Computational Intelligence and Deep Learning for Next-Generation Edge-Enabled Industrial IoT

Shunpu Tang, Lunyuan Chen, Ke He, Junjuan Xia, Lisheng Fan, and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—In this paper, we investigate how to deploy computational intelligence and deep learning (DL) in edge-enabled industrial IoT networks. In this system, the IoT devices can collaboratively train a shared model without compromising data privacy. However, due to limited resources in the industrial IoT networks, including computational power, bandwidth, and channel state, it is challenging for many devices to accomplish local training and upload weights to the edge server in time. To address this issue, we propose a novel multi-exit-based federated edge learning (ME-FEEL) framework, where the deep model can be divided into several sub-models with different depths and output prediction from the exit in the corresponding sub-model. In this way, the devices with insufficient computational power can choose the earlier exits and avoid training the complete model, which can help reduce computational latency and enable devices to participate into aggregation as much as possible within a latency threshold. Moreover, we propose a greedy approach-based exit selection and bandwidth allocation algorithm to maximize the total number of exits in each communication round. Simulation experiments are conducted on the classical Fashion-MNIST dataset under a non-independent and identically distributed (non-IID) setting, and it shows that the proposed strategy outperforms the conventional FL. In particular, the proposed ME-FEEL can achieve an accuracy gain up to 32.7% in the industrial IoT networks with the severely limited resources.**

***Index Terms*—Computational intelligence, edge computing, federated learning, exit selection.**

I. Introduction

With the advance of communication technologies and the emergence of new communication standards such as 5G and WiFi-6, more and more devices are connected to the Internet through wireless access points, which is called the era of Internet of Things (IoT) [1]. A large amount of data is generated and collected by many sensors and mobile devices, promoting the development of novel applications, including autonomous driving, augmented reality (AR) and smart cities. However, a mass of data at the edge of the network brings great challenges to the central cloud servers. Unfortunately, the conventional paradigm of cloud computing is unfriendly to latency-sensitive applications. To address these issues, mobile edge computing (MEC) [2] was proposed to deploy computational resources close to the data source. In the concept of MEC, IoT devices can offload data and computational tasks to the edge server to achieve a lower latency, lower energy consumption, and higher service quality.

In recent years, artificial intelligence (AI) technologies, e.g., machine learning (ML) and deep learning (DL), have made a breakthrough and even reached beyond human-level performances in image recognition, natural language processing, anomaly detection, and other domains [3]. To enhance the abilities of information processing and analysis, AI applications are deployed on edge devices to facilitate the emergence of edge intelligence [4], [5]. However, intelligent algorithms are usually computation- and energy-intensive. Meanwhile, limited computational power, battery capacity, and dynamic channel states restrict the applications of edge intelligence.

Generally, collecting data and training models are two critical steps to deploy edge intelligence in the IoT networks. Due to the decentralized nature of data in the IoT networks, it is inappropriate to apply the conventional centralized learning in the cloud, which will increase the communication overhead. More importantly, uploading personal data may cause the leakage of privacy. In addition, many new and even fiercer laws were passed to prevent storing users' data on third-party servers. To tackle these issues, a collaborative training approach named federated learning (FL) was proposed to reduce the communication overhead, storage space, and energy consumption. In the FL, the privacy can be protected by enabling multiple users to train a shared model by exchanging the weights instead of the sensitive raw data. It breaks away data islands and makes it possible to improve the performance of the data-driven model under the cooperation of multiple parties.

However, FL still faces some challenges in edge-enabled IoT networks. One major challenge in the industrial IoT networks is the large processing a latency from communication and computation, due to the limited resources such as computational power and bandwidth. Accordingly, a latency threshold is often used in the federated learning to coordinate and synchronize the FL training, and avoid the unnecessary wait time [6]–[9], as a large latency may cause many devices fail to participate into the federated learning in time. Another major challenge is the system heterogeneity, where there exist different kinds of IoT devices with different computational power and channel state. In this case, some devices with

S. Tang, J. Xia and L. Fan are all with the School of Computer Science, Guangzhou University, Guangzhou 510006, China (e-mail: tangshunpu@e.gzhu.edu.cn, {xiajunjuan, lsfan}@gzhu.edu.cn).

L. Chen is with the School of Electronics and Communication Engineering, Guangzhou University, Guangzhou 510006, China (e-mail: 2112019037@e.gzhu.edu.cn).

K. He was with the School of Computer Science and Cyber Engineering, Guangzhou University, China, and is now with the Signal Processing & Satellite Communications Research Group (SIGCOM), Interdisciplinary Centre for Security, Reliability and Trust (SnT) - University of Luxembourg, L-1855 Luxembourg (ke.he@uni.lu).

A. Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS, London, U.K (e-mail: a.nallanathan@qmul.ac.uk).

S. Tang and L. Chen contributed equally to this work.

The corresponding author of this paper is L. Fan.

sufficient resources can successfully accomplish the local training and uploading within the required latency, while the other devices with insufficient resources fail. Intuitively, we can choose a lightweight model for all devices. Nevertheless, this approach may suffer a severe performance degradation and lose flexibility when the channel state is improved. These two major challenges may severely degrade the system performance, and motivate us to design a novel and flexible framework of FEEL which enables devices to adaptively train a model with different scales according to the available resources, and the server can still aggregate weights from different devices.

Inspired by the multi-exit mechanism proposed in [10], this paper incorporates this mechanism into FL and proposes the multi-exit-based federated edge learning (ME-FEEL). Specifically, we firstly add multiple exits in the deep model and enable it to output prediction from any exit. Therefore, the deep model can be divided into several sub-models, and the devices can choose a suitable exit. Instead of training the whole deep model, all the devices can train parts of sub-models flexibly. Thanks to the multi-exit mechanism, the sub-models trained by different devices still have a practical common architecture that can be easily aggregated together. Moreover, a self-knowledge distilling (KD) approach is utilized to improve the performance of those early exits that lack of feature extraction and fitting abilities.

In further, how to choose the best exit still remains an open problem to be solved. Although each device can choose an exit as deep as possible without violating the latency constraint, uploading time can not be neglected in bandwidth-limited IoT networks, and it is usually a critical factor raising the failure aggregation. As described in [6]–[9], it is useful to make the server aggregate the updates as much as possible with the given bandwidth allocation, promoting the training performance and stability. Thus, we aim to maximize the number of devices which successfully upload their updates within the constrained latency. Notably, due to the use of multi-exit mechanism, an improved optimization objective is built to maximize the total number of exits which can upload successfully per FL round. We hence propose a heuristic exit selection and bandwidth allocation algorithm based on the greedy approach to solve the optimization problem.

The practical scenarios of the proposed framework include many applications such as visual drone inspection, self-driving cars, automatic sorting, and intelligent safeguard system. In these scenarios, the proposed framework can help more devices with limited resources to participate into the collaborative learning, promoting the system performance. In summary, this paper makes the following contributions.

- We take into account the complicated scenario of edge intelligence, where there are heterogeneous devices with different computing power and channel state. The gap of computational power among devices can even reach tens of times, and the total communication bandwidth is limited. All devices will train a shared deep model collaboratively under a latency constraint.
- Inspired by the multi-exit mechanism, we propose a novel FL training framework named ME-FEEL, where devices with limited computational power can still participate into the FL by training a part of the deep model. We also modify the teacher selection of KD according to the characteristic of the FEEL system. Moreover, a layer-wised model average strategy is presented to aggregate the models of different sizes. To the best of our knowledge, it is the first time applying these approaches in the federated edge learning system to overcome the limited-resource and system heterogeneity in the industrial IoT network and improve the flexibility of the federated edge learning.
- Incorporating the limited bandwidth into the IoT networks, we aim to maximize the number of devices which can successfully participate into the aggregation and propose an enhanced optimization objective that is to maximize the total number of exits per round. We also propose a joint exit selection and bandwidth allocation strategy based on the greedy approach to solve the problem.
- We conduct simulations and evaluate the proposed ME-FEEL by using a popular model and a classical dataset. Numerical results demonstrate that the proposed ME-FEEL can outperform the conventional strategies and it can be flexibly deployed in the edge-enabled industrial IoT networks.

The rest of this paper is organized as follows. Section II briefly discusses related works on MEC and edge intelligence, and then Section III presents the multi-exit FL and the associated workflow. Section IV builds the exit selection and bandwidth allocation problem and further provides the optimization algorithm. Simulations are performed in Section V to evaluate the proposed strategy. At last, the whole paper is concluded in Section VI.

II. Related works

The research of MEC has attracted much attention in recent years, and there are lots of relevant works about the offloading design and resource allocation [11]–[18]. With the development of AI technologies, edge intelligence has become one of the most interesting topics. The definition of edge intelligence includes two aspects of *AI for edge* and *AI on edge* [19], which are detailed as follows.

Firstly, *AI for edge* focuses on applying AI to deal with the complicated optimization problems, such as non-convex and NP-hard problems. For example, deep learning and reinforcement learning have achieved great success in the design of MEC networks, where a novel edge cache strategy based on deep Q-network (DQN) was proposed in [20], and DRL and FL could be exerted to optimize multiuser multi-CAP MEC networks [21], [22].

Moreover, *AI on edge* focuses on the deployment of intelligent algorithms at the edge of network. To deal with the problem of limited resources in the edge devices, efficient model architecture, model compressing approach and hardware acceleration techniques should be proposed [23]–[25]. Many researchers have designed some novel inference protocols from different perspectives. For example, the authors in [26] studied the characteristics of deep networks and presented an offloading strategy named Neurosurgeon, which could automatically partition DNN computation between mobile devices and cloud servers. Besides, the authors in [27], [28] proposed a framework of distributed adaptive inference between edge devices and servers to minimize the memory footprint and energy consumption. Overall, these works focused on the inference stage and assumed that the deep models were obtained from the central server. However, it is challenging to collect massive data for training the deep models in the practical IoT networks.

To enhance the performance of the edge-enabled IoT networks, researchers have made many efforts in the deployment of FL into the system. Specifically, the authors in [9] solved the problem of client selection in resource-constraint mobile edge computing networks, where only the bandwidth schedule protocol was investigated and the training latency was ignored in the control. The authors in [29] proposed a physical-layer quantization strategy for the uplink and downlink communications, where the strong heterogeneity in IoT networks was ignored and little attention was paid to the local training process. For the heterogeneous system, an intelligent schedule approach was proposed in [30] to enable the devices to reduce unnecessary energy consumption, where some devices were not the bottleneck of the system. In further, the authors in [8] studied the IoT devices powered by different batteries and proposed a deep deterministic policy gradient (DDPG)-based approach to prolong the battery life, where the conventional power control approaches were used with certain limitations. The work in [31] proposed a comprehensive optimization method by controlling the local iteration number, sampled devices number, and device scheduling, where the model size/scale was still ignored in the control.

In summary, the previous works assumed that the IoT devices trained a shared model with the same size/scale, which is however not practical in the edge-enabled IoT networks. Hence, this paper presents a flexible framework and enables all devices to train their local models with different scales from the perspective of the model architecture. Moreover, this paper jointly incorporates the training and uploading stages into the framework design, which can help optimize the performance of the FEEL.

III. Multi-exit based federated learning

In this section, we investigate the FEEL system in the IoT networks, which consists of one edge server and multiple IoT devices. We then present the key design of the proposed ME-FEEL by giving the formulation and working mechanism of multiple exits. We further explore the aggregation strategy for the models of different sizes. The main notations used throughout the paper are summarized in Table I.

A. Federated edge learning system

Fig. 1 depicts the system model of federated edge learning (FEEL), where there is one edge server and K IoT devices. For IoT device k with $1 \leq k \leq K$, it firstly collects the local dataset $\mathcal{D}_k = \{(\boldsymbol{x}_n, y_n) | 1 \leq n \leq N_k\}$, where N_k is the number of the local training samples in device k, while $\boldsymbol{x}_n$ and y_n denote the training sample and label, respectively. These K devices will train a shared deep model with the help of the edge server, which aims to minimize the global objective function $f(w)$, given by

$$\min_{w} f(w) = \sum_{k=1}^{K} \frac{N_k}{N} F_k(w), \quad (1)$$

where w is the model weights, $F_k(\cdot)$ represents the local loss function, and $N = \sum_{k=1}^{K} N_k$ is the total number of samples. For device k, it performs the training on its local dataset in parallel, and updates the weights by the gradient descent

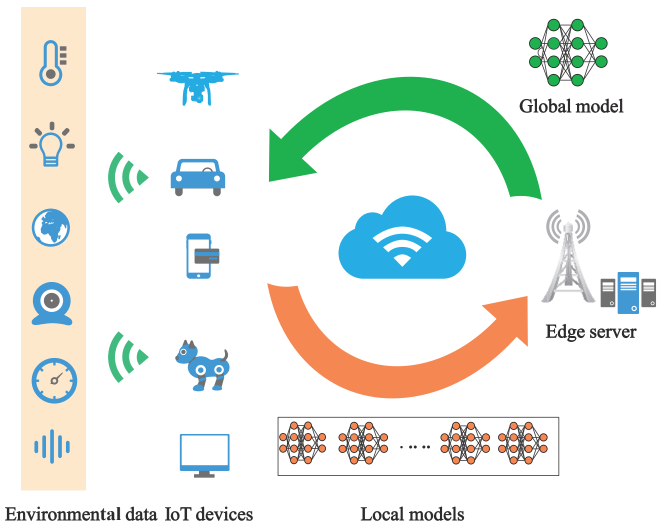


Fig. 1. Federated edge learning system.

TABLE I
Notation Summary

Notation	Definition
B_k	Bandwidth of device k
D_k	Dataset of device k
e	Local training times
$F_k(w)$	Local loss function of device k
$f(w)$	Global objective function
K	Number of devices
L	Number of layers in deep network
$\mathcal{L}_{pre}(\cdot)$	Loss function of a single sample
$\mathcal{L}_{kd}(\cdot)$	Loss function of KD
M	Number of exits
N_k	Size of dataset D_k
p_m	Output of exit m
R_k	Data rate of device k
$\boldsymbol{s}$	Output logits of the students
$\boldsymbol{t}$	Output logits of the teacher
$T_{local,k}$	Local training time of device k
$T_{up,k}$	Uploading time of device k
$\boldsymbol{x_n}$	Input vector of the n-th sample
y_n	Label of $\boldsymbol{x_n}$
$\mathcal{X}$	Set of input samples
w^r	Global weights at communication round r
w_k^r	Local weight of device k at communication round r
η	Learning rate
τ	Temperature parameter of KD

Algorithm 1: FEEL procedure

```
1 Input $K$, $R$, $\eta$ $w^0$;
2 for Round $r = 0, \cdots, R-1$ do
3     Server randomly chooses a subset $\mathcal{K}$ from $K$ devices;
4     Each device $k \in \mathcal{K}$ downloads the global weights $w^r$ from the server;
5     Each device $k \in \mathcal{K}$ obtains the weights $w_k^{r+1}$ by performing SGD with a learning rate $\eta$ on the local dataset;
6     Each device uploads the weights $w_k^{r+1}$ to the server;
7     The edge server aggregates the weights as $w^{r+1} = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} w_k^{r+1}$
8 end
```

algorithm,

$$w_k \leftarrow w_k - \eta \nabla F_k(w_k), \tag{2}$$

where η is a learning rate. In practice, to reduce the commutation overhead, the edge server randomly chooses one device subset $\mathcal{K}$ in per round, where $|\mathcal{K}| \ll K$. Then, the chosen device $k \in \mathcal{K}$ performs the gradient descent algorithm e times. After that, to optimize the global objective function, the local update will be uploaded to the edge server, which can aggregate the weights from different devices and obtain a global update by

$$w^{r+1} = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} w_k^{r+1}, \tag{3}$$

where r and $r+1$ are the indices of the current and next training rounds, respectively. The overall procedure of FEEL is summarized in Algorithm 1.

However, the aforementioned FEEL system ignores the realistic conditions in the IoT networks. Specifically, due to the limited computing power, the SGD processing with intensive computing in Step 5 of Algorithm 1 may cause a large latency. Moreover, the poor channel state and limited bandwidth also limit the uploading operation in Step 6. Accordingly, the failure of model aggregation may occur in Step 7. Therefore, it is of vital importance to significantly improve the system flexibility, which can adaptively control the number of training weights to reduce the latency of Steps 5-6 according to the available resources.

B. Proposed multi-exit approach

To tackle the system heterogeneity in the IoT networks, we propose a novel FEEL training framework named ME-FEEL, which enables the chosen devices to participate into training to the best of their abilities by setting multiple exits for the deep models.

As shown in Fig. 2 (a), in the original deep model such as a fully connected layer based classifier, there is only one exit which is usually the last layer. The features of input will be extracted by continuous convolution kernels and then be fed

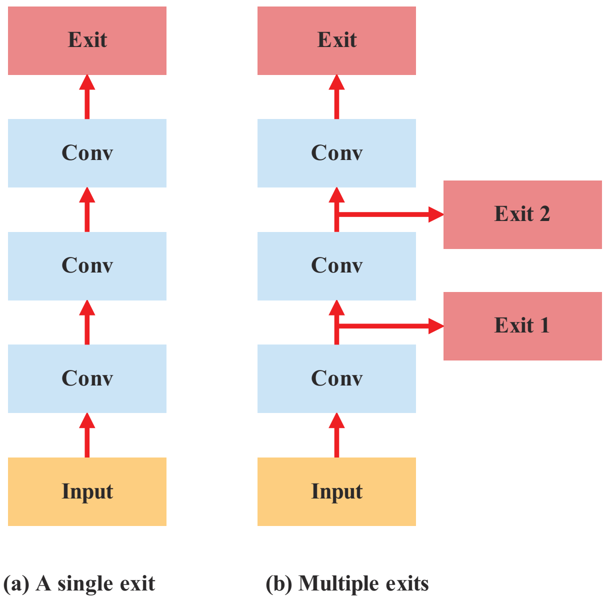


Fig. 2. An example of a single exit and multi-exit architecture for deep models.

into the classifier. The multi-exit architecture is illustrated in Fig. 2 (b), where there are multiple exits at different depths. In contrast, it is not necessary to wait until the computation in the last layer is completed to obtain the prediction result. All exits share a part of weights, and there are not many additional parameters added in the model. This makes the model separable and can be divided into multiple sub-models. Obviously, deeper exits cause more resource consumption and a larger latency. Formally, the output of the model with M exits can be expressed as

$$\mathcal{P} = \{\boldsymbol{p}_1, \cdots \boldsymbol{p}_M\}, \tag{4}$$

where $\boldsymbol{p}_m \in \mathcal{P}$ denotes the output logits of the m-th exit, and we use $p_m(x)$ to denote the mapping function from the input $x \in \mathcal{X}$ to $\boldsymbol{p_m}$, in which $\mathcal{X} = \{x_n | 1 \le x_n \le N_k\}$ is the set of input samples. In practice, device k can adaptively choose $M_k \le M$ and train the continuous part $p_1(x), \cdots p_{M_k}(x)$, depending on its computational power and channel state. Thus, it can quickly adapt to the dynamic environment in the IoT networks.

We further design the training framework of ME-FEEL shown in Fig. 3, and it consists of the following pivotal steps,

1) Local training: For an arbitrary device k, it can choose M_k exits according to its current state. Given the local dataset $\mathcal{D}_k$, the local model can output M_k results for each sample. Thus, to train all exits and shared weights, the local loss function on dataset $\mathcal{D}_k$ can be regarded as the sum loss of all exits, which can be written as

$$F_k(w; \mathcal{D}_k) = \frac{1}{N_k}\sum_{n=1}^{N_k} \mathcal{L}_{pre}(y_n, p_m(\boldsymbol{x}_n)), \tag{5}$$

where

$$\mathcal{L}_{pre}(y, p_m(\boldsymbol{x})) = \frac{1}{M_k}\sum_{m=1}^{M_k} \ell_{pre}(y, p_m(\boldsymbol{x})), \tag{6}$$

in which $\ell_{pre}(\cdot)$ represents the mean square error of the regression task and cross-entropy for classification tasks.

In general, the later exit outperforms the earlier one, because a deeper layer can often extract more abstract features and exhibit a stronger representational ability. In other words, the later exits often have more knowledge than the earlier ones. Hence, the earlier exits are hard to train well from scratch and do not have enough ability to process difficult samples properly. From the viewpoint of knowledge distilling (KD) [32], the exits with more knowledge can be viewed as teachers which can transfer their knowledge to the students with less knowledge in order to help improve the accuracy [10]. Therefore, each device can employ the KD-based strategy to train the multiple exits. The objective of KD is to minimize the difference between the output distributions of the teacher and students. Formally, we use Kullback-Leibler (KL) divergence to describe the difference, which can be written as

$$\ell_{kd}(\boldsymbol{s}, \boldsymbol{t}) = -\tau^2 \sum_i z_i(\tau) \log v_i(\tau), \tag{7}$$

where $\boldsymbol{s}$ and $\boldsymbol{t}$ are the output logits of the students and teacher, respectively. Notation τ is the temperature parameter of KD, and $v_i(\tau)$ can be represented as

$$v_i(\tau) = \frac{e^{s_i/\tau}}{\sum_j e^{s_j/\tau}}. \tag{8}$$

Analogously, notation $z_i(\tau)$ is defined as

$$z_i(\tau) = \frac{e^{t_i/\tau}}{\sum_j e^{t_j/\tau}}. \tag{9}$$

Note that the temperature parameter τ can be used to soften the output of the teacher [32]. In particular, when $\tau = 1$, (8)-(9) degrade into the SoftMax function, and it is easy to ignore the information from the negative samples. On the contrary, a higher temperature with $\tau > 1$ can avoid the overconfidence of neural network and it can allow the output to contain more information on the similarity of different classes. Moreover, it is important to multiply τ^2 in (7) to ensure that the change of temperature will not affect the gradient magnitude.

However, the later exits may not always outperform the earlier ones in the practical FEEL systems. These results lie in that, due to the limited resources, many devices cannot choose the later exits, and the aggregation number of later exits may be smaller than that of the earlier ones. Besides, there is a risk of overfitting in the later exits. Consequently, directly choosing the later exits as teachers in [32] is unsuitable for the FEEL systems. Inspired by the fact that the ensemble model has more robust generalization performance than a single model, we choose the ensemble output of all exits as the teacher $\boldsymbol{t}(\boldsymbol{x})$, which can be written as

$$\boldsymbol{t}(\boldsymbol{x}) = \frac{1}{M_k} \sum_{m=1}^{M_k} p_m(\boldsymbol{x}). \tag{10}$$

Hence, the total KD loss can be computed as

$$\mathcal{L}_{kd}(\boldsymbol{x}) = \frac{1}{M_k} \sum_{m=1}^{M_k} \ell_{kd}(p_m(x), \boldsymbol{t}(\boldsymbol{x})). \tag{11}$$

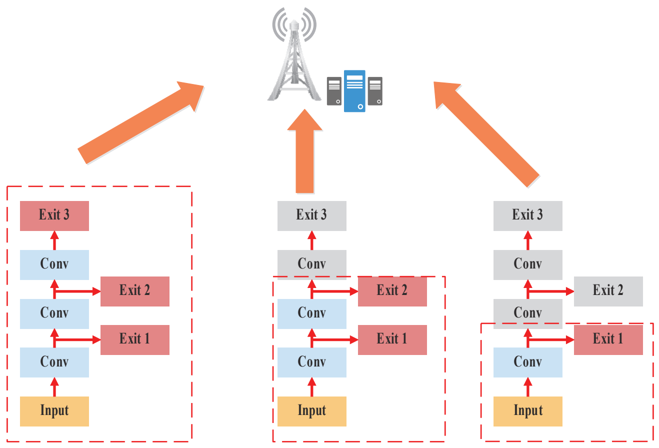


Fig. 3. Training framework of ME-FEEL.

By jointly using the prediction loss and KD loss, (5) can be rewritten

$$F_k(w; \mathcal{D}_k) = \frac{1}{N_k} \sum_{n=1}^{N_k} \left[\mathcal{L}_{pre}(y, p_m(\boldsymbol{x}_n) + \mathcal{L}_{kd}(\boldsymbol{x}_n)\right], \tag{12}$$

which will be minimized by the local gradient descent.

2) Model aggregation: After each device completes the local update in parallel, the updated weights will be uploaded to the edge server. In the conventional FEEL, the updated weights of different devices are of the same dimension. Therefore, model aggregation should be modified to adapt the architecture of ME-FEEL.

To tackle this problem, a simple but efficient strategy is to aggregate the shared parts of different models. Although the upload weights are of different sizes, they all have the same sub-architecture. Specifically, each layer in the model with fewer exits can be found in the model with more exits. Thus, we propose a layer-wised model average strategy to obtain the global model, namely multi-exit FedAvg (ME-FedAvg). The edge server will take the initial global model as a reference, and then perform the search layer by layer in the uploaded updates. If one layer is found to exist in the model uploaded by some devices, then the edge server will average the weights of the layer from different devices.

Mathematically, suppose that there are L layers with parameters in the initial global model, and L_k is used to denote the number of layers in the k-th device. For each layer $l \in [1, L]$ in the global model, its weight $w(l)$ can be updated by

$$w(l) = \frac{\sum_{c \in \mathbb{C}_l} |D_c| w_c(l)}{\sum_{c \in \mathbb{C}_l} |D_c|}, \forall l \in [1, L], \tag{13}$$

where $\mathbb{C}_l$ is a subset of $\mathcal{K}$. The edge server searches for devices with the l-th layer, and then produces the device subset $\mathbb{C}_l$. This means that all the devices in $\mathbb{C}_l$ have layer l. In other words, the number of layers in the device $c \in \mathbb{C}_l$ is larger than the value of l, which can be written as

$$L_c \geq l, \forall c \in \mathbb{C}_l. \tag{14}$$

Algorithm 2: Multi-exit FEEL

```
1  Input $K$, $T$, $\eta$ $w^0$, $L$;
2  for Round $t = 0, \cdots, T-1$ do
3      The edge server randomly chooses a subset $\mathcal{K}$ from $K$ devices;
4      Each device $\in \mathcal{K}$ downloads the global weights $w^t$ from the server;
5      Device $k$ chooses $M_k$ exits and the MEC controller allocates bandwidth to the device according to the current environment;
6      Device $k$ obtains the weights $w_k^{t+1}$ by performing SGD on (12) with the learning rate $\eta$;
7      Device $k$ uploads the weights $w_k^{t+1}$ to the server;
8      for $l = 1, \cdots L$ do
9          The edge server produces the subset of devices $\mathbb{C}_l$;
10         The edge server aggregates the weights of the $l$-th layer as $w^{t+1}(l) = \frac{\sum_{c\in\mathbb{C}_l}|D_c|w_c(l)}{\sum_{c\in\mathbb{C}_l}|D_c|}$;
11     end
12 end
```

To summarize, we provide the procedure of the proposed ME-FEEL in Algorithm 2. Different from the convention FL, we set multiple exits in the deep model, and enable devices to train the model with different sizes. We will further propose a joint exit selection and bandwidth allocation strategy as shown in line 5 of Algorithm 2, which is detailed in the following section.

IV. Exit selection and bandwidth allocation

In this section, we present how to choose proper exit points and propose a joint exit selection and bandwidth allocation strategy, which can adapt the dynamic environment in the IoT networks. Specifically, we firstly describe the latency constrained model of FEEL, and then provide the problem formulation. We further propose a heuristic algorithm based on the greedy approach to solve the optimization problem.

A. Latency constrained model

As mentioned before, all devices train the shared model under a latency constraint denoted by γ_{th}. The total communication bandwidth is B_A, and it is managed by the MEC controller. Therefore, for each device in $\mathcal{K}$, it uploads its weights through wireless channel within γ_{th}, i.e.,

$$T_{\text{local},k} + T_{\text{up},k} \leq \gamma_{th}, \tag{15}$$

where $T_{\text{local},k}$ and $T_{\text{up},k}$ are the latency of local training and data transmission, respectively. The local latency depends on the number of exits and the number of samples in the local dataset D_k, which can be characterized by

$$T_{\text{local},k} = \frac{\alpha_k |D_k| \cdot g_1(M_k)}{C_k}, \tag{16}$$

where $\alpha_k > 0$ is a relative coefficient of computational power, C_k is the batch size of device k, and $g_1(\cdot)$ represents the mapping from the number of exits to the training latency of only one batch[1].

According to the Shannon theorem, the transmission data rate between device k and BS is

$$R_k = B_k \log_2\left(1 + \frac{P_k|h_k|^2}{\sigma_k^2}\right), \tag{17}$$

where B_k is the allocated channel bandwidth, P_k denotes the transmit power, h_k is the channel parameter, and σ_k^2 is the variance of the additive white Gaussian noise (AWGN). The size of the uploaded weights is also varying when the device chooses different exits. Therefore, the transmission latency is given by

$$T_{\text{up},k} = \frac{g_2(M_k)}{R_k}, \tag{18}$$

where $g_2(\cdot)$ is the mapping from the number of exits to the number of bits, and it is only dependent on the network architecture.

B. Problem formulation

In the FEEL system, it is useful to make the server aggregate updates as much as possible through reasonable resource management and scheduling, which can help improve training performance and stability. Similarly, in the proposed ME-FEEL, the number of exits also affects the learning performance. Thus, we aim to maximize the number of exits that can upload their updates without violating the latency threshold γ_{th}.

Mathematically, we use $\mathcal{K}_s$ to denote the device set in which the devices can upload their updates successfully, where $\mathcal{K}_s \subseteq \mathcal{K}$. Further, the problem formulation is give by

$$\max_{\{M_k, B_k\}} \quad \sum_{k\in\mathcal{K}_s} M_k \tag{P1}$$

$$\text{s.t.} \quad T_{\text{local},k} + T_{\text{up},k} \leq \gamma_{th}, \tag{c1}$$

$$\sum_{k\in\mathcal{K}_s} B_k \leq B_A. \tag{c2}$$

C. Optimization strategy

Note that the problem (P1) is a variant of classical knapsack problem (KP) and it belongs to the NP-complete problem [33]. It has been pointed out in the literature that (P1) can be solved at the cost of pseudo-polynomial time. To solve this problem, we propose a heuristic exit selection and bandwidth allocation algorithm based on the greedy approach[2], detailed as follows.

[1]Although $g_1(\cdot)$ is strongly related to the used baseline hardware and some differences may be caused when the baseline hardware is changed, we can set α to a reasonable value in order to reflect the CPU capabilities of IoT devices in the practical networks. Specifically, if the baseline hardware is powerful, we can set α to a large value to simulate the IoT devices with limited CPU capabilities. On the contrary, we can set α to a small value to reflect the IoT devices with powerful CPU capabilities. In this way, we can use an arbitrary baseline hardware and model the heterogeneous devices flexibly. The detailed simulation settings can be found in Sec. V.

[2]There exist some other exit selection criteria such as the brute force algorithm, which may outperform the proposed selection criterion in our work at the cost of a much higher computational complexity due to exhaustive search.

Algorithm 3: Exit selection and bandwidth allocation

```
1  Input $K$, $B_A$, $M$;
2  for $k \in \mathcal{K}$ do
3      $m = M$;
       // Computing the required bandwidth
          according to (20)
4      while $B_k(m) \leq 0$ or $m > 0$ do
5          $m = m - 1$;
6      end
7      if $m$>0 then
8          Device $k$ chooses exit $M_k = m$;
9          Device $k$ sets bandwidth to $B_k = B_k(M_k)$;
10         Add device $k$ to $\mathcal{K}_s$
11     end
12 end
   // Adjust the exit selection
13 $B = \sum_{k \in \mathcal{K}_s} B_k$;
14 while $B > B_A$ do
15     $k_{min} = \arg\min_{k \in \mathcal{K}_s} \frac{M_k}{B_k}$;
16     $M_{k_{min}} = M_{k_{min}} - 1$;
17     $B_{k_{min}} = B_{k_{min}}(M_{k_{min}})$;
18     $B = \sum_{k \in \mathcal{K}_s} B_k$;
19 end
20 Output $\{B_k, M_k\}$
```

From (15)-(18), the bandwidth $B_k(m)$ for uploading m exits with the constraint c1 should meet the following requirement

$$\frac{g_2(m)}{B_k(m) \log_2\left(1 + \frac{P_k |h_k|^2}{\sigma_k^2}\right)} + \frac{\alpha_k |D_k| \cdot g_1(m)}{C_k} \leq \gamma_{th}, \quad (19)$$

where $m \in [1, M]$ is the index of exits. We then have

$$B_k(m) \geq \frac{g_2(m)}{\left(\gamma_{th} - \frac{\alpha_k |\mathcal{D}_k| g_1(m)}{C_k}\right) \log_2\left(1 + \frac{P_k |h_k|^2}{\sigma_k^2}\right)}. \quad (20)$$

Therefore, we can set $B_k(m)$ to the minimum required bandwidth for the m-th exit,

$$B_k(m) \triangleq \frac{g_2(m)}{\left(\gamma_{th} - \frac{\alpha_k |\mathcal{D}_k| g_1(m)}{C_k}\right) \log_2\left(1 + \frac{P_k |h_k|^2}{\sigma_k^2}\right)}. \quad (21)$$

The key idea of making decision is to adjust the exit according to the ratio of exit number M_k to the minimum required bandwidth $B_k(m)$. This exit-to-bandwidth ratio $\frac{M_k}{B_k}$ indicates that whether the device can provide exits to the system at a low bandwidth cost or not. In particular, if the total required bandwidth exceeds the bandwidth limitation, a greedy approach will be used to reduce the exit of the device with the lowest exit-to-bandwidth ratio until the bandwidth limitation is met. For example, in FEEL, different devices have different exits and different exit-to-bandwidth ratios. If the total required bandwidth exceeds the bandwidth limitation, the algorithm will reduce the exit point of the IoT devices with the lowest bandwidth utilization efficiency.

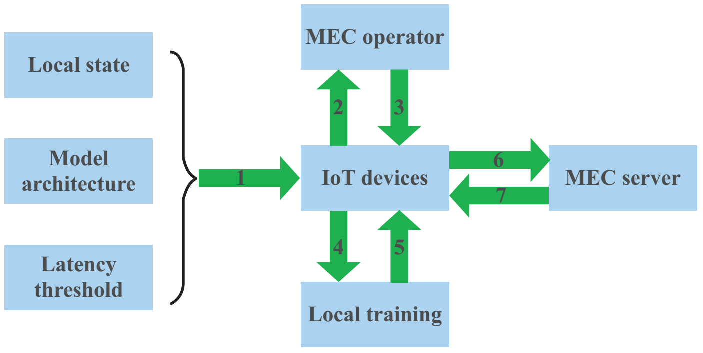


Fig. 4. Architecture of the proposed system.

In detail, as described in Algorithm 3, each device firstly checks whether it can accomplish the training within the latency threshold γ_{th} from the latest exit to the first one (M to 1) by computing $B_k(m)$ according to (21) (lines 2-6). Specifically, $B_k(m) \leq 0$ represents that the device fails to finish the local training, due to the non-positive item of $\gamma_{th} - \frac{\alpha |\mathcal{D}_k| g_1(m)}{C_k}$. In contrast, $B_k(m) > 0$ represents that the device can accomplish the local training within the latency threshold γ_{th}. After each device preliminarily chooses a proper exit, it will report this information to the MEC controller and apply for the required bandwidth (lines 7-10). Then, the MEC controller will play a very important role, which takes charge of global scheduling. In line 13, the MEC controller will integrate the information from different devices and compute the current bandwidth requirement. If the bandwidth constraint $B \leq B_A$ is not satisfied, the edge server will fine-tune the exit of device in a greedy approach.

In further, as shown in lines 14-18, the edge server finds the device $k_{min} = \arg\min_{k \in \mathcal{K}_s} \frac{M_k}{B_k}$ whose exit-to-bandwidth ratio is the minimum, and then sets the exit forward as $M_{k_{min}} = M_{k_{min}} - 1$. In this way, the training latency of device k_{min} is alleviated due to the decreased number of trainable parameter and FLOPs. As well, the bit number to be uploaded is reduced. Hence, the bandwidth demand of this device can be substantially decreased. This process of exit adjustment will repeat until the bandwidth constraint is satisfied.

V. Simulation and Discussion

In this section, we conduct simulations to evaluate the performance of the proposed ME-FEEL. Specifically, we firstly introduce the architecture of the whole ME-FEEL framework, setting and the implementation details, and then we will present the simulation results and discussion.

A. Architecture of the proposed system

The architecture of the proposed system is shown in Fig. 4, consisting of multiple IoT devices, one MEC operator, and one MEC sever on the BS. Before the local training stage, the chosen IoT devices will firstly collect their local state about model architecture and latency threshold (Step 1). Then, the MEC operator executes exit selection and bandwidth allocation after it receives the report from devices (Step 2 and 3). After that, the devices will train the local model with the help of

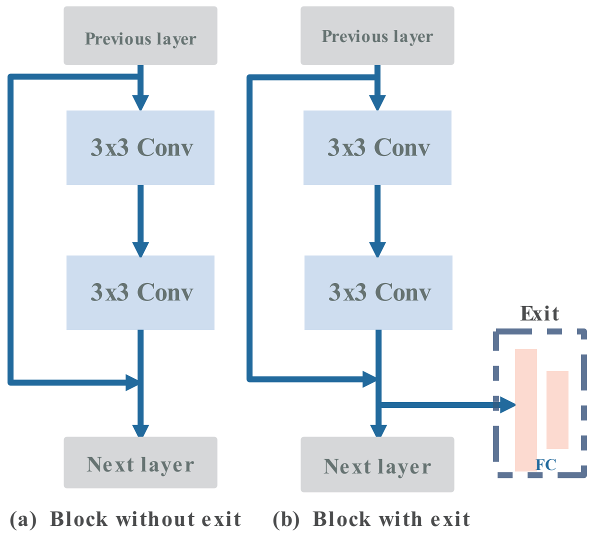


Fig. 5. Block architecture in the ME-ResNet.

TABLE II
MODIFIED RESNET-18 WITH 7 EXITS.

Input	Operator	Channels	Stride	Output	Exit
$32^2 \times 1$	Conv3×3	64	1	$32^2 \times 64$	-
$32^2 \times 64$	Block	64	1	$32^2 \times 64$	-
$32^2 \times 64$	Block	64	1	$32^2 \times 64$	✓
$32^2 \times 64$	Block	128	2	$16^2 \times 128$	✓
$16^2 \times 128$	Block	128	1	$16^2 \times 128$	✓
$16^2 \times 128$	Block	256	2	$8^2 \times 256$	✓
$8^2 \times 256$	Block	256	1	$8^2 \times 256$	✓
$8^2 \times 256$	Block	512	2	$4^2 \times 512$	✓
$4^2 \times 512$	Block	512	1	$4^2 \times 512$	-
$1^2 \times 512$	FC	-	-	10	✓

KD (Step 4 and 5). Finally, the MEC server will aggregate the local updates and broadcast the global update to all devices (Step 6 and 7).

B. Simulation Settings

1) Task and Dataset: We select the dataset of Fashion-MNIST[3] to perform the simulation in our work, as visual intelligence is one of the most widely used technologies in the aforementioned application scenarios, and the Fashion-MNIST dataset is used as the benchmark in the literature of visual intelligence and FL. The dataset has 5000 training samples and 1000 test samples, and they will be divided into 100 parts. The training samples are partitioned with the same non-IID setting [34]. We firstly sort the data by the index of classes and then divide them into 200 groups of size 300, ensuring that only two groups are assigned to one device.

2) Multi-exit FL: We use the classical ResNet-18 [35] due to its outstanding ability of feature extraction. We modify the first layer to make it adapt the input size. Besides, the original building block consists of two continuous 3×3 convolutions and one shortcut. As shown in Fig. 5, we add one exit on the bottom of the block, where the exit includes a fully-connected network to output the prediction. Similarly, 7 exits in total are set in ResNet-18, and the complete model architecture named ME-ResNet is shown in Table II. Notably, necessary activation functions and reshape operations are omitted for simplification. As well, all the batch normalization (BN) layers are replaced by group normalization for improving the performance under non-IID setting. We use Adam optimizer to update weights with a learning rate of 0.001. Batch size C_k is 10, and we set the temperature of KD to 3. The total number of communication rounds is 750, and only the chosen 10 devices will train the local model for 5 times in each round. Another important setting is the mapping $g_1(\cdot)$ and $g_2(\cdot)$, where we run practical measurement on the AMD workstation, and the associated results are presented later.

3) Network environment: As a typical setting in the IoT network, we set up all devices to connect to the base station over LTE cellular networks. The available bandwidth of BS is 40MHz and the transmit power of each device is set to 1W. The coefficient α is randomly generated subjected to a uniform distribution $\mathcal{U}(0, 10)$, which can simulate the heterogeneity in the IoT networks. To simulate the variable channel state, $|h_k|^2$ is subject to the exponential distribution with the average channel gain of unity. It remains the same within one round, while varies between different rounds. The variance of AWGN is set to $1e-3$.

C. Competitive methods

In order to verify the effectiveness of the proposed strategy, we compare it with various methods of FEEL listed below.

- **FEEL-ideal**: The conventional FEEL algorithm in the ideal IoT networks, where no constraint is considered at all.
- **FEEL**: Based on the FEEL-ideal, we incorporate the practical IoT networks under latency and bandwidth constraint. For the limited bandwidth, we apply the bandwidth allocation method introduced in [7], where the MEC controller always allocates the bandwidth firstly to the clients requiring less.
- **FEEL-UB**: Different from the FEEL, the edge server allocates the bandwidth evenly among the devices.

D. Simulation results and discussion

Fig. 6 depicts the local training time versus the index of exit of the original ResNet, ME-ResNet, and ME-ResNet with KD technology (ME-ResNet-KD), where the number of exits is 7 in ME-ResNet, and the batch size is 10. From this figure, we can find that the original ResNet with only one exit requires about 14ms to train a batch of samples. When the exit is the last one, the training time of ME-ResNet-KD and ME-ResNet is 16.9ms and 14.70ms, respectively. However, when we choose some earlier exits, the training time of ME-ResNet and ME-ResNet-KD can be reduced significantly. In particular, compared with the ResNet, ME-ResNet and ME-ResNet-KD can reduce the training time to

[3]This dataset can be found through the link https://github.com/zalandoresearch/fashion-mnist.

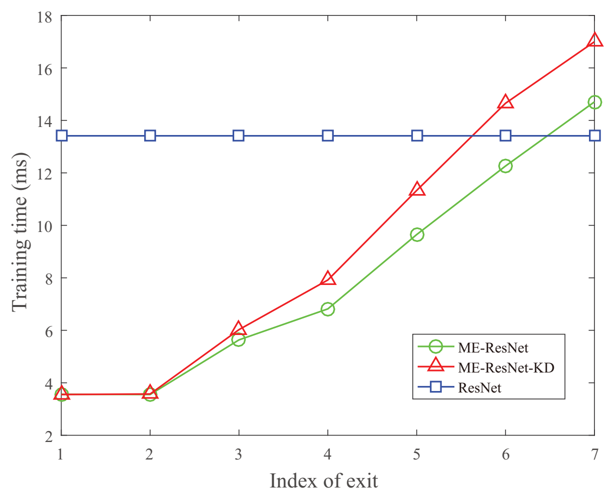


Fig. 6. Local training time versus the index of exit.

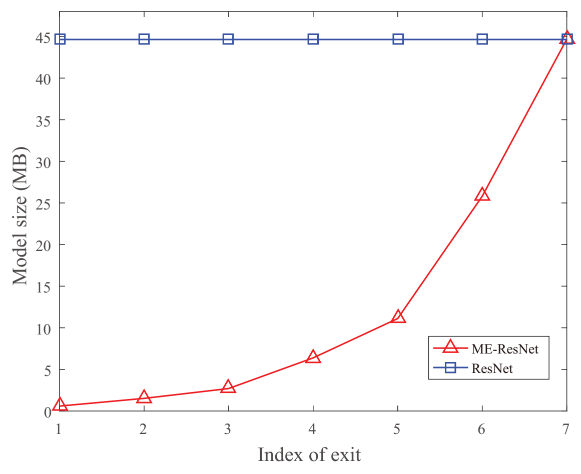


Fig. 7. Model size in MBytes versus the index of exit.

half when the 4-th exit is chosen. Moreover, when the ME-ResNet and ME-ResNet-KD quit on the first exit, they both require only 4ms to finish the training, which reduces about 10ms compared with the original ResNet. Another interesting phenomenon is that when the model quits on the first exit, the KD technology does not increase the latency compared to the ME-ResNet. When the exit changes from 2 to 7, there is at most 1ms added in latency. It should also be noted that although applying the KD technology brings extra latency in the worst case of choosing the last exit compared to the conventional FEEL, , the proposed framework can reduce the complexity in terms of latency and parameters to about 30% and 10%, respectively, for most IoT devices which choose the earlier exits. This indicates that the proposed framework only exerts little extra hardware complexity at a few devices with the powerful computational capacity to obtain a better accuracy and flexibility.

Fig. 7 shows the model size in MBytes versus the index of exit of ME-ResNet and the original ResNet, where the number of exits is 7 in ME-ResNet. We can find from this figure that the mode size of the ResNet is 44.6MB, which is similar to 44.7MB of ME-ResNet when the exit is the last one. However, when the ME-ResNet chooses some earlier exits, its model size decreases drastically. In particular, when each device chooses the first three exits, the associated model sizes are all less than 5MB, which is ten times lower than that of the original ResNet. This can significantly reduce the uploading latency and system communication overhead. These results further demonstrate that adding 7 exits for ResNet does not cause a significant increase in the number of parameters. More importantly, the ME-ResNet can control its size by exiting from different depths, which enables it to reduce the model size as needed.

Fig. 8 illustrates the test accuracy versus the communication round, where the dataset is the Fashion-MNIST under the non-IID setting, the total bandwidth B_A is 40MHz, and the latency threshold γ_{th} is 15s. We also compare the convergence performance of the proposed ME-FEEL with different exits to the convergence of FEEL and FEEL-UB in which only one exit is available. This figure shows that the proposed ME-FEEL outperforms the other methods in both the convergence rate and accuracy. In particular, when the 7-th exit is chosen, the proposed ME-FEEL has a remarkable improvement in the test accuracy compared to FEEL and FEEL-UB. Moreover, the training process of ME-FEEL is more stable than the other methods. In particular, when the 4-th, 5-th, and 6-th exits are chosen, the proposed ME-FEEL still exhibits a clear advantage on the convergence over the other methods. Notably, early exits usually underperform the single exit in FEEL due to the insufficient fitting ability. However, we can find that the early exits in ME-FEEL, such as the first, second, and third one, still perform better than FEEL and FEEL-UB in the experiments, although with a little degradation compared to its later exits. These results show the effectiveness of the proposed ME-FEEL as well as the exit selection and bandwidth algorithm, which can perform better than the conventional methods by aggregating more updates in each communication round. Besides, it verifies that the later exits can transfer knowledge to the earlier ones by the KD technology and help improve the performance.

In addition, we present a detailed performance comparison on the Fashion-MNIST dataset in Table III, where there are five methods, including FEEL-ideal, FEEL, FEEL-UB, ME-FEEL (without KD) and ME-FEEL. We show the average accuracy of each available exit and the maximum accuracy of different methods in this table. Except the FEEL-ideal, the communication bandwidth B_A is set to 40MHz and γ_{th} is set to 15s for the other four methods. From this table, we can find that the FEEL-ideal can achieve a maximum accuracy of 87% under the non-IID setting. Notably, there is not any constraint at all here. However, in the resource-limited environments with aforementioned setting, the performance of the conventional FEEL will be severely degraded. In particular, the FEEL achieves only a best accuracy of 80.8%, which is about 7% worse than that of the FEEL-ideal. Similarly, the FEEL-UB faces a more serious situation, where the accuracy loss is 22%. In contrast, the proposed ME-FEEL achieves a maximum accuracy of 84% without applying the KD technology. These results further verify the effectiveness of the proposed ME-

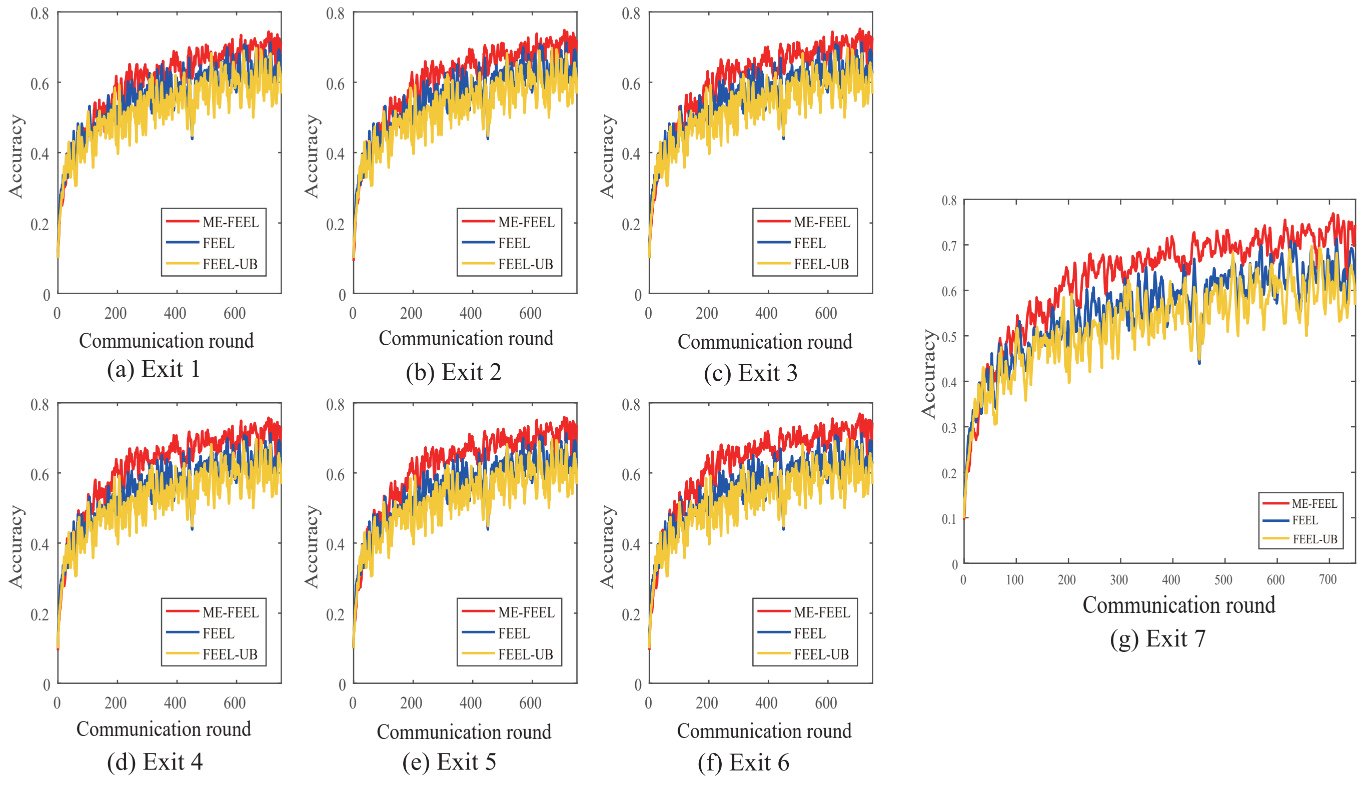


Fig. 8. Test accuracy versus the communication round for the Fashion-MNIST dataset with $\gamma_{th} = 15$s and $B_A = 40$MHz.

TABLE III
ACCURACY COMPARISON OF SEVERAL METHODS WITH $\gamma_{th} = 15$S AND $B_A = 40$MHZ.

	Accuracy							
Methods	Exit 1	Exit 2	Exit 3	Exit 4	Exit 5	Exit 6	Exit 7	Maximum
FEEL-ideal	-	-	-	-	-	-	0.87	0.87
FEEL	-	-	-	-	-	-	0.8079	0.8079
FEEL-UB	-	-	-	-	-		0.6542	0.6542
ME-FEEL (without KD)	0.7535	0.7735	0.7824	0.7975	0.8075	**0.8374**	**0.8174**	**0.8390**
ME-FEEL	**0.8159**	**0.8185**	**0.8178**	**0.8200**	**0.8205**	0.8205	0.8148	0.8286

FEEL framework.

Moreover, we can also find from Table III that there is about 1% accuracy loss when we apply the KD technology in ME-FEEL, compared to the ME-FEEL (without KD). However, the early exits of ME-FEEL, including the exits from 1 to 5, all perform better than those of ME-FEEL (without KD). In particular, the ME-FEEL with the first exit achieves an accuracy gain of 6%, compared to the method without applying the KD technology. Notably, ME-FEEL with the first exit also outperforms the conventional FEEL and FEEL-UB with all exits in terms of a lower cost in computation and model size. Similarly, due to the use of the KD technology, the accuracy improvement of the exits from 2 to 5 are 4.5%, 3.53%, 2.25%, and 1.3%, respectively, compared to the situation of not utilizing the KD technology. This is helpful for the computation-limited devices in the edge-enabled IoT networks. These results further verify the effectiveness of applying the KD technology into the proposed ME-FEEL.

To further verify the robustness of the proposed ME-FEEL, the performance comparison under different latency thresholds[4] is presented in Fig. 9, where the Fashion-MNIST dataset is used, the total communication bandwidth B_A is set to 40MHz, and the latency threshold γ_{th} varies from 12s to 15s. We can observe from this figure that the performances of different methods are improved with a larger γ_{th}. In detail, under the loose latency thresholds such as 21s, the proposed ME-FEEL achieves the accuracy gain of 0.5% and 5.0% compared to those of the FEEL and FEEL-UB, respectively. Moreover, the performance gap enlarges along with the decreased latency threshold. For example, when $\gamma_{th} = 15$s and $\gamma_{th} = 12$s, compared with the FEEL, the accuracy gain achieved by the proposed ME-FEEL is 1.56% and 2.68%, respectively. In addition, the proposed ME-FEEL can obtain the accuracy gain

[4]When the latency threshold is set too large, the server needs to wait for the update from the bottleneck device and the system efficiency decreases. In contrast, when the latency threshold is small, few devices can upload updates successfully, which deteriorates the system accuracy. By taking into account these two aspects, we set the latency threshold in the range of 12-21s.

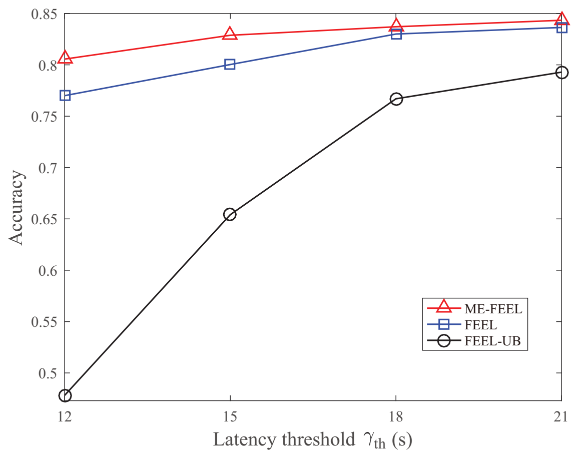


Fig. 9. Test accuracy versus the latency threshold γ_{th} for the Fashion-MNIST dataset with $B_A = 40$MHz.

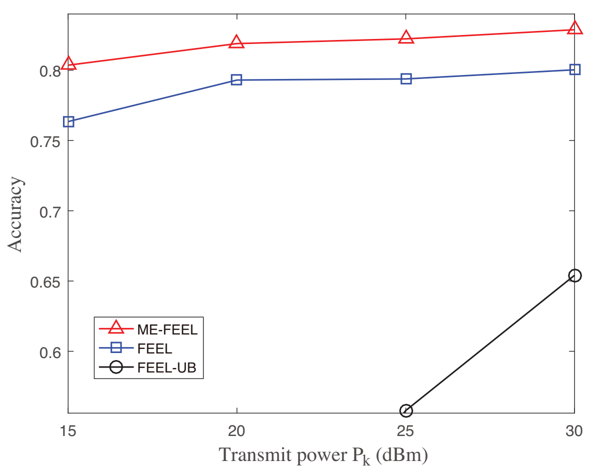


Fig. 11. Test accuracy versus the transmit power P_k for the Fashion-MNIST dataset with latency threshold $\gamma_{th} = 15$s.

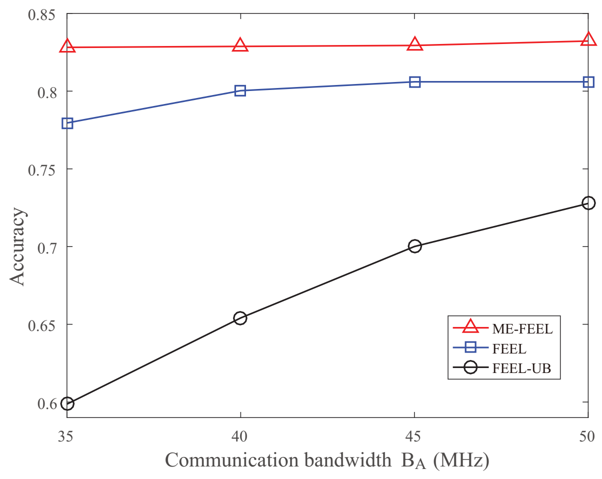


Fig. 10. Test accuracy versus the total communication bandwidth B_A for the Fashion-MNIST dataset with latency threshold $\gamma_{th} = 15$s.

up to 16.6% and 32.7% compared to the FEEL-UB. These results indicate that the proposed ME-FEEL can adapt well to the practical edge-enabled IoT environments by reasonable exit selection and bandwidth allocation.

Fig. 10 shows the accuracy performance versus the communication bandwidth B_A, where the Fashion-MNIST dataset is used, the latency threshold γ_{th} is set to 15s, and B_A varies from 35MHz to 50MHz. From this figure, we can find that the accuracy performances of all methods deteriorate along with the decreased B_A. However, the performance degradation exhibits different sensitiveness to the decreased bandwidth. In particular, the accuracy of FEEL-UB deteriorates from 72.7% to 59.9%, when the bandwidth varies from 50MHz to 35MHz, indicating that the FEEL-UB is significantly affected by the decrease of total communication bandwidth. The FEEL performs a little bit better than FEEL-UB, while its accuracy performance is still limited by the bandwidth, where the accuracy deteriorates from 80.6% to 77.9% when B_A changes from 50MHz to 35MHz. Notably, the proposed ME-FEEL outperforms the other methods, and it has a relatively minor performance degradation. In particular, when $B_A = 50$MHz, the accuracy gain of ME-FEEL is 2.6% and 10.5% compared to FEEL and FEEL-UB, respectively. Moreover, the accuracy gain increases when the communication bandwidth decreases, and it is up to 20.75%. This is because that the proposed framework can reduce the exit number of devices whose computational power is limited, which can reduce the training latency. In this way, the proposed framework can mitigate some dependence on the bandwidth resources. In contrast, the competitive methods fail to do that, and the accuracy performance deteriorates severely when the available bandwidth is insufficient.

Fig. 11 demonstrates the test accuracy versus the transmit power P_k varying from 15dBm to 30dBm, where the latency threshold is 15s, and the total communication bandwidth B_A is 40MHz. From this figure, we can find that the accuracy performances of all methods improve with a larger transmit power. In particular, when $P_k = 30$dBm, the accuracy of the proposed ME-FEEL, FEEL, and FEEL-UB is 82.88%, 80.03% and 65.41%, respectively. When the transmit power decreases, the accuracy of FEEL-UB deteriorates significantly, and it almost fails to obtain enough accuracy performance. The accuracy of FEEL deteriorates from 80.03% to 76.35% when the transmit power reduces to 15dBm. In contrast, the proposed framework shows its robustness, and there is only 2% decrease in the test accuracy in the same situation. These results further indicate that the proposed framework can surmount the limited-resource environments, and meet the demands of practical edge-enabled IoT networks.

VI. Conclusions

In this paper, we investigated the problem of facilitating an efficient and flexible FEEL in the edge-enabled industrial IoT networks. We proposed a novel FL framework named FEEL to tackle the system heterogeneity problem. The proposed ME-FEEL employed multiple exits on the local model so that we could avoid training the complete model under the latency constraint. In this case, the devices were able to choose the best exit and train a specific part of the model according to their needs. Since the early exits may cause some performance degradation, we applied the KD technology to solve the problem. Moreover, we proposed a joint exit selection and bandwidth allocation algorithm based on the greedy approach to maximize the expected number of exits in each communication round. Finally, we conducted simulations to evaluate the performance of the proposed ME-FEEL by employing the Fashion-MNIST dataset with non-IID setting. Simulation results showed that the proposed ME-FEEL could outperform the conventional FEEL in the resource-limited IoT networks.

Nevertheless, there are still some challenges regarding the deployment of computational intelligence and federated learning in the edge-enabled industrial IoT networks. For example, how to set the latency threshold according to the available resources in time-varying environments is a critical challenge. Besides, how to use the huge number of unlabeled samples in the IoT devices and how to train a shared model when the data is presented as a stream still remain challenging. Moreover, a more efficient distributed inference protocol needs to be devised to further reduce the latency and energy consumption.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] 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, 2016.

[3] S. Tang, "Dilated convolution based CSI feedback compression for massive MIMO systems," *IEEE Trans. Vehic. Tech.*, vol. 71, no. 5, pp. 211–216, 2022.

[4] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," *arXiv:2003.12172*, 2020.

[5] Y. Qian, "Edge intelligence for next generation wireless networks," *IEEE Wirel. Commun.*, vol. 28, no. 2, pp. 2–3, 2021.

[6] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 2, pp. 1188–1200, 2020.

[7] Z. Zhao, J. Xia, L. Fan, X. Lei, G. K. Karagiannidis, and A. Nallanathan, "System optimization of federated learning networks with a constrained latency," *IEEE Trans. Veh. Technol.*, vol. 71, no. 1, pp. 1095–1100, 2021.

[8] S. Tang, W. Zhou, L. Chen, L. Lai, J. Xia, and L. Fan, "Battery-constrained federated edge learning in UAV-enabled IoT for B5G/6G networks," *Phys. Commun.*, vol. 47, p. 101381, 2021.

[9] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE International Conference on Communications, ICC*, 2019, pp. 1–7.

[10] M. Phuong and C. Lampert, "Distillation-based training for multi-exit architectures," in *IEEE International Conference on Computer Vision, ICCV*, 2019, pp. 1355–1364.

[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, 2015.

[12] S. Zhu, W. Xu, L. Fan, K. Wang, and G. K. Karagiannidis, "A novel cross entropy approach for offloading learning in mobile edge computing," *IEEE Wirel. Commun. Lett.*, vol. 9, no. 3, pp. 402–405, 2020.

[13] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks," *IEEE Trans. Ind. Inform.*, vol. 16, no. 8, pp. 5424–5434, 2020.

[14] J. Xia, L. Fan, N. Yang, Y. Deng, T. Q. Duong, G. K. Karagiannidis, and A. Nallanathan, "Opportunistic access point selection for mobile edge computing networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 1, pp. 695–709, 2021.

[15] X. Lai, "Outdated access point selection for mobile edge computing with cochannel interference," *IEEE Trans. Vehic. Tech.*, vol. PP, no. 99, pp. 1–12, 2022.

[16] X. Li, M. Zhao, M. Zeng, S. Mumtaz, V. G. Menon, Z. Ding, and O. A. Dobre, "Hardware impaired ambient backscatter NOMA systems: Reliability and security," *IEEE Trans. Commun.*, vol. 69, no. 4, pp. 2723–2736, 2021.

[17] X. Li, Z. Xie, Z. Chu, V. G. Menon, S. Mumtaz, and J. Zhang, "Exploiting benefits of IRS in wireless powered NOMA networks," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 175–186, 2022.

[18] W. Zhou, L. Xing, J. Xia, L. Fan, and A. Nallanathan, "Dynamic computation offloading for MIMO mobile edge computing systems with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5172–5177, 2021.

[19] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7457–7469, 2020.

[20] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wirel. Commun.*, vol. 25, no. 3, pp. 116–122, 2018.

[21] C. Li, J. Xia, F. Liu, D. Li, L. Fan, G. K. Karagiannidis, and A. Nallanathan, "Dynamic offloading for multiuser muti-CAP MEC networks: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2922–2927, 2021.

[22] Y. Guo and S. Lai, "Distributed machine learning for multiuser mobile edge computing systems," *IEEE J. Sel. Top. Signal Process.*, vol. PP, no. 99, pp. 1–12, 2021.

[23] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," in *International Conference on Learning Representations, ICLR*, 2016.

[24] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE International Conference on Computer Vision, CVPR*, 2019, pp. 1314–1324.

[25] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations, ICLR*, 2016.

[26] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. N. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS*, 2017, pp. 615–629.

[27] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "Deepthings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, 2018.

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

[29] S. Zheng, C. Shen, and X. Chen, "Design and analysis of uplink and downlink communications for federated learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2150–2167, 2021.

[30] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *IEEE International Parallel and Distributed Processing Symposium, IPDPS*, 2020, pp. 234–243.

[31] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3606–3621, 2021.

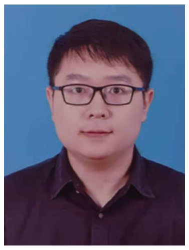
[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv:1503.02531*, 2015.

[33] A. G. Myasnikov, A. Nikolaev, and A. Ushakov, "Knapsack problems in groups," *Math. Comput.*, vol. 84, no. 292, pp. 987–1016, 2015.

[34] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2017, pp. 1273–1282.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR*, 2016, pp. 770–778.

Shunpu Tang received the B.E. degree in 2020 and he is currently pursuing the master degree, both with the School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China. His current research interests include edge computing, content cache and deep learning system in wireless networks.



Lunyuan Chen received the bachelor's degree in Communication Engineering from Xidian university in 2019. He is currently pursuing the master¡'s degree with the school of Electronics and Communication Engineering, Guangzhou University. His current research interests focus on statistical machine learning and deep learning.

Ke He received the bachelor's degree from the Wuhan University of Technology, in 2015. He is currently pursuing the master's degree with the School of Computer Science, Guangzhou University. His current research interests include signal processing and artificial intelligence.

Junjuan Xia received the bachelor degree from the Department of Computer Science, Tianjin University in 2003, and obtained the master degree from the Department of Electronic Engineering, Shantou University in 2015. Now she is working for the School of Computer Science, Guangzhou University. Her current research interests include wireless caching, IoT networks, physical-layer security, and cooperative relaying.

Lisheng Fan received the bachelor and master degrees from Fudan University and Tsinghua University, China, in 2002 and 2005, respectively, both from the Department of Electronic Engineering. He received the Ph.D degree from the Department of Communications and Integrated Systems of Tokyo Institute of Technology, Japan, in 2008. He is now a Professor with the School of Computer Science, GuangZhou University. His research interests span in the areas of wireless cooperative communications, physical-layer secure communications, intelligent communications, and system performance evaluation. Lisheng Fan has published many papers in international journals such as IEEE Transactions on Wireless Communications, IEEE Transactions on Communications, IEEE Transactions on Information Theory, as well as papers in conferences such as IEEE ICC, IEEE Globecom, and IEEE WCNC. He is an editor of China Communications, and has served as a guest editor for many journals such as Physical Communication, EURASIP Journal on Wireless Communications and Networking, Wireless Communications and Mobile Computing, and so on. He has been awarded as Exemplary Reviewer by IEEE Transactions on Communications and IEEE Communications Letters.

Arumugam Nallanathan (S'97-M'00-SM'05-F'17) is Professor of Wireless Communications and Head of the Communication Systems Research (CSR) group in the School of Electronic Engineering and Computer Science at Queen Mary University of London since September 2017. He was with the Department of Informatics at King's College London from December 2007 to August 2017, where he was Professor of Wireless Communications from April 2013 to August 2017 and a Visiting Professor from September 2017. He was an Assistant Professor in the Department of Electrical and Computer Engineering, National University of Singapore from August 2000 to December 2007. His research interests include 5G Wireless Networks, Internet of Things (IoT) and Molecular Communications. He published nearly 500 technical papers in scientific journals and international conferences. He is a co-recipient of the Best Paper Awards presented at the IEEE International Conference on Communications 2016 (ICC'2016) , IEEE Global Communications Conference 2017 (GLOBECOM'2017) and IEEE Vehicular Technology Conference 2018 (VTC'2018). He is an IEEE Distinguished Lecturer. He has been selected as a Web of Science Highly Cited Researcher in 2016.

He is an Editor for IEEE Transactions on Communications. He was an Editor for IEEE Transactions on Wireless Communications (2006-2011), IEEE Transactions on Vehicular Technology (2006-2017), IEEE Wireless Communications Letters and IEEE Signal Processing Letters. He served as the Chair for the Signal Processing and Communication Electronics Technical Committee of IEEE Communications Society and Technical Program Chair and member of Technical Program Committees in numerous IEEE conferences. He received the IEEE Communications Society SPCE outstanding service award 2012 and IEEE Communications Society RCC outstanding service award 2014.