuracy:* Various AI tasks impose different accuracy levels. For computer vision, we wish to match with human accuracy in face recognition. Autonomous vehicles and emergency room services demand higher accuracy due to life-risk concerns. On the other hand, with smart home sensors, we may be able to tolerate some ML errors.

b) *Latency Tolerance:* Many latency problems come from communication delays or interrupts. Some DCE applications may tolerate time delays in responses. However, critical real-time response is desired in autonomous vehicles [7].

c) *Device Mobility:* Mobile means the end devices are not static in one place. Mobile Edge Computing (MEC) is a crucial technology and architectural concept in edge computing [26]. In our four scenarios, smart transportation is a typical scenario using mobile edge computing technology. In a MEC scenario, stable network connection and state transition are two fundamental problems that must be solved.

d) *Massive Users:* Usually, one edge server needs to correspond to multiple users. In ICU monitoring, one edge server corresponds to multiple patients; In the surveillance camera scenario, one edge server corresponds to multiple cameras; In smart home, one edge server corresponds to multiple devices. However, due to the particularity of the autonomous vehicle, one AI chip only serves one smart car.

e) *Big Datasets:* As IHS Markit reported, one billion surveillance cameras will be installed globally in 2021 [8]. Moreover, it has been calculated that a 24-hour 4K 60FPS video takes nearly 35TB storage. Therefore, there are enormous data in community surveillance compared to any edge computing scenario.

f) *Resources Diversity:* In DCE computing scenarios such as smart home, there are multiple heterogeneous devices, including smartphones, computers, routers, cameras, floor cleaning robots, watches, clocks, windows, doors, curtains, lamps, etc. Those devices generate heterogeneous data: text, image, audio, video, etc.

3 AIRS: The Testbed for DCE Computing

Our work evaluates general DCE computing platforms. The AIRS cloud offers a typical example of a DCE platform, which is specifically designed for AI-oriented cloud/edge computing. This AIRS cloud design contains a large number of creative features for DCE computing. The hardware and software architecture of this system are presented below for the first time. For detailed architecture features, readers can refer to the AIRS website [4].

3.1 AIRS Cloud/Edge System Architecture

Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS) has built a DCE system on the campus of Chinese University of Hong Kong, Shenzhen (CUHK-SZ). This is a large-scale AI-oriented cloud/edge system (Fig. 4) as a public cloud used by all research communities in the Greater Bay Area of the Pearl River.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

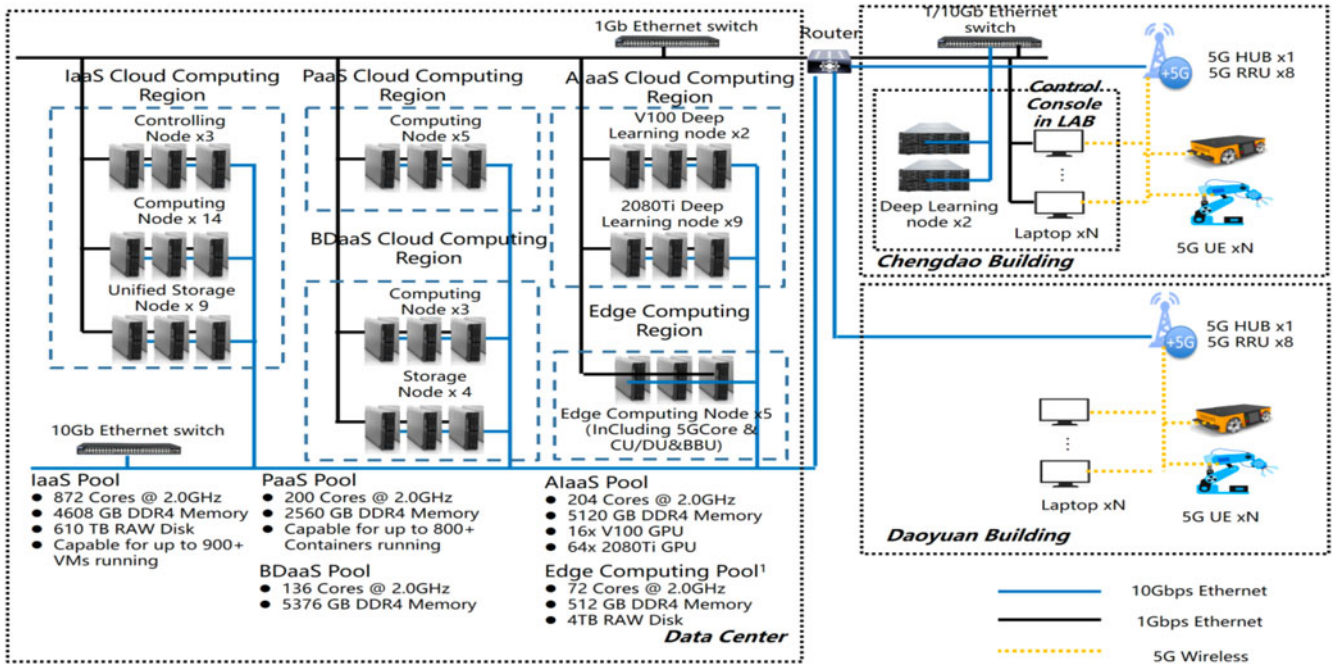

Fig. 4. AIRS cloud was built at CUHK-Shenzhen to carry out all AI benchmark experiments reported herein.

AIRS Cloud is designed with a three-level architecture: Cloud-Edge-End. The central cloud supports high performance computing, big data analysis, and deep learning with 1520 physical CPU cores, 18.7 TB memory, 72 2080Ti GPUs, and 16 V100 GPUs. The edge computing platform is constructed jointly with a 5G research network.

The system supports experiments on various AI-oriented applications. We aim to upgrade the IoT environment to promote the digital economy. The *intercloud management system* (ICM) handles all resources across multiple server clusters.

Fig. 4 shows the major hardware configuration in the AIRS cloud. This cloud is built with five server clusters, which incorporate with the IaaS cluster, PaaS cluster, big-data storage, AI engine, and edge platform as mutually supportive resources regions.

For general AI applications, their computing and reasoning services are deployed on *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and edge cloud platforms. The data collection, storage, and analysis work are deployed on the *Big Data as a Service* (BDaaS) cloud platform for big data analysis. The model file training involving deep learning is deployed on the *AI as a Service* (AIaaS) platform for artificial intelligence.

The whole distributed system is interconnected with the 5G special network. Compared with a 5G public network, the core network function and configuration of this 5G special network are adjusted according to the requirements of different development scenarios.

The 5G special network is upgraded with the *network function virtualization* (NFV) technique. This could make the distributed DCE system even more appealing to building some industrial Internet slices. This will enable the deployment of special AI or IoT applications on cloud/edge systems even closer to satisfying user demands in the 5G era.

3.2 AIRS Cloud Software Stack and Services

AIRS cloud supports five categories of mutually supportive cloud/edge services as illustrated in Fig. 5.

- *Infrastructure as a Service* (IaaS): Resource pool with OpenStack+KVM+Ceph commercial distributions.
- *Platform as a Service* (PaaS): Resource pool with Kubernetes+Docker commercial distributions.
- *Edge Computing platform*: Resource pool with Kubernetes + Docker +KVM in the 3GPP's framework.
- *Big Data as a Service* (BDaaS): has a Hadoop software stack, such as HDFS, YARN, and MapReduce engine. Spark, etc.
- *AI as a Service* (AIaaS): supports Docker's containers for DL with PyTorch, TensorFlow, Caffe, and AutoML.

Cloud-edge coherence control is of central importance here. For example, the software management stacks of OpenStack, Kubernetes, Hadoop, and various mobile and AI service libraries need to achieve a high degree of resource sharing and load balancing in both HPC and AI workloads. More details of these software environments of AIRS cloud and its special software support for federated machine learning applications can be found in [4].

4 Programs Created in Edge AIBench

AI programs depend on the workload and dataset used in model training operations. This section introduces ten AI programs that we have created and implemented in the Edge Bench framework for testing DCE systems in general.

4.1 Edge AIBench Execution Steps

To cover four representative DCE scenarios, we proposed 10 AI application programs. These programs are considered the kernel functions performed in general AI tasks. This new benchmark provides a reference AI model, open-source

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

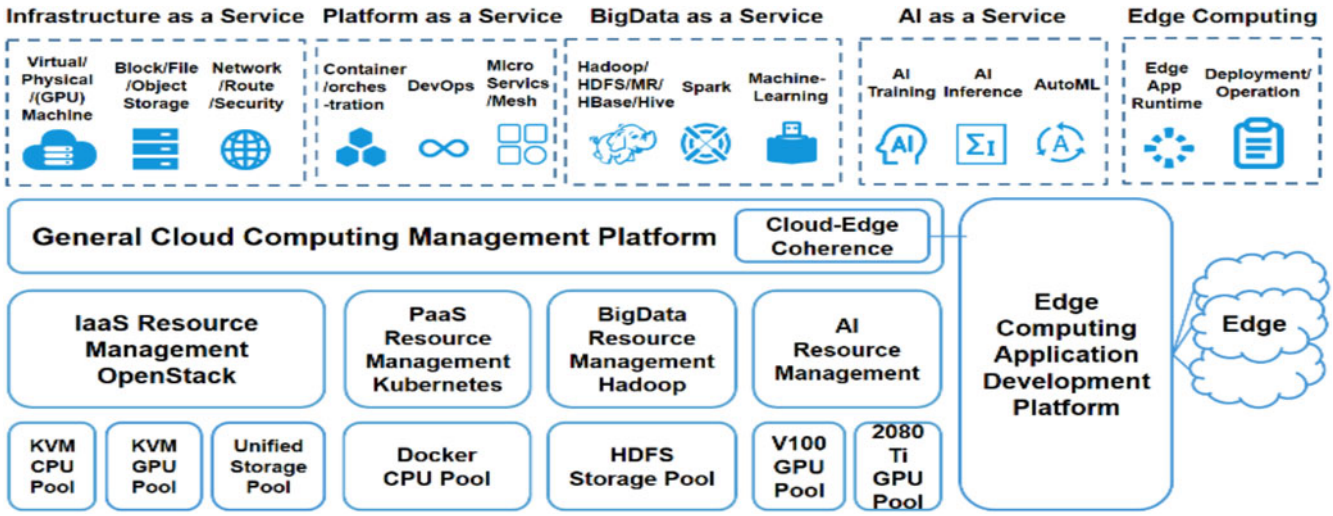

Fig. 5. Resource pools and benchmark programming environment in the AIRS cloud platform.

datasets, and evaluation metrics. Fig. 6 shows five development stages of the Edge AIBench by BenchCouncil [3].

The Edge AIBench was fully developed in the lead author's Ph.D. thesis [12]. The preliminary version of a few benchmark programs was reported in a conference paper [13]. However, the suite keeps changing with the addition of more performance metrics tested on real computer systems. We investigate special DCE AI computing requirements for four typical scenarios as introduced in Fig. 2.

As shown in Fig. 7, this new benchmark suite consists of the data input, micro benchmarks, typical AI program calls, end-to-end communication, and metrics measurements. Micro benchmarks are included to support offline training and online cross-validation. Using this benchmark development model., users can build their own benchmark suite, specially addressing the concerns in other AI application scenarios.

We show how each benchmark program is built with an end-to-end kernel and micro benchmarks. The Edge AIBench suite collects a wide range of data types, including structured, semi-structured, and unstructured data. Most of the data are from open-source datasets.

The datasets are fed as input to the micro benchmarks. During offline training, the micro benchmarks complete training and transmit the trained models to the online inference components. Online inference benchmarks use the trained model to conduct testing and inference. Different scenarios may require some special component programs. We also provide communication components for data transmitting among different layers of DCE systems. We construct the end-to-end benchmarks, which was not done in other AI benchmarks listed in Table 1.

4.2 AI Kernel Operations in Edge AIBench

The Edge AIBench has implemented ten programs using our framework, as listed in Table 2. Additional details can be found on the BenchCouncil website [3]. This benchmark suite is subject to future extensions if new domains are explored.

P1: Traffic monitoring is a kernel program for autonomous vehicle operations. This program enables autonomous driving along proper road lanes. Traffic monitoring must be accurately carried out in real-time. Thus, this program is sensitive to high accuracy and low latency in making steering decisions.

P2: Traffic sign detection is crucial to avoid an accident in maintaining safe autonomous vehicle driving. The correct sign detection may be affected by the brightness of the natural environment and other vehicles' speed. A wrong recognition of the road sign may trigger severe traffic disasters.

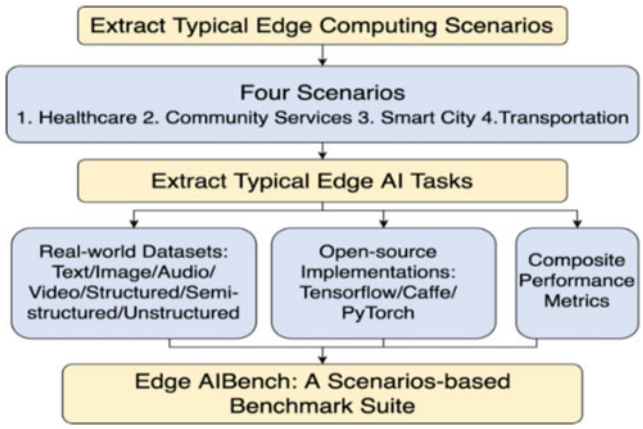

Fig. 6. Edge AIBench development by BenchCouncil [3], [13].

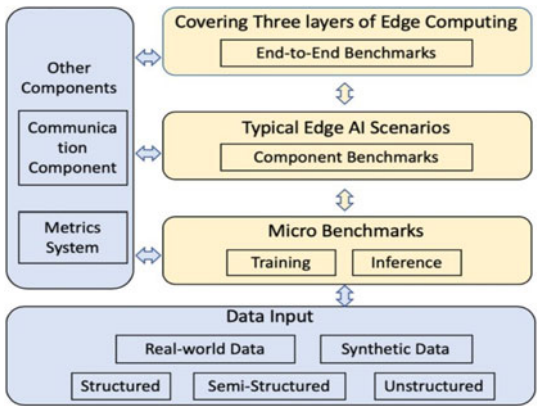

Fig. 7. Edge AIBench framework and execution steps.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Some Kernel Programs Implemented in Edge AIBench Up to 2022

Program Name	DCE Scenarios	Datasets Applied	Implement
P1. Traffic monitoring	Smart car transportation	Tusimple [5]/ CULane [24]	PyTorch/ Caffe
P2. Traffic sign detection	Smart car transportation	Traffic Sign Recognition [31]	Keras
P3. Heartattach alert	Healthcare (ICU monitoring)	MIMIC-III [21]	Tensorflow/ Keras
P4. Decompensation	Healthcare (ICU monitoring)	MIMIC-III	Tensorflow/ Keras
P5. Phenotype classification	Healthcare (ICU monitoring)	MIMIC-III	Tensorflow/ Keras
P6. Person reidentification	Community Surveillance	Market 1501 [36]	PyTorch
P7. Event discovery	Community Surveillance	UCF101 [30]	PyTorch/ Caffe
P8. Face recognition	Smart city (Smart home)	LFW [17]	TensorFlow/ Caffe
P9. Speech recognition	Smart city (Smart home)	LibriSpeech [25]	TensorFlow
P10. Image classification	For DCE scenarios	ImageNet [9]	PyTorch

P3: Heartattack alert must be made during the first 48 hours in ICU care of patients. It uses a two-layer LSTM neural network to make a binary classification [15]. It is a standard model dealing with sequential data and performs well in medical machine learning. The input is diastolic blood pressure, fraction inspired oxygen, etc.

P4. Patient decomposition prediction is based on the first hour information at ICU to predict the patient's treatment in the next 24 hours. It uses the LSTM to make a binary classification [15]. The input is physiological information of the current hour, and the output is the probability of whether a patient will suffer decomposition in the next 24 hours.

P5. Phenotype classification aims to classify 25 ICU patients' conditions. It uses LSTM to do 25 separate binary classification tasks [15]. The input is physiological information of a full record, and the output is 25 conditions such as sepsis and diabetes.

P6. Person re-identification aims to establish identity across different cameras [33]. We retrieve target images in surveillance to locate a person and record movements.

P7. Event discovery is used to recognize special events and record abnormal behaviors in video surveillance. The tasks need to finish detecting within a short time window.

P8. Face Recognition detects a human's face in a video and is commonly used in smart home scenarios to verify authentic users. It needs a high accuracy to ensure security.

P9. Speech recognition is a natural language processing program to recognize voice commands in smart home management.

P10. Image Classification aims to identify the features of an image and classify it. It's a kernel program that must be used in most DCE AI scenarios. It's a multi-class classification task.

5 AI and ML Performance Metrics

We have surveyed 300 AIRS Cloud users. We summarized what factors they are most concerned about in using our platform. This leads to the six benchmark performance metrics. Among these metrics, ML model training, cross validation accuracy, and inference latency are considered most relevant to AI/ML performance. The total job response time and quality of service are more relevant to distributed network latency and bandwidth issues. The reliability issue is related to the availability and security of the DCE system.

TABLE 3
Weighted Demand Applications of Six Performance Metrics in Four Scenarios

Performance Metrics	Healthcare Services	Community Services	Smart City	Transportation
Accuracy (A)	★★★★★	★★★★	★★★	★★★★★
Inference (I)	★★★★	★★★	★★	★★★★★
Training (T)	★★	★★★★★	★★	★★★★
Response (R)	★★★★★	★★★★	★★	★★★★
QoS (Q)	★★★	★★	★★★★★	★★★★
Reliability (Re)	★★★★	★★★★	★★★★	★★★

5.1 Special Metrics for AI Benchmarks

As shown in Table 3, the mark from one star ★ to five ★★★★★ represents the increasing demands of each AI application scenario. More stars refer to higher demand for that performance metric under a specific AI application. For instance, healthcare has the highest demands on Accuracy (A) and Response (R).

Table 3 shows the weights for a healthcare scenario as {5/23, 4/23, 2/23, 5/23, 3/23, 4/23} for six metrics, where the common denominator 23 accounts for the total number of stars under the healthcare column in Table 3. Different scenarios may apply different weights. The intelligent transportation system at the rightmost column shows the highest demands.

The metric weight distributions are determined based on users' practice experience, specialist opinion, and long-term surveys on the state-of-the-practice AI applications. The table star entries are based on user-specific requirements. These weights can be modified by different user groups in their specific application scenarios.

5.2 Effects of Network Latency

In a distributed computing environment, the major hurdle is the end-to-end network latency experienced. A large portion of data transmission lies in waiting for remote servers to respond. Fig. 8 shows two patterns for dataflow and control flow in ML training and logic inference locations [14].

Fig. 8a shows the ill effects of skipping the edge servers, resulting in longer network latency in data transmission or control commands [34]. Fig. 8b is a more popular configuration by involving both cloud and edge resources, resulting in higher throughput, cost-effectiveness, and faster in reducing response time to end users.

Major factors in determining the latency of the model training and logic inference process include the computing power at each level, the complexity of the model, the size of the inference data, and the network latency. In addition, the data are gathered at the end devices. Therefore, data transmission delays may come primarily from data streams flowing between edge servers and the cloud datacenters.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

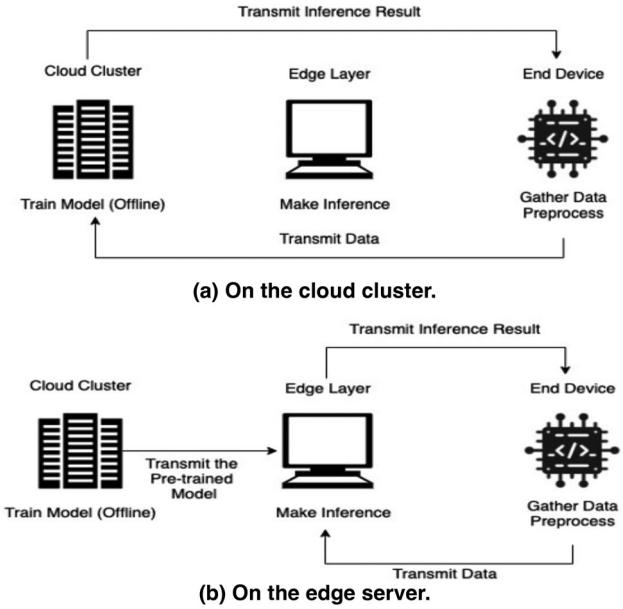

Fig. 8. Dataflow and model execution differences at cloud clusters and edge servers for AI/ML operations [14].

In the DCE environment, the deep learning model is first pre-trained on the central cloud and then deployed on the edge server. Thus, the AI training speed on the central cloud, AI accuracy, and inference speed on the edge server are three fundamental performance measures.

Our performance model can be applied to any selected set of performance metrics. Those metrics used here match batter to satisfy the 4 AI application scenarios introduced in Fig. 2. For other scenarios, one can certainly choose other metrics to satisfy particular user concerns. In other words, the AI performance frame is for general-purpose AI applications.

A. AI Model Training Time

The training workload of AI is compute-intensive. For example, the training process updates neural network parameters iteratively based on massive input image data arrays. This process involves an enormous amount of matrix multiplications. Therefore, training speed is an effective metric to measure the capability of the AI platform. The platform can be accelerated by using some GPUs or AI chips.

We need to calculate the workload by estimating the number of floating-point operations required for processing each image data array. This training time involves three phases: a forward pass (forward flops, denoted as α), backward pass, and update pass, where each pass includes multiply-add operations

Therefore, the total number of floating-point operations γ for the training process is estimated by:

$$\gamma = 3 \times \alpha \times W \times N \tag{1}$$

where $3 \times \boldsymbol{\alpha}$ accounts for the FLOPs in forward, backward, and update passes. OpenAI [1] assumes that the numbers of FLOPs for these three passes are equal. W is the workload or the total number of image data arrays processed. N is the number of iterations in using a neural network.

Let $\boldsymbol{\beta}$ be the cloud computing speed denoted by FLOPS (floating-point operations per second). This speed is determined by the number and type of GPU applied. The training time T for the neural network is calculated as:

$$Training\ Time\ (T) = \gamma/\beta \tag{2}$$

B. Model Accuracy in ML/DL Process

Model accuracy refers to the correct prediction or classification percentage using a trained machine learning or neural network model. Let TP, FP, TN, and FN be the model classification of true positive, false positive, true negative, and false negative, respectively.

TP refers to a positive case correctly predicted. TN refers to a negative case correctly predicted. FP refers to a positive case correctly predicted. FN refers to a negative case incorrectly predicted. Then the accuracy can be defined as:

$$Model\ Accuracy\ (A) = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

The higher the value of A, the better the classification accuracy of this deep learning model.

C. AI Task Inference Speed

The pretrained deep learning model is deployed on an edge server to infer new inputs to make predictions. Given an input image array, this input will be processed by a forward pass to generate the prediction result. Therefore, the number of floating-point operations for inference of each image data array is equal to α defined in Eq. (1).

Let $\boldsymbol{\delta}$ be the FLOPS speed of the edge server, and the inference time I for one input image data array is calculated by:

$$Inference\ Time\ (I) = \alpha/\delta \tag{4}$$

The larger the I, the longer the inference time of this deep learning model on the edge platform. Note that training time in Eq. (1) and inference latency in Eq. (4) are very different. In general, the model training time is much longer than that of the inference on a single image array.

5.3 AI Service Quality and Reliability

The pre-trained AI model is often deployed at the edge server to provide AI services for users. Thus, it is also necessary to select user request response time and quality of services as two service quality metrics for users. We choose a system reliability metric to assess the quality of the entire system.

D. AI Job Response Time

The total job response time needs to include data uploading time, inference time at the edge, and the returning time needed. Table 5 summarizes three AI service and reliability metrics. The AI task inference time at the edge server is estimated by Eq. (4). The prediction results returning time is short that can be ignored due to sufficient downlink resources. The uploading transmission time is related to the available bandwidth.

Let D be the size of the user's testing data set, B be the network bandwidth, and F be the inference time on an edge server. Since 1 byte has 8 bits, we have 1 Mbps = 1 Mb/s = 1/8 MB/s The transmission time is equal to $8D/B$. Thus, the total system response time is estimated by:

$$Response\ Time\ (R) = 8D/B + I \tag{5}$$

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
Three Key AI Performance Metrics

Metric	Notation and Definition	Eq. No.
Training Time on the cloud	α = Number of floating-point operations (FLOPs) per each image array in one epoch pass	Section 5.3
	β = Floating-point operations per second of GPU speed (FLOPS) of the cloud platform	
	γ = Total number of floating-point operations in the neural network training process	Eq. (1)
	T = Training Time of a neural network model	Eq. (2)
Model Accuracy	A = The percentage of correct prediction or classification in using a trained machine learning or neural network model	Eq. (3)
Inference Time on an edge server	δ = Floating-point operation per second of the GPU speed (FLOPS) at the edge server	Eq. (4)
	I = Inference time of 1 input image data array on an edge server	

TABLE 5
Performance Metrics for AI Service Quality and Reliability

Metrics	Notation and Definition	Eq. No.
System Reliability	Re = The percentage of time the system is available to deliver stable work. (%)	Eq. (8)
Job ResponseTime	D = User's testing dataset size (GB). B = Network bandwidth (Gbps) R = Total response time	Eq. (5)
Quality of Service	G = Time gain in using the DEC platform over the use of local servers or devices η = A penalty factor in using edge resources Q = Quality of Service (%)	Eq. (7)

E. Quality of Services

We define QoS as the degree of users' satisfaction per service with the response time. Let ε be the FLOPS speed of the local user device. It is desired to upload the data to the edge server only if the response time R is smaller than the inference time in a local device. The time gain of a DEC platform is estimated by:

$$G = \frac{\left(\frac{\alpha}{\varepsilon} - R\right)}{\left(\frac{\alpha}{\varepsilon}\right)} = 1 - \frac{\varepsilon R}{\alpha} \tag{6}$$

In Eq. (6), we have assumed that $R < \frac{\alpha}{\varepsilon}$, meaning we offload the inference job to the edge server only if it is faster than the local device speed at the user end. Let $\boldsymbol{\eta}$ be the penalty factor in using an edge server, which is a fraction number.

The value of $\boldsymbol{\eta}$ is determined by δ, the FLOPS of the edge server. The higher is the computing power of an edge server, the higher is the penalty. However, the lower is the response time R, the higher is the value of G. Thus, we define QoS by $=$

$$Quality\ of\ Services\ (Q) = (1 - \eta) \times G \tag{7}$$

Essentially, the QoS offers a tradeoff metric between the penalty and computing power at an edge server. For the penalty $\boldsymbol{\eta}$ of QoS, we imitate the AWS *Total Cost of Ownership* (TCO) pricing strategy [1] and determine the user cost in terms of server cost, storage cost, network cost, IT labor cost, and extra. Among these costs, the former three items are calculated based on the hardware costs, software costs, and energy consumption of power and cooling.

F. System Reliability

We define the performance metric "reliability" as follows. In Table 5, reliability refers to the percentage of time the system is available to deliver useful work. In traditional benchmarking, MTTF (mean time to failure) and MTTR (mean time to repair) are adopted to calculate availability [19].

We imitate these two factors to define two new factors, MTTE (mean time to exception) and MTTR (mean time to recover). MTTE refers to the average uptime between two abnormal system states, such as system hardware failure, power breakdown, device upgrade, and virus attack. MTTR accounts the average recovering time from the occurrence of exception to the time spot of system normal operation. The system reliability is defined by:

$$System\ Reliability\ (Re) = \frac{MTTE}{MTTE + MTTR} \tag{8}$$

Compared with availability, reliability considers not only system failures but also some security issues. MTTR accounts both the system maintenance and repairing time, and the operational anti-virus time while facing a security attack. Using this newly defined reliability, we evaluate how reliable a DCE system is in providing stable services for users.

5.4 Metric Composition Models

The performance of a DCE system is represented by the following six tuples of performance metrics, where T, A, I, R, Q, and Re represent the training time, model accuracy, inference speed, response time, quality of service, and system reliability, respectively.

$$Performance\ Metric\ Tuple = \{T,\ A, I,\ R,\ Q,\ Re\} \tag{9}$$

In general, the larger the area of the polygon, the higher the performance demonstrated. Here, we propose two different types of models for assessing the composite performance of AI applications in DCE systems. The uniform-weight model assumes all six metrics are equally weighted, while the weighted composite metric performance model allocates them with different weights.

(a) The Uniform Performance Model:

In a uniform model, we assume that all six metric dimensions are equally weighted. Each metric corresponds to a triangle area on the radar chart, which is calculated by:

$$M_i = \frac{1}{2} \times \boldsymbol{a} \times \boldsymbol{b} \times \sin \tag{10}$$

where $\boldsymbol{\theta}$ is the angle between lines a and b. The total polygon is the summation of six triangle areas corresponding to six performance metrics.

Suppose the performance value is $\{M_i \mid i = 1,\ldots,6\}$, the larger is the M_i, the better the performance is shown with this metric. In our experiment, the value of each metric dimension scales from 1 to 5. Consider M_i and M_{i+1} are the lines to calculate triangle areas, the polygon area is calculated by:

$$Polygon\ Area = \sum_{i=1}^{n} \left(\frac{1}{2} \times \sin\left(\frac{2\pi}{n}\right) \times M_i \times M_{i+1} \right) \tag{11}$$

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

TABLE 6
AI Servers and Key Parameters

Server/Device	Server Type	Computing Capacity	Storage (GB)
S1. Inspur Cloud Server	Cloud server	110 TFLOPS	6700
S2. Huawei Atlas 200DK	Edge Server	22 TOPS	52
S3. Inspur Edge Server	Edge Server	563.2 FLOPS	960
S4. Raspberry Pi 4B	End Device	192 GFLOPS	15

The notation $\frac{1}{2} \times \sin(\frac{2\pi}{n}) \times M_i \times M_{i+1}$ stands for the triangle area between metric i and $i+1$ (M_{n+1} is M_1 due to the polygon setting). The summation adds up the six triangles. The maximum value of each metric is $\boldsymbol{\omega} = 5$ in our experiment. By using $\boldsymbol{\omega}$, we calculate the upper bound area, which stands for the best performance this system can realize when all performance metrics reach the maximum values. Using the same polygon area equation in Eq. (10), this upper bound area is calculated by:

$$Upper\ bound\ Area = \sum_{i=1}^{n}\left(\frac{1}{2} \times \sin\left(\frac{2\pi}{n}\right) \times \omega \times \omega\right) \tag{12}$$

(b) The Weighted Performance Model:

In a weighted performance model, each metric is assigned with some weights, as defined by $\boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_n\}$, where n is the number of metrics. Ω is scaled until the maximum value of $\boldsymbol{\Omega}$ equals to the maximum value of M_i, which is ω.

The metric with higher weight accounts more for the overall assessment of the system performance, which is more suitable for the platform with special needs. Since ω is the maximum value of $\boldsymbol{\Omega}$. The importance of each performance metric M_i will be measured by $\frac{\omega_i}{\omega}$. The polygon area of the weighted area for each AI scenario is defined as:

$$Polygon\ Area = \sum_{i=1}^{n}\left(\frac{1}{2} \times \sin\left(\frac{2\pi}{n}\right) \times \frac{\omega_i}{\omega} M_i \times \frac{\omega_{i+1}}{\omega} M_{i+1}\right) \tag{13}$$

Since ω is also the maximum value of M_i, the upper bound of $\frac{\omega_i}{\omega} M_i$ will be given by ω_i. And the upper bound area for the maximum values can be calculated as follows:

$$Upper\ Bound\ Area = \sum_{i=1}^{n}\left(\frac{1}{2} \times \sin\left(\frac{2\pi}{n}\right) \times \omega_i \times \omega_{i+1}\right) \tag{14}$$

6 AI Benchmark Experiments and Results

This section presents our experimental settings. We report the measured performance results from the AIRS cloud platform. We focus on three AI performance metrics: namely the *model accuracy*, *training time*, and *logic inference speed*.

6.1 Experiment Settings and Model Training

We conduct our experiments on the AIRS cloud. We have two types of experiments: ML model training and logic inference. For model training, we use a server with an Intel Xeon6252 CPU with 48 cores and 8 Nvidia V100 GPUs. For logic inference, we tested four server types as listed in Table 6.

The cloud server is deployed on a 4-node GPU cluster of AIRS Clouds. We use an Inspur server with an Nvidia 2080Ti GPU card. For the edge server, we use an Inspur edge server without GPU cards. The Huawei Atlas 200DK has an AI accelerator chip. We use a Raspberry Pi 4B with four 1.5-GHz clock and quad-core Broadcom CPU cores and 4GB RAM.

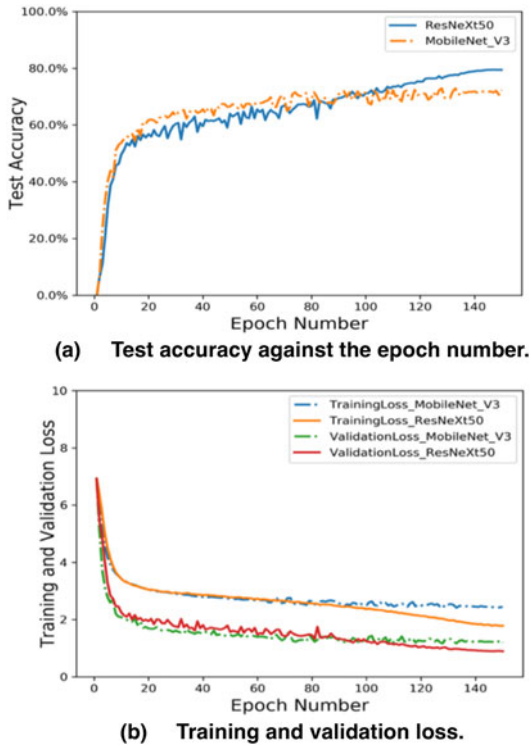

Fig. 9. Accuracy and loss comparison between two image classifier programs.

We train the image classifier on two large benchmark programs on the cloud servers. The dataset used is Imagenet, which has 1000 categories of images. After 150 epochs, the ResNeXt50 and MobileNetV3 take 41 and 38 hours on the S1 servers, respectively. The training times on S2, S3, and S4 servers are much greater than S1. Then we analyze the cross-validation accuracy on those edge servers or end devices.

6.2 Cross Validation Accuracy

Fig. 9 reveals that the MobileNetV3 classifier outperforms the ResNeXt50 classifier. They both reach 60% accuracy after 20 epoch iterations. After 150 epochs of training, the ResNet50 model improves to 79.42% accuracy. However, the MobileNetV3 model just reaches 72.31% accuracy. Fig. 9b shows that training loss is always higher than that validation loss. They become saturated after 150 epoch iterations.

Fig. 10(a, b) show the average and worse-case RoC (*Receiver Operating Characteristics*) curves of two image classifiers. The average RoC (dotted curves) serves as an upper bound of the worse-case RoC (solid curves) performance. The area under the RoC curve shows the tradeoff between the pros (false positive) and cons (false negative) of the classifiers. The larger is the underlying RoC area, the lower is the cross-validation loss, and the better is the test accuracy.

What we have measured shows an underlying area of size 1.0 for the average performance. In Fig. 10a, the ResNeXt

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

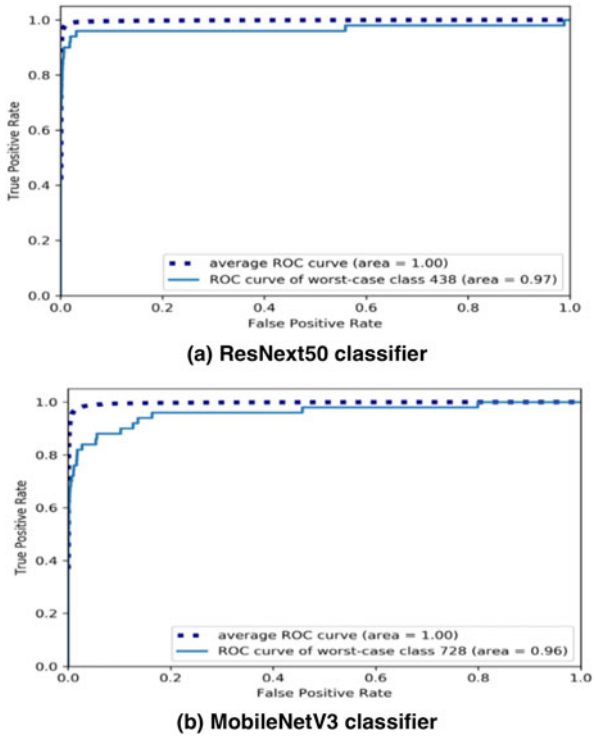

Fig. 10. ROC curves showing the average and worst-case performance of two image classifiers.

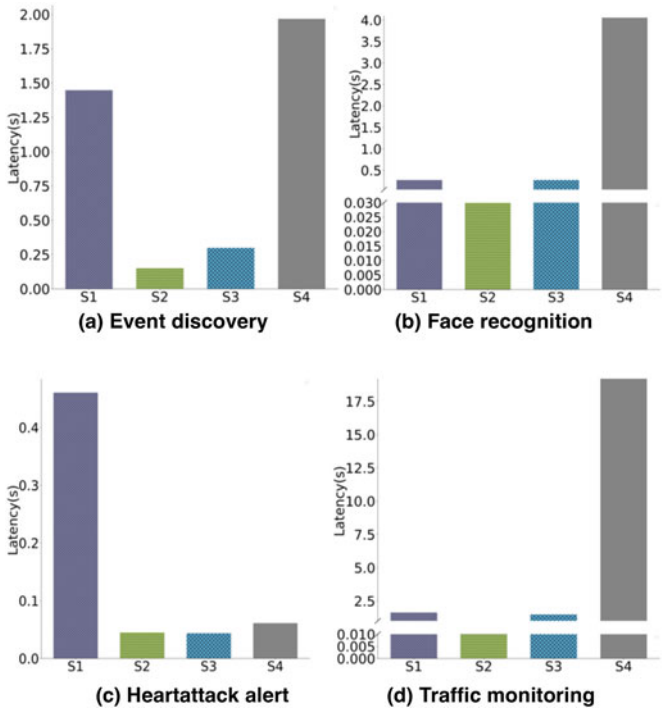

Fig. 12. ML response time is experienced in four scenarios where S1, S2, S3, and S4 are four server types defined in Table 6.

classifier has a worst-case ROC area of 0.96, corresponding to class 438. Fig. 10b shows a lower RoC curve for the MobileNetV3 Classifier with a 0.96 area corresponding to class 728 performances.

6.3 Inference Speed of Pretrained Model

We execute inference tasks on four server types and compare their inference times. The results are reported in Fig. 11. The cloud server S1 is certainly the fastest one in all scenarios.

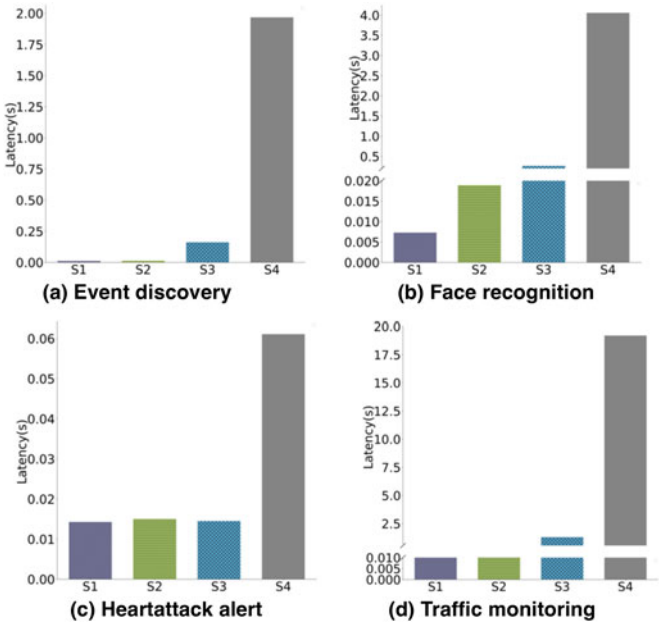

Fig. 11. ML Inference time on four server types, S1 denotes cloud server, S2 is Atlas edge server, S3 is Inspur edge server, and S4 is Raspberry device.

Two edge servers S2 and S3 are next higher and S4. The raspberry server is very slow in training ML models, 1000 slower than that of using cloud server S1.

For scenario (c) heartattack alert, the inference times of S1, S2, and S3 are basically at the same level because the model program is small in size. For event discovery, face recognition, and other tasks of the same magnitude, it's better to offload the tasks to the edge server, even to the cloud server. Nevertheless, for some other tasks, such as heartattack alert and traffic monitoring, it gains little.

6.4 Response Time At Various Servers

Fig. 12 shows the end-to-end response time from sending requests to completing processing on different servers. We assume the cloud server is far enough from the end device. Therefore, we use the communication network latency between servers in China and the USA.

Fig. 12 shows that all data bars are longer than the corresponding bars in Fig. 11. We find that server S1 has the longest response time in scenario (a)(b)(c). For scenario (d), since the inference time is really long for server S4, the response time of S4 is longer than S1. The server type S1 has a much longer response time due to the offloading delays to the cloud.

For the remaining servers, there exists little difference between the inference time and response time. We recommend using the cloud server type S1 for model training and cross-validation purposes. The two edge servers S2 and S3 could be used for logic inference purposes due to the small response time involved. The Raspberry Pi server S4 is used only for data collection. The recorded machine downtime was 86 hours in 497 days in 2020-2021. The system suffered from some virus or worm attacks during the period.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

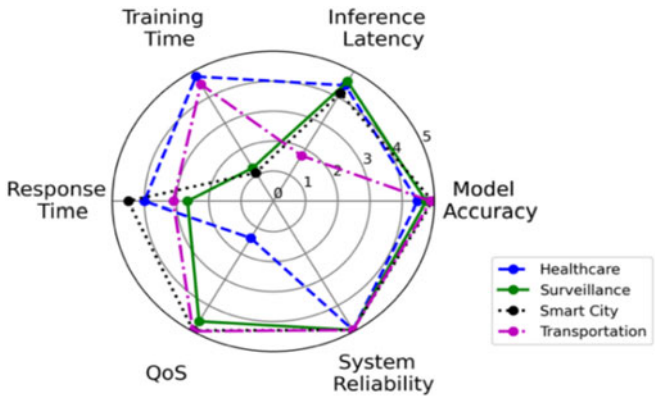

Fig. 13. The composite performances of 4 AI application scenarios under six specifically selected performance metrics under the uniform weighting.

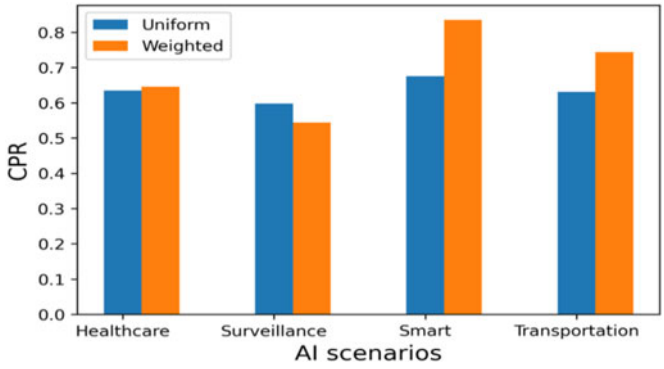

Fig. 15. Performance of a general-purpose DCE system with a comparison between the uniform metric model and the weighted metric model for four AI scenarios studied in this paper.

7 Composite Performance Results

We use the uniform and weighted models defined in Section 5.5 to present the composite performance of our DEC system. The results are displayed by weighted radar charts.

7.1 Weighted Composite Performance

Radar charts shown in Fig. 13 are often used in showing the composite performance along various metric dimensions. We consider six performance metrics defined in Section 5. Each metric is shown in 5 scales ranging from 0 at the center to 5 at the outmost ring. The higher is the scale ring, the higher is the performance along various dimensions.

The training, inference, and response are all-time measures, which are inversely proportional to their dimensional scale. For the uniform model, the measured performance of four AI application scenarios is plotted by four polygons in Fig. 13 under the assumption that all six metric dimensions are equally weighted.

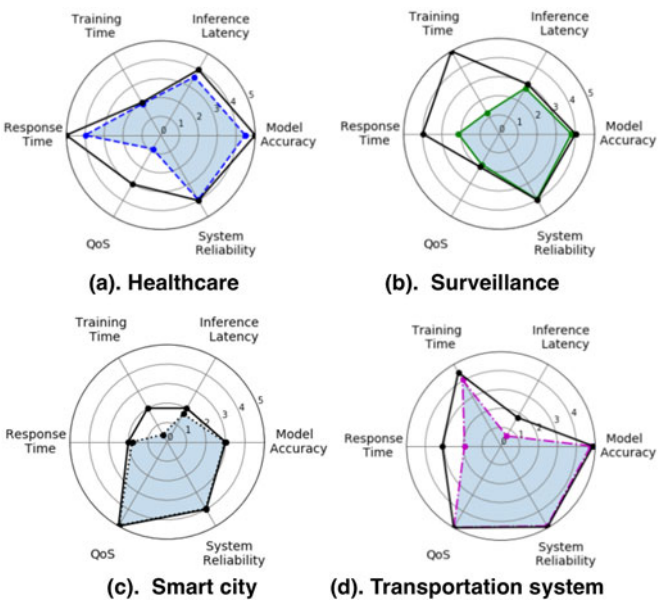

Fig. 14. Composite performance of four application scenarios under six weighted metrics. The black curve is the upper bound area corresponding to the metric weights in Table 3.

The weight distribution determines the upper-bound on the scenario performance, displayed by shaded polygons. The closer is the shaded polygon to the outer upper bound, the better the performance achieved. Thus, the weight distribution really sets a theoretical bound on the composite performance. The metric weights of 4 AI scenarios are specified in Table 3.

We use shaded polygons to highlight the fact that they plot the actual experimental performance results. The outer polygon without shading corresponds to an upper bound (Eq. (12)) on the measured performance for each scenario. The weighted performance display appeals better to satisfy users' concerns.

7.2 Performance Across Multiple Scenarios

We define below the composite performance ratio (CPR) for any given AI scenario.

$$Composite\ Performance\ Ratio\ (CPR) = \frac{Measured\ Polygon\ Area}{Upper\ Bound\ Area} \tag{15}$$

This CPR metric compares the composite performance area with the upper bound area for each AI scenario. The higher is the CPR value, the better is the DCE performance for a specific AI application as seen in Figs. 13 and 14 under the uniform and weighted distribution of all 6 metrics selected.

Fig. 15 compares the relative performance between unform and weighted assumptions on the CPR. The weighted performance is shown by orange bars is higher in 3 out of 4 scenario cases. Only the surveillance camera case has a better performance of uniform distribution than that of the weighted distribution. This is due to the fact that distributed surveillance applications demand even performance in all metric dimensions. Thus, unform performance model applies better. In summary, we see the performance gains of 2% to 20% in the three weighted AI application scenarios.

The smart city bar ranks the highest for having the smallest white gap area from its upper bound. The transportation map ranks the second place. The healthcare map ranks the third. The camera surveillance System has the lowest performance. With a different set of metrics selected or weight distribution changed by user demand, the relative performance could be different from what we have reported.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

8 Conclusion

This paper presents a new AI benchmark suite, Edge AIBench, for testing the performance of distributed cloud/edge computing systems. This is a scenario-driven benchmark suite initiated by BenchCouncil for general-purpose AI applications. We have modified the original codes to fit with the special DCE environments. We proposed two cloud/edge performance models with six performance metrics inspired by the chosen AI application scenarios.

We have tested the Edge AIBench programs on the AIRS cloud, a research platform specifically designed for distributed AI and machine learning applications. This AI cloud system has won the 2020 Wu-Wenjun Award for AI Natural Science Achievement from China's Association for Artificial Intelligence (CAAI). The system architecture of the AIRS cloud/edge platform was first-hand reported here.

Our scientific contributions are summarized below in 6 technical aspects: These claims are based on our custom system design and real-life benchmark experiences in building the 5G cloud/edge system, AIRS at CUHK-Shenzhen.

1) We have identified major resource requirements in using 5G IoT-connected cloud and edge (DCE) systems for distributed AI and ML applications.
2) A new weighted performance model was developed for evaluating the DCE computing systems for massive AI and IoT mobile applications.
3) We have developed a new suite of scenario-based AI benchmark programs and tested them successfully on the AIRS system at CUHK-Shenzhen.
4) Six scenario-based performance metrics are fully specified with mathematical formulations. We substantiate the effectiveness of these metrics by enclosure ten ML, DL, and AI kernel functions developed in our benchmark suite.
5) We revealed the relative strength and shortcomings in 5 existing AI benchmark suites. Our finding is that they do complement with each other to offset their shortcomings in assessing the performance of specific platforms.
6) Our DCE model enables the visualization of scenario-based performance. If the metric weights cannot be estimated in advance, one can apply the uniform model without any bias in user expectation.

For further research, we suggest expanding the Edge AIBench to include a full-scale library of AI operations at all three resource levels of the DCE system. At the cloud level, AI benchmark programs in MLPerf and AIPerf can be used jointly with our Edge AIBench. They are all subject to cluster parallelism and network virtualization constraints.

At the edge or embedded level, the DeFog can complement with those programs listed in our Edge AIBench. At the mobile device and embedded level, testing on privacy and security threats posts an important issue worthy of further studies. In summary, we feel that all existing AI benchmark suites are complementing with each other to cover both general-purpose and special AI applications.

Acknowledgments

We thank Inspur Company for the construction and maintenance of the AIRS Cloud being reported here. For the planning and testing of the AIRS cloud, we would like to thank our team members: Ma, Renwen, Shuai, Kefan, Li, Zhengdao, Li, Yonggang, Dai, Wei and Liu, Jian for their technical assistance. We want to acknowledge the useful discussions with Prof. Chen Min of Huazhong Univ. of Science and Technology and with our colleague, Dr. Miao Yiming in CUHK-Shenzhen. Mr. Liang, Minghao at Inspur Company have assisted us in the development of the AIRS Cloud.

References

[1] AWS Cloud Economics, "Internal report by amazon corp.," Accessed: May 18, 2022. [Online]. Available: https://pages.awscloud.com/rs/112-TZM766/images/Cloud%20Economics%20Ebook_October%202018.pdf
[2] "Artificial intelligence and compute," Accessed: May 18, 2022. [Online]. Available: https://openai.com/blog/ai-and-compute
[3] "Benchcouncil: International open benchmark council," Accessed: Jun. 16, 2021. [Online]. Available: https://www.benchcouncil.org/
[4] 5G Cloud, AIoT, and Edge Computing, Accessed: May 18, 2022. [Online]. Available: https://airs.cuhk.edu.cn/en/event/455
[5] Tusimple competitions for cvpr2017. Accessed: Jun. 16, 2022. [Online]. Available: https://github.com/TuSimple/tusimple-benchmark
[6] Y. H. Chang, J. Pu, W. M. Hwu, and J. Xiong, "MLHarness: A scalable benchmarking system for MLCommons," *BenchCouncil Trans. Benchmarks Standards Evaluations*, vol. 1, no. 1, 2021, Art. no. 100002.
[7] K. Cheng, Y. Zhou, B. Chen, R. Wang, Y. Bai, and Y. Liu, "Guardauto: A decentralized runtime protection system for autonomous driving," *IEEE Trans. Comput.*, vol. 70, no. 10, pp. 1569–1569–1581, Oct. 2020.
[8] E. Cosgrove, "One billion surveillance cameras will be watching around the world in 2021, a new study says," Accessed: Jun. 16, 2021. [Online]. Available: https://www.cnbc.com/2019/12/06/one-billion-surveillancecameras-will-be-watching-globally-in-2021.html
[9] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[10] EEMBC, "EEMBC MLMark," 2019. [Online]. Available: https://www.eembc.org/mlmark/index.php
[11] W. Gao *et al.*, "Aibench scenario: Scenario-distilling ai benchmarking," in *Proc. 30th Int. Conf. Parallel Architectures Compilation Techn.*, 2021, pp. 142–158.
[12] T. Hao, "Benchmarking, performance metrics, and modeling of cloud-edge-end AI computing systems," Ph.D. thesis, Res. Center Adv. Comput. Syst. Inst. Comput. Technol. Chinese Acad. Sci., Beijing, China, 2022.
[13] T. Hao *et al.*, "Edge AIBench: Towards comprehensive end-to-end edge computing benchmarking," in *Proc. Int. Symp. Benchmarking Measuring Optim.*, 2018, pp. 23–30.
[14] T. Hao, J. Zhan, K. Hwang, W. Gao, and X Wen, "AI-oriented workload allocation for cloud-edge computing," in *Proc. IEEE/ACM 21st Int. Symp. Cluster Cloud Internet Comput.*, 2021, pp. 555–564.
[15] H. Harutyunyan, H. Khachatrian, D. Kale, G. Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Sci. Data*, vol. 6, no. 1, pp. 1–18, 2019.
[16] J. Hennessy and D Patterson, *Computer Architecture: A Quantitative Approach*. Amsterdam, Netherlands: Elsevier, 2011.
[17] G. Huang and E. Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep. 14, no. 003, 2014.
[18] K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*, Burlington, MA, USA: Morgan Kaufmann, 2011.
[19] K. Hwang, X. Bai, Y. Shi, M. Li, W. Chen, and Y. Wu, "Cloud performance modeling with benchmark evaluation of elastic scaling strategies," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 130–143, Jan. 2015.
[20] K. Hwang and M Chen, *Big-Data Analytics for Cloud, IoT and Cognitive Computing*. Hoboken, NJ, USA: Wiley, 2017.
[21] A. Johnson *et al.*, "Mimic-iii, a freely accessible critical care database," *Sci. Data*, vol. 3, no. 1, pp. 1–9, 2016.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.

[22] P. Mattson *et al.*, "MLPerf: An industry standard benchmark suite for machine learning performance," *IEEE Micro*, vol. 40, no. 2, pp. 8–16, Feb. 2020.
[23] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "DeFog: Fog computing benchmarks," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, 2019, pp. 47–58.
[24] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[25] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 5206–5210.
[26] S. Raza, S. Wang, M. Ahmed, and M. Anwar, "A survey on vehicular edge computing," *Wireless Commun. Mobile Comput.*, vol. 2019, 2019, Art. no. 3159762.
[27] V. Reddi *et al.*, "MLPerf inference benchmark," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Architecture*, 2020, pp. 446–459.
[28] Z. Ren *et al.*, "AIPerf: Automated machine learning as an AI-HPC benchmark," *Big Data Mining Analytics*, vol. 4, no. 3, pp. 208–220, 2021.
[29] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[30] K. Soomro, A. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.
[31] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2011, pp. 1453–1460.
[32] F. Tang *et al.*, "AIBench training: Balanced industry-standard AI training benchmarking," in *Proc. IEEE Int. Symp. Perf. Anal. Syst. Softw.*, 2021, pp. 24–35.
[33] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *Proc. IEEE 22nd Int. Conf. Pattern Recognit.*, 2014, pp. 34–39.
[34] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jan. 2020.
[35] J. Zhan, "Call for establishing benchmark science and engineering," *BenchCouncil Trans. Benchmarks Stand. Eval.*, vol. 1, no. 1, 2021, Art. no. 100012.
[36] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1116–1124.

Tianshu Hao received the bachelor's degree in software engineering from Nankai University, China, in 2015. She is currently working toward the PhD degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. This work was done while she was visiting the AIRS during 2020∼2021. Her research interests include cloud/edge computing, artificial intelligence, and benchmarking high-performance computer systems.

Kai Hwang (Life Fellow, IEEE) received the PhD degree in electrical engineering and computer sciences from the University of California at Berkeley. He is a presidential chair professor with the Chinese University of Hong Kong (CUHK), Shenzhen. He has worked with Purdue University and University of Southern California for 45 years prior joining the CUHK in 2018. He has published 10 scientific books and more than 300 scientific papers with a Google citation exceeding 22000 times and h-index 65, he has received numerous awards including the very first CFC Oversea Achievement Award in 2005, the Lifetime Achievement Award from IEEE CloudCom in 2012. and the Wu Wenjung AI Natural Science Award from China's Association for Artificial Intel.

Jianfeng Zhan received the PhD degree in computer science from the Institute of Software, CAS, and UCAS, in 2002. He is a full professor with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and University of Chinese Academy of Sciences (UCAS), and director of Research Center for Advanced Computer Systems, ICT, CAS. He founds and chairs BenchCouncil and serves as the Co-EIC of TBench with Prof. Tony Hey. He has served as *IEEE Transactions on Parallel and Distributed Systems* associate editor since 2018. He received the second-class Chinese National Technology Promotion Prize in 2006, the Distinguished Achievement Award of the Chinese Academy of Sciences in 2005, and the IISWC Best Paper Award in 2013, respectively.

Yuejin Li received BS degree from Wuhan University, in 2016, and the two master's degrees from the London School of Economics and Political Science, in 2017, and the University of St. Andrews, in 2018. He is currently working toward the PhD degree in computer and information engineering with the Chinese University of Hong Kong, Shenzhen. His research interests include distributed cloud/edge computing and artificial intelligence.

Yong Cao is currently working toward the PhD degree with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests include cognitive computing, machine learning, and signal processing.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Turun Yliopisto. Downloaded on December 09,2025 at 07:20:55 UTC from IEEE Xplore. Restrictions apply.