Queries to the Author

When you submit your corrections, please either annotate the IEEE Proof PDF or send a list of corrections. Do not send new source files as we do not reconvert them at this production stage.

Authors: Carefully check the page proofs (and coordinate with all authors); additional changes or updates WILL NOT be accepted after the article is published online/print in its final form. Please check author names and affiliations, funding, as well as the overall article for any errors prior to sending in your author proof corrections. Your article has been peer reviewed, accepted as final, and sent in to IEEE. No text changes have been made to the main part of the article as dictated by the editorial level of service for your publication.

Per IEEE policy, one complimentary proof will be sent to only the Corresponding Author. The Corresponding Author is responsible for uploading one set of corrected proofs to the IEEE Author Gateway

Q1. Please confirm or add details for any funding or financial support for the research of this article.
Q2. Please provide the last accessed date for Refs. [39], [40], [41], [42], and [43].

Please see the notes below.

Thank you very much for your reminder, it has been confirmed.

Towards Diversified IoT Image Recognition Services in Mobile Edge Computing

Chuntao Ding, Ao Zhou, Xiao Ma, Ning Zhang, *Member, IEEE*, Ching-Hsien Hsu, *Senior Member, IEEE*, and Shangguang Wang, *Senior Member, IEEE*

Abstract—With the rapid development of the Internet of Things (IoT) and emerging Mobile Edge Computing (MEC) technologies, various IoT image recognition services are revolutionizing our lives by providing diverse cognitive assistance. However, most existing related approaches are difficult to meet the diversified needs of users because they believe that the MEC platform is a single layer. In addition, due to the mutual interference between the data, it is not easy for them to extract the discriminative features (DFs) necessary to analyze the input data. To this end, this article proposes an IoT image recognition services framework for different needs in the MEC environment, which consists of Hierarchical Discriminative Feature Extraction (HDFE) and Sub-extractor Deployment (Sub-ED) algorithms. We first propose HDFE, which can avoid mutual interference between data by separately optimizing the data structure, thereby generating an extractor that extracts effective DFs. Then there is Sub-ED, which divides the extractor into a series of sub-extractors and deploys them on appropriate MEC platforms. By doing so, the IoT device can connect to the corresponding MEC platform according to its service types, and use the sub-extract to extract DFs. Then, the MEC platform uploads the extracted feature data to the cloud server for further processing, e.g., feature matching. Finally, the cloud server sends the processed result back to the IoT device. Experimental results show that compared with the state-of-the-art approaches, the proposed framework improves recognition accuracy by about 6% and reduces network traffic by up to 94%.

Index Terms—Internet of Things, mobile edge computing, IoT services, discriminative features

1 Introduction

1.1 Motivation & Problem Formulation

Internet of Things (IoT) image recognition services are revolutionizing our lives by providing diverse cognitive assistance [1], [2], [3]. Most related IoT services are based on the Mobile Cloud Computing (MCC) [4], [5], [6]. Their process is as follows: IoT devices first use pre-processing algorithms (e.g., Principal Component Analysis (PCA) [7], [8] and Haar [9]) to extract features from the captured data. Then, the extracted feature data is sent to the cloud server for further processing. Finally, the cloud server sends the processed results to IoT devices. The main limitation of this solution is that the distance between the IoT device and the cloud server is long (e.g., multiple hops), resulting in a long delay. Mobile Edge Computing (MEC) [10], [11], [12], [13], [14] can solve the long delay by deploying the MEC platform with computing and storage resources close to the IoT device.

This paper studies the problem of IoT image recognition services in the MEC environment. Fig. 1 illustrates the system architecture consisting of an end layer, an edge layer, and a cloud layer. The end layer includes some IoT devices, such as smartphones, driverless cars, and drones. The edge layer includes three MEC platforms, namely the access-level MEC platform, the district-level MEC platform, and the metro-regional-level MEC platform [15], [16]. The cloud layer includes a large number of cloud servers. IoT devices communicate with the MEC platform through base stations and connect to the base station through 5G or Super WiFi [17]. The base station connects the cloud servers through the Internet backbone. We assume that the image data set $(\mathbf{x}_i, y_i)_{i=1}^{N}$ is stored on the cloud servers, where N is the number of image data, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^c$, d is the dimension of $\mathbf{x}_i$, and c is the number of classes. The workflow is as follows: The IoT device first captures the image data and then uploads it to the edge server for pre-processing. Then, the edge server uploads the preprocesssed image data to the cloud server for further processing. Finally, the cloud server sends the processed results back to the IoT device.

1.2 Limitations of Prior Art

Many related approaches [18], [19], [20] have achieved excellent results in providing IoT image recognition

- *Chuntao Ding is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China. E-mail: chuntaoding@163.com.*
- *Ao Zhou and Xiao Ma are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {aozhou, maxiao18}@bupt.edu.cn.*
- *Ning Zhang is with the Department of Computing Sciences, Texas A&M University at Corpus Christi, Corpus Christi, TX 78412 USA. E-mail: ning.zhang@tamucc.edu.*
- *Ching-Hsien Hsu is with the Department of Computer Science and Information Engineering, Asia University, Taichung 41354, Taiwan, and with the Department of Medical Research, China Medical University Hospital, China Medical University, Taichung 406040, Taiwan, and also with the School of Mathematics and Big Data, Foshan University, Foshan 528000, China. E-mail: robertchh@gmail.com.*
- *Shangguang Wang is with the Beiyou Shenzhen Research Institute, Shenzhen 518172, China. E-mail: sgwang@bupt.edu.cn.*

Manuscript received 25 June 2020; revised 10 June 2021; accepted 18 Aug. 2021. Date of publication 0 . 0000; date of current version 0 . 0000. (Corresponding author: Shangguang Wang.) Recommended for acceptance by A. Boukerche. Digital Object Identifier no. 10.1109/TCC.2021.3109385

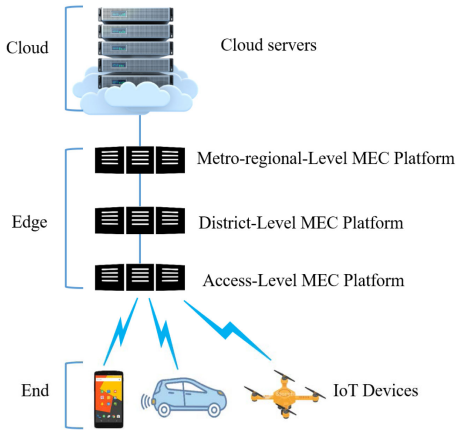


Fig. 1. The system architecture.

services. However, they are usually not easy to adapt to the hierarchical structure of the MEC platform to meet the diverse demands of IoT image recognition services, because they treat the MEC platform as a single layer. In addition, because of mutual interference between data when generating extractors, it is not easy for them to generate extractors that can extract effectively discriminative features.

1.3 Proposed Approach

In this paper, we propose an IoT image recognition services framework for different needs in MEC, where the edge layer consists of three MEC platforms. In the proposed framework, we first propose Hierarchical Discriminative Feature Extraction (HDFE) algorithm to generate an extractor $\mathbf{E}$ on the cloud server. Follow that, we propose a Sub-extractor Deployment (Sub-ED) algorithm to divide $\mathbf{E}$ into sub-extractors, i.e., $\mathbf{E}1$, $\mathbf{E}2$, and deploy them on appropriate MEC platforms. Then, we use $\mathbf{E}$ and these sub-extractors to extract different levels of Discriminative Features (DFs) from the data set on the cloud servers and form DFs sets, i.e., $\mathbf{E}^T\mathbf{X}$, $\mathbf{E}1^T\mathbf{X}$ and $\mathbf{E}2^T\mathbf{X}$. When receiving the image data (e.g., $\mathbf{x}$) from IoT devices, the corresponding MEC platform uses the deployed sub-extractor (e.g., $\mathbf{E}1$) to extract DFs (e.g., $\mathbf{E}1^T\mathbf{x}$) from the image data. Then, the MEC platform uploads the extracted DFs data to the cloud server for processing. Finally, the cloud servers send the processed results to IoT devices. Our goal is to deploy different extractors on the hierarchical MEC platforms to satisfy various IoT image recognition services in terms of accuracy and response time.

1.4 Challenges and Proposed Solutions

The first key challenge is how to generate an extractor that can extract effective DFs. DFs are important because they determine the performance of IoT image recognition services, e.g., the recognition accuracy is a very important indicator. In addition, DFs affect network traffic from MEC platforms to cloud servers as well as network transmission time and feature matching time on the cloud servers. Effective features refer to a small number of features that can achieve high recognition accuracy. However, it is challenging to extract effective DFs in MEC platforms. There are two reasons. (i) Generating an extractor to extract DFs requires the use of the label information of the data set. However, there is no such information in the MEC platform. (ii) Existing algorithms simultaneously optimize the structure of data, which leads to mutual interference, making it difficult to generate extractors that can extract effective DFs.

To solve this challenge, we propose the HDFE algorithm to generate an extractor on the cloud servers. Since the extraction process of the extractor uses the label information of the data, the extractor can extract the DFs. To avoid mutual interference between data, the HDFE explores the structure information of the data hierarchically. In the HDFE algorithm, there are two ways to analyze the structural information of the data set. The first is to first minimize the intra-class structure, then maximize the inter-class structure and name it HDFE1. The second is to first maximize the inter-class structure, then minimize the intra-class structure and name it HDFE2. Both HDFE1 and HDFE2 can generate extractors that extract effective DFs from the data set by avoiding mutual interference between data. In addition, we also conclude through discussions and experiments that HDFE1 is better in terms of recognition accuracy.

The second key challenge is how to divide the extractor $\mathbf{E}$ *into sub-extractors to meet the diverse demands of IoT devices*. In real-world scenarios, multiple layers of MEC platforms are deployed between IoT devices and cloud servers. These different levels of MEC platforms are targeted at different applications. For example, the access-level MEC platform is suitable for applications that are slow to move, even not mobile and latency-sensitive. The district-level MEC platform is primarily intended for fast-moving and latency-sensitive applications. One solution is to deploy different sub-extractors on MEC platforms to extract DFs of different sizes. However, it is challenging to divide $\mathbf{E}$ into sub-extractors and deploy them on appropriate MEC platforms.

To address this challenge, we propose the Sub-ED algorithm. In the Sub-ED algorithm, we analyze the nature of the extractor $\mathbf{E}$ and divide it into two sub-extractors, $\mathbf{E}1$ and $\mathbf{E}2$. Note that, $\mathbf{E}1 \subset \mathbf{E}2 \subset \mathbf{E}$. For instance, since access-level and district-level MEC platforms requires to provide fast response for IoT services, we place $\mathbf{E}1$ on them to extract a small number of DFs, at the expense of recognition accuracy in exchange for fast response. The metro-regional-level MEC platform requires to consider both response time and recognition accuracy, we can place $\mathbf{E}2$ on it. In addition, we deploy $\mathbf{E}$ on the cloud servers, if we require the highest recognition accuracy, we can use the $\mathbf{E}$ to extract the most effective DFs. Thus, we can meet the diverse demands in terms of recognition accuracy and response time.

1.5 Novelty and Advantages Over Prior Art

The technique novelty of this paper is to propose the IoT services for different needs framework, which consists of HDFE and Sub-DE algorithms. The key technical depth of this paper is to generate an extractor $\mathbf{E}$, and divide $\mathbf{E}$ into sub-extractors to adapt the hierarchical structure of the MEC architecture. The key advantages of the proposed framework over the previous approaches are two-fold: (i) It has a higher accuracy because it can obtain an extractor $\mathbf{E}$ that can extract more effective DFs. (ii) It can meet diverse demands of IoT image recognition services. Experimental results show that the proposed framework improves

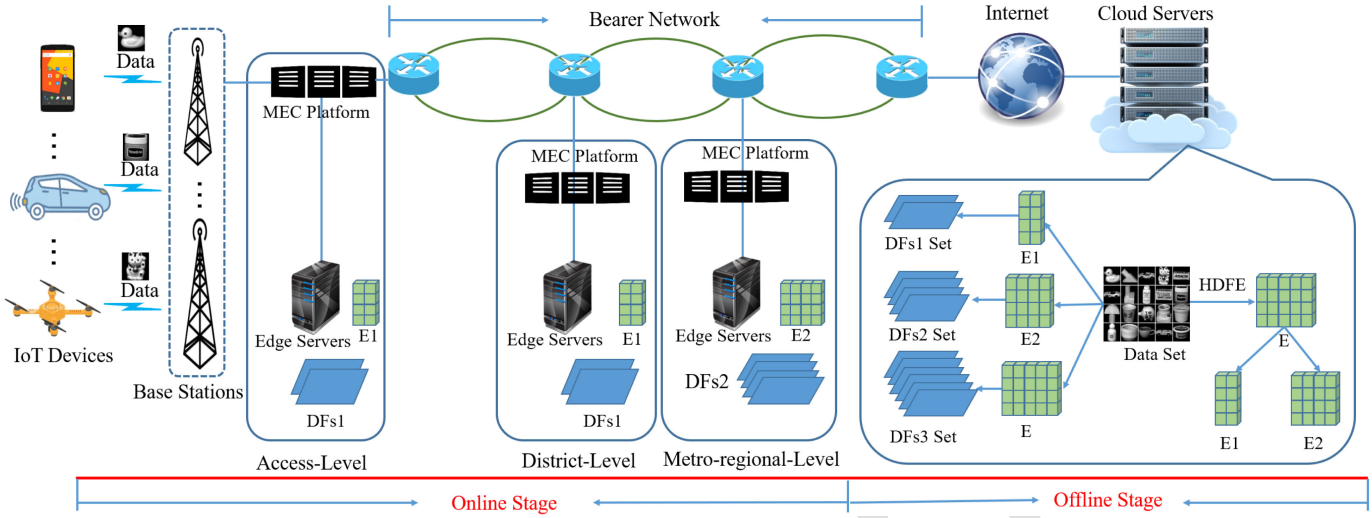


Fig. 2. The proposed framework overview.

accuracy by about 6% and reduces the network traffic by up to 94% compared with the state-of-the-art approaches.

The following paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the design of the proposed framework. Section 4 provides an experimental evaluation. Section 5 concludes and outlines future work.

2 Related Work

2.1 IoT Image Recognition Services

IoT image recognition services refers to the use of IoT devices to provide users with various image recognition services. Due to the limited computing and storage resources of IoT devices, most services delivery methods are usually based on MCC [21] and MEC [22], [23]. That is, when capturing the the data, the IoT device first uses feature extraction algorithms [24], [25] to pre-process it or sends the raw image data to the MEC platform to pre-process it [18]. Then, the IoT device or the MEC platform sends the pre-processed image data to the cloud server. After receiving the pre-processed image data, the cloud server performs feature matching algorithms to obtain results and sends the results back to the IoT device. For example, Li *et al.* [19] upload downscaled image data to the cloud server to reduce the corresponding overhead of data transfer. Hu *et al.* [18] use the local binary patterns [26] to extract features on the edge server before uploading the raw image data to the cloud server to reduce network traffic. Drolia *et al.* [27] use the caching model along with novel optimization to minimize latency by adaptively balancing load between the edge and the cloud. Wang *et al.* [20] send the raw image data to the edge server for pre-processing. Then, the edge server sends the pre-processed data to the cloud server for processing.

2.2 Discriminative Feature Extraction Algorithms

Effective Discriminative Features (DFs) refer to features that enable the sample to be well distinguished from other samples. In recognition services, effective DFs are important because they determine recognition accuracy. In addition, they affect network traffic from IoT devices to the cloud server and feature matching time on the cloud server. In order to obtain effective DFs, a large number of related algorithms have been proposed in recent years [20], [28], [29], [30], [31], [32], [33], [34], [35], [36]. Motivated by [20], [28], [34], [36], there are four structures in the image data set, the global intra-class structure, the global inter-class structure, the local intra-class structure, and the local inter-class structure. However, most of these algorithms either include some of structures (e.g., linear discriminant analysis [28], marginal fisher analysis [34], discriminant neighborhood embedding [33] and double adjacency graphs-based discriminant neighborhood embedding (DAGDNE) [35]) or contain all of structures but optimize them at the same time (e.g., joint global and local structure discriminant analysis (JGLDA) [36] and weight-adaptive projection matrix learning (WAPL) [20]). Algorithms that contain only a subset of structures can affect the capabilities of the generated extractor because some of the structural information is lost. In addition, algorithms that optimize these structures at the same time cause insufficient optimization of each structure and affect the ability of the generated extractor to extract DFs.

3 Design of the Proposed Framework

3.1 Overview

Fig. 2 shows the overview of the proposed framework, which is split into an offline stage and an online stage.

In the offline stage, we first develop the HDFE algorithm to generate the extractor $\mathbf{E}$ on cloud servers. Then, we propose the Sub-DE algorithm to divide extractor $\mathbf{E}$ into sub-extractors $\mathbf{E}1$ and $\mathbf{E}2$. Note that, $\mathbf{E}1 \subset \mathbf{E}2 \subset \mathbf{E}$. Follow that, we use $\mathbf{E}1$, $\mathbf{E}2$, and $\mathbf{E}$ to extract different levels of DFs from the data set and form DFs sets, i.e., DFs1 set, Fs2 set, and DFs3 set. Finally, we deploy $\mathbf{E}1$, $\mathbf{E}2$ and $\mathbf{E}$ on appropriate MEC platforms.

In the online stage, IoT devices first capture the image data and upload it to the corresponding MEC platform. Then, the MEC platform pre-processes the image data [18], such as object detection, grayscale, and alignment. Follow that, the MEC platform uses the extractor (e.g., $\mathbf{E}1$) to extract DFs (e.g., DFs1) from the pre-processed data and sends the data to cloud servers. After receiving the extracted data, the cloud servers match the DFs (e.g., DFs1) with DFs

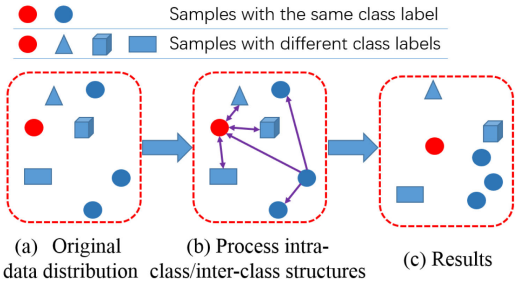


Fig. 3. An illustration of optimizing intra-class and inter-class structures at the same time.

set (e.g., DFs1 set) and finds the most similar DFs with the extracted DFs from EDFs1 set. Finally, cloud servers return the label information of the most similar DFs to IoT devices.

According to China Unicom's MEC platform, the edge layer is divided into three MEC platforms: access-level MEC platform, district-level MEC platform and metro-regional-level MEC platform [15], [16], [37]. The access-level MEC platform is close to the base station and is suitable for massive machine type communication (mMTC) applications, which means that it needs to support a large number of devices in a small area, such as Internet of Things (IoT) use cases. Compared with the access-level MEC platform, the coverage of the MEC platform at the district-level is increased. The district-level MEC platform is suitable for ultra-reliable low-latency communication (uRLLC) applications, which is used for fast-moving and latency-sensitive applications. The convergence-level MEC platform can cover the areas of a city. The metro-regional-level MEC platform is primarily used for enhanced mobile broadband (eMBB) applications for data-driven use cases that require high data rates across a wide coverage area.

3.2 The HDFE Algorithm

The goal of the HDFE algorithm is to generate an extractor $\mathbf{E}$ that can extract effective DFs by exploring the structural information of the data set. $\mathbf{E}$ aims to extract DFs from the data set on the cloud server and the uploaded data on the edge server. DFs indicate that features from samples of the same class label are compacted together, and features from samples of different class labels are separated. However, most existing algorithms have difficulty obtaining DFs because they simultaneously optimizing the structural information of the data set, causing the data to interfere with each other. Fig. 3 shows the process of optimizing intra-class and inter-class structures simultaneously. The red circle wants to "push" away samples of a different class, i.e., the blue triangle, the blue cube and the blue rectangle. At the same time, the sample with the same class label (i.e., the blue circle) wants to "pull" the red circle, as shown in Fig. 3b. When intra-class and inter-class structures are optimized simultaneously, samples interfere with each other and affect the performance of the extracted DFs, as shown in Fig. 3c.

To solve this problem, we propose the HDFE algorithm, which hierarchically explores the structural information of the data set to avoid the interference between samples. We first explore intra-class and inter-class structures as follows.

3.2.1 Exploring the Intra-Class Structure

The intra-class structure includes global intra-class sub-structure S_{gw} and local intra-class sub-structure S_{lw}. The global intra-class sub-structure S_{gw} indicates the relationship between the i-th sample in class m, $\mathbf{x}_i^m$, and the mean of the samples in class m, μ^m, which can be quantified as

$$S_{gw} = \sum_{m=1}^{c}\sum_{i=1}^{N_m} \mathbf{E}_{intra1}^T(\mathbf{x}_i^m - \mu^m)(\mathbf{x}_i^m - \mu^m)^T\mathbf{E}_{intra1}, \quad (1)$$

where N_m is the number of samples in class m, $\mathbf{E}_{intra1}$ is the extractor. The local intra-class sub-structure S_{lw} indicates the pairwise relationship between the samples with the same class label, which can be quantified as

$$\begin{aligned} S_{lw} &= \frac{1}{2}\sum_{ij} ||\mathbf{E}_{intra1}^T\mathbf{x}_i - \mathbf{E}_{intra1}^T\mathbf{x}_j||^2 W_{ij}^w \\ &= tr\{\mathbf{E}_{intra1}^T\mathbf{X}(\mathbf{D}^w - \mathbf{W}^w)\mathbf{X}^T\mathbf{E}_{intra1}\} \end{aligned}, \quad (2)$$

where $W_{ij}^w = 1$, if and only if $i \in \upsilon_{k_1}^w(j)$ or $j \in \upsilon_{k_1}^w(i)$; otherwise, $W_{ij}^w = 0$. $\upsilon_{k_1}^w(i)$ denotes the index set of the k_1 nearest neighbors of sample $\mathbf{x}_i$ with the same class label. $\mathbf{D}^w$ is a diagonal matrix, i.e., $D_{ii}^w = \sum_j W_{ij}^w$.

The global and local intra-class sub-structures help to generate the extractor $\mathbf{E}_{intra1}$. In addition, the global and local intra-class sub-structures have different contributions to the generation of the extractor. To this end, we introduce a parameter $\alpha \in [0, 1]$ to tradeoff global and local intra-class sub-structures. The intra-class structure is designed to generate an extractor to extract DFs from samples with the same class label. Therefore, we minimize the extracted DFs and quantify the objective function of the intra-class structure as

$$\min_{\mathbf{E}_{intra1}} \alpha S_{gw} + (1-\alpha)S_{lw} \quad s.t.\ \mathbf{E}_{intra1}^T\mathbf{E}_{intra1} = \mathbf{I}. \quad (3)$$

3.2.2 Exploring the Inter-Class Structure

The inter-class structure includes global inter-class sub-structure S_{gb} and local inter-class sub-structure S_{lb}. The global inter-class sub-structure S_{gb} indicates the relationship between μ^m and the mean of all samples μ, which can be quantified as

$$S_{gb} = \sum_{m=1}^{c} N_m\mathbf{E}_{inter1}^T(\mu^m - \mu)(\mu^m - \mu)^T\mathbf{E}_{inter1}, \quad (4)$$

where $\mathbf{E}_{inter1}$ is the extractor. The local inter-class sub-structure S_{lb} indicates the pairwise relationship between the samples with different labels, which can be quantified as

$$\begin{aligned} S_{lb} &= \frac{1}{2}\sum_{ij} ||\mathbf{E}_{inter1}^T\mathbf{x}_i - \mathbf{E}_{inter1}^T\mathbf{x}_j||^2 W_{ij}^b \\ &= tr(\mathbf{E}_{inter1}^T\mathbf{X}(\mathbf{D}^b - \mathbf{W}^b)\mathbf{X}^T\mathbf{E}_{inter1}) \end{aligned}, \quad (5)$$

where $W_{ij}^b = 1$, if and only if $i \in \upsilon_{k_2}^b(j)$ or $j \in \upsilon_{k_2}^b(i)$; otherwise, $W_{ij}^b = 0$. $\upsilon_{k_2}^b(i)$ denotes the index set of the k_2 nearest neighbors of sample $\mathbf{x}_i$ with different class labels. $\mathbf{D}^b$ is a diagonal matrix, i.e., $D_{ii}^b = \sum_j W_{ij}^b$.

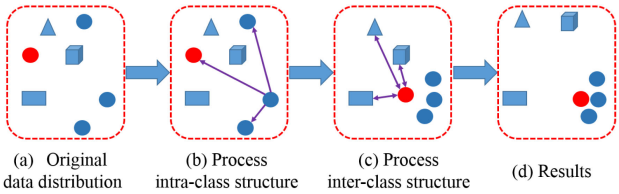


Fig. 4. An illustration of first minimizing the intra-class structure, and then maximizing the inter-class structure.

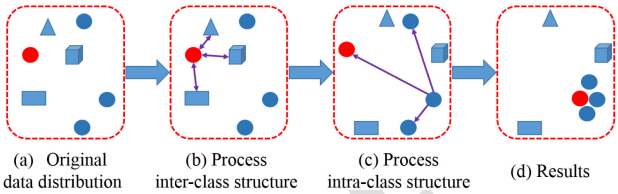


Fig. 5. An illustration of first maximizing the inter-class structure, and then minimizing the intra-class structure.

Similarly, we introduce a parameter $\beta \in [0,1]$ to tradeoff global and local inter-class sub-structures. The inter-class structure is designed to generate the extractor to extract DFs from images with different class labels. Thus, we maximize the extracted DFs and quantify the objective function of the inter-class structure as

$$\max_{\mathbf{E}_{inter1}} \beta S_{gb} + (1-\beta) S_{lb} \quad s.t.\ \mathbf{E}_{inter1}^T \mathbf{E}_{inter1} = \mathbf{I}. \tag{6}$$

3.2.3 Optimization

Our goal is to generate an extractor $\mathbf{E}$ that can extract effective DFs. The nature of DFs is that if they come from the same class of samples, they are compacted together; otherwise, they will be separated. There are two ways to generate the extractor by exploring these sub-structures hierarchically: the first way is to first optimize Eq. (3) and then optimize Eq. (6), called HDFE1. The second way is to first optimize Eq. (6) and then optimize Eq. (3), called HDFE2.

Fig. 4 illustrates the process of HDFE1. The red circle is first "pulled" by the blue circle of the same class label, as illustrated in Fig. 4b. Then, the red circle "pushes" the blue triangle, the blue cube and the blue rectangle away, as illustrated in Fig. 4c. Fig. 4d shows the results. Similarly, Fig. 5 illustrates the process of HDFE2. That is, the red circle first "pushes" the blue triangle, the blue cube and the blue rectangle away, as illustrated in Fig. 5b. Then, the red circle is "pulled" by the blue circle, as illustrated in Fig. 5c. Fig. 5d shows the results.

Algorithm 1. $\mathbf{E}_{intra1}$

Input: Data set $(\mathbf{x}_i, y_i)_{i=1}^N$, parameters α and k_1;
Output: $\mathbf{E}_{intra1}$;
1: Compute $\sum_{m=1}^{c}\sum_{i=1}^{N_m}(\mathbf{x}_i^m-\mu^m)(\mathbf{x}_i^m-\mu^m)^T$;
2: Compute $\mathbf{X}(\mathbf{D}^w-\mathbf{W}^w)\mathbf{X}^T$;
3: Compute Φ_{intra1}, where $\Phi_{intra1} = \alpha\sum_{m=1}^{c}\sum_{i=1}^{N_m}(\mathbf{x}_i^m-\mu^m)(\mathbf{x}_i^m-\mu^m)^T + (1-\alpha)tr(\mathbf{X}(\mathbf{D}^w-\mathbf{W}^w)\mathbf{X}^T)$;
4: Eigendecompose Φ_{intra1}, suppose its eigenvalues are λ_i, $i=1,\ldots,d$, and its corresponding eigenvectors are $\mathbf{e}_i$. Assume $\lambda_1 <,\ldots,< \lambda_i <,\ldots,< \lambda_d$;
5: $\mathbf{E}_{intra1} = [\mathbf{e}_1,\ldots,\mathbf{e}_r]$, where r is the number of negative eigenvalues;
6: **return** $\mathbf{E}_{intra1}$.

The objective functions of HDFE1 and HDFE2 include two steps. For HDFE1, its first step is to minimize $\alpha S_{gw} + (1-\alpha)S_{lw}$. Based on the results, the second step is to maximize $\beta S_{gb} + (1-\beta)S_{lb}$. That is, for the first step, we obtain the extractor $\mathbf{E}_{intra1}$ and based on it we obtain $\mathbf{V} = \mathbf{E}_{intra1}^T\mathbf{X}$. For the second step, we obtain the extractor $\mathbf{E}_{inter1}$ and based on it and features $\mathbf{V}$, we obtain DFs, i.e., $\mathbf{L} = \mathbf{E}_{inter1}^T\mathbf{V}$.

To gain more insight, according to Eq. (3), the first step of the HDFE1 is quantified as

$$\begin{aligned}\min_{\mathbf{E}_{intra1}} \mathbf{E}_{intra1}^T \Big[&\alpha\sum_{m=1}^{c}\sum_{i=1}^{N_m}(\mathbf{x}_i^m-\mu^m)(\mathbf{x}_i^m-\mu^m)^T \\ &+(1-\alpha)tr(\mathbf{X}(\mathbf{D}^w-\mathbf{W}^w)\mathbf{X}^T)\Big]\mathbf{E}_{intra1}, \\ s.t.\ &\mathbf{E}_{intra1}^T\mathbf{E}_{intra1} = \mathbf{I}.\end{aligned} \tag{7}$$

Since the $\mathbf{D}^w$ is a real diagonal matrix, it is a real symmetric matrix [38]. Thus, $\mathbf{D}^w - \mathbf{W}^w$ is a real symmetric matrix. For simplicity, we denote $\Phi_{intra1} = \alpha\sum_{m=1}^{C}\sum_{i=1}^{N_m}(\mathbf{x}_i^m-\mu^m)(\mathbf{x}_i^m-\mu^m)^T + (1-\alpha)tr(\mathbf{X}(\mathbf{D}^w-\mathbf{W}^w)\mathbf{X}^T)$. Based on [20], Φ_{intra1} is a real symmetric matrix. We switch Eq. (7) to a simple eigenvalue and eigenvector problem. Based on the Lagrange multiplier, Eq. (7) forms the Lagrangian function

$$\begin{aligned}\zeta(\mathbf{E}_{intra1}, \Lambda) = &tr(\mathbf{E}_{intra1}^T\Phi_{intra1}\mathbf{E}_{intra1}) \\ &-tr(\Lambda(\mathbf{E}_{intra1}^T\mathbf{E}_{intra1}-\mathbf{I}))\end{aligned}, \tag{8}$$

where $\Lambda = [\lambda_1,\ldots,\lambda_n]$. Then, we have $\Phi_{intra1}\mathbf{e}_i = \lambda_i\mathbf{e}_i$ by setting $\frac{\partial\zeta(\mathbf{E}_{intra1},\Lambda)}{\partial\mathbf{E}_{intra1}} = 0$. Thus, Eq. (8) can be rewritten as

$$E_{intra1} = \arg\min_{\mathbf{e}_i}\sum_{i=1}^{d}\mathbf{e}_i^T\Phi_{intra1}\mathbf{e}_i = \arg\min_{\mathbf{e}_i}\sum_{i=1}^{d}\lambda_i. \tag{9}$$

To optimize Eq. (9), we choose all negative eigenvalues of Φ_{intra1}. Thus, assuming that the number of negative eigenvalues of Φ_{intra1} is r, $\mathbf{E}_{intra1} = [\mathbf{e}_1,\ldots,\mathbf{e}_r]$. The process of the first step is given in Algorithm 1.

Based on the first step, we obtain $\mathbf{E}_{intra1}$ and features $\mathbf{V}$, i.e., $\mathbf{V} = \mathbf{E}_{intra1}^T\mathbf{X}$. According to Eq. (6) and features $\mathbf{V}$, the second step of HDFE1 can be quantified as

$$\begin{aligned}\max_{\mathbf{E}_{inter1}} \mathbf{E}_{inter1}^T[&\beta\sum_{m=1}^{c}N_m(\mu^m-\mu)(\mu^m-\mu)^T \\ &+(1-\beta)tr(\mathbf{V}(\mathbf{D}^b-\mathbf{W}^b)\mathbf{V}^T)]\mathbf{E}_{inter1}, \\ s.t.\ &\mathbf{E}_{inter1}^T\mathbf{E}_{inter1} = \mathbf{I}.\end{aligned} \tag{10}$$

We denote $\Phi_{inter1} = \beta\sum_{m=1}^{C}N_m(\mu^m-\mu)(\mu^m-\mu)^T + (1-\beta)tr(\mathbf{V}(\mathbf{D}^b-\mathbf{W}^b)\mathbf{V}^T)$. Similarly, Φ_{inter1} is a real symmetric matrix. In addition, we switch Eq. (10) to a simple eigenvalue and eigenvector problem. We form Eq. (10) as the Lagrangian function and set $\frac{\partial\zeta(\mathbf{E}_{inter1},\Lambda)}{\partial\mathbf{E}_{inter1}} = 0$, we have $\Phi_{inter1}\mathbf{e}_i = \lambda_i\mathbf{e}_i$. Thus, Eq. (10) can be rewritten as

$$E_{inter1} = \arg\max_{\mathbf{e}_i}\sum_{i=1}^{d}\mathbf{e}_i^T\Phi_{inter1}\mathbf{e}_i = \arg\max_{\mathbf{e}_i}\sum_{i=1}^{d}\lambda_i. \tag{11}$$

Algorithm 2. $\mathbf{E}_{inter1}$

Input: Data set $(\mathbf{x}_i, y_i)_{i=1}^N$, parameters β and k_2;
Output: $\mathbf{E}_{inter1}$;
1: Compute $\sum_{m=1}^{c} N_m(\mu^m - \mu)(\mu^m - \mu)^T$;
2: Compute $\mathbf{X}(\mathbf{D}^b - \mathbf{W}^b)\mathbf{X}^T$;
3: Compute Φ_{inter1}, where $\Phi_{inter1} = \beta\sum_{m=1}^{c} N_m(\mu^m - \mu)(\mu^m - \mu)^T + (1-\beta)tr(\mathbf{X}(\mathbf{D}^b - \mathbf{W}^b)\mathbf{X}^T)$;
4: Eigendecompose Φ_{inter1}, suppose its eigenvalues are λ_i, $i = 1, \ldots, d$, and its corresponding eigenvectors are $\mathbf{e}_i$. Assume $\lambda_1 > , \ldots, > \lambda_i > , \ldots, > \lambda_d$;
5: $\mathbf{E}_{inter1} = [\mathbf{e}_1, \ldots, \mathbf{e}_p]$, where p is the number of positive eigenvalues;
6: **return** $\mathbf{E}_{inter1}$.

Algorithm 3. HDFE1

Input: Data set $(\mathbf{x}_i, y_i)_{i=1}^N$, $\mathbf{E}_{intra1}$, $\mathbf{E}_{inter1}$;
Output: Extractor $\mathbf{E}$;
1: Perform $\mathbf{E}_{intra1}$ based on the input data $(\mathbf{x}_i, y_i)_{i=1}^N$;
2: Compute $\mathbf{V}$, where $\mathbf{V} = \mathbf{E}_{intra1}^T\mathbf{X}$;
3: Perform $\mathbf{E}_{inter1}$ based on the input data $(\mathbf{v}_i, y_i)_{i=1}^N$;
4: **return** $\mathbf{E}$, where $\mathbf{E} = \mathbf{E}_{intra1}\mathbf{E}_{inter1}$.

Algorithm 4. HDFE2

Input: Data set $(\mathbf{x}_i, y_i)_{i=1}^N$, $\mathbf{E}_{intra1}$, $\mathbf{E}_{inter1}$;
Output: Extractor $\mathbf{E}$;
1: Perform $\mathbf{E}_{inter1}$ based on the input data $(\mathbf{x}_i, y_i)_{i=1}^N$;
2: Compute $\mathbf{V}$, where $\mathbf{V} = \mathbf{E}_{inter1}^T\mathbf{X}$;
3: Perform $\mathbf{E}_{intra1}$ based on the input data $(\mathbf{v}_i, y_i)_{i=1}^N$;
4: **return** $\mathbf{E}$, where $\mathbf{E} = \mathbf{E}_{inter1}\mathbf{E}_{intra1}$.

To optimize Eq. (11), we choose all positive eigenvalues of Φ_{inter1}. Thus, assuming that the number of positive eigenvalues of Φ_{inter1} is p, $\mathbf{E}_{inter1} = [\mathbf{e}_1, \ldots, \mathbf{e}_p]$. The process of the second step is given in Algorithm 2.

After the two steps of the HDFE1 algorithm, we obtain two extractors $\mathbf{E}_{intra1}$ and $\mathbf{E}_{inter1}$. The DFs can be extracted by $\mathbf{E}_{intra1}$ and $\mathbf{E}_{inter1}$ and equals to $\mathbf{E}_{inter1}^T(\mathbf{E}_{intra1}^T\mathbf{X})$. In addition, the extractor $\mathbf{E} = \mathbf{E}_{intra1}\mathbf{E}_{inter1}$. The detailed process of HDFE1 is given in Algorithm 3.

For HDFE2, its first step is to maximize $\beta S_{gb} + (1-\beta)S_{lb}$. Based on the results of the first step, the second step is to minimize $\alpha S_{gw} + (1-\alpha)S_{lw}$. Similar to HDFE1, HDFE2 first computes the extractor $\mathbf{E}_{inter1}$, and then compute $\mathbf{E}_{intra1}$. Thus, in HDFE2, the extractor $\mathbf{E} = \mathbf{E}_{inter1}\mathbf{E}_{intra1}$. The detailed process of HDFE2 is given in Algorithm 4.

In addition, the extractor generated by HDFE1 can extract more effective DFs. The reason is that HDFE1 first minimizes the intra-class structure of the data. Thus, samples with the same class label will be aggregated, and linearly inseparable samples may become linearly separable. Then, HDFE1 maximizes the inter-class structure information of data to separate samples of different classes. In contrast, HDFE2 first maximizes the inter-class structure information of the data. Thus, different class labels of samples will push each other, which may make the distribution of the data is confusing. It would be that the linearly separable samples became linearly inseparable. Thus, in the subsequent minimization of the intra-class structure information, the same class label of samples cannot be aggregated together. Therefore, the extractor generated by HDFE1 can extract more effective DFs.

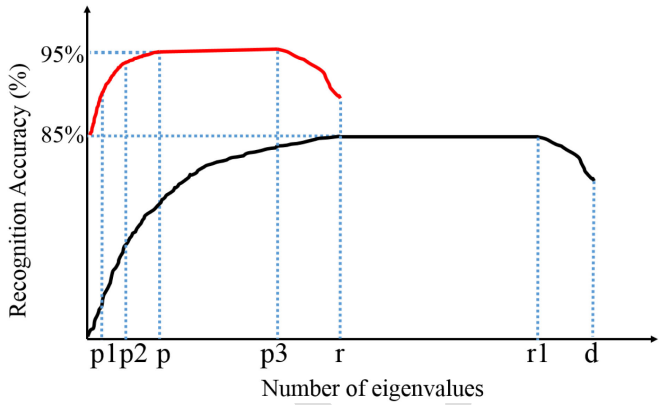


Fig. 6. An illustration of the relationship between recognition accuracy and the number of eigenvalues. The eigenvalues have been sorted from small to large in the first step and sorted from large to small in the second step. r is the number of negative eigenvalues, $r1-r$ is the number of zero eigenvalues, and $d-r1$ is the number of positive eigenvalues of the first step. p is the number of positive eigenvalues, $p3-p$ is the number of zero eigenvalues, and $r-p3$ is the number of negative eigenvalues in the second step. $p1$, $p2$, p, $p3$, r, $r1$ and d are positive real numbers.

3.3 The Sub-ED Algorithm

In real-world scenarios, to meet the user's requirements for delay and performance (e.g., recognition accuracy), the edge layer consists of three MEC platforms: access-level MEC platform, district-level MEC platform, and metro-regional-level MEC platform. As shown in Fig. 2. The access-level MEC platform is designed to support mMTC applications, e.g., augmented reality and smart city. The district-level MEC platform is designed to support uRLLC applications, e.g., connected cars. The metro-regional-level MEC platform is designed to support eMBB applications, e.g., video content delivery optimization. Therefore, we should deploy different extractors with different capabilities to different MEC platforms. For the access-level and district-level MEC platforms, we should deploy the extractor that extracts a small number of DFs to reduce network transmission delay and feature matching time. For the metro-regional-level MEC platform, we should deploy the extractor that extracts a large number of DFs to obtain high recognition accuracy.

To this end, we propose the Sub-ED algorithm, which divides $\mathbf{E}$ into two sub-extractors for deployment on the three-layer MEC platforms. We take the HDFE1 as an example. As shown in Fig. 6, the black curve indicates the relationship between the accuracy and the number of eigenvalues in the first step. As shown, in the first step, the eigenvectors corresponding to negative eigenvalues are advantageous for extracting DFs. The eigenvectors corresponding to zero eigenvalues are useless for extracting DFs. The eigenvectors corresponding to positive eigenvalues are detrimental to extracting DFs. When $\mathbf{E}_{intra1}$ consists of eigenvectors corresponding to r negative eigenvalues, the first step of HDFE1 can achieve the highest accuracy, e.g., 85%. Thus, for the first step, we obtain $\mathbf{E}_{intra1}$, which consists of eigenvectors corresponding to all negative eigenvalues of Φ_{intra1}.

In Fig. 6, the red curve indicates the relationship between the accuracy and the number of eigenvalues in the second step. Based on the results of the first step, we maximize $\beta S_{gb} + (1-\beta)S_{lb}$. Thus, as shown, the eigenvectors corresponding to positive eigenvalues are advantageous for extracting DFs. The eigenvectors corresponding to zero eigenvalues are useless for extracting DFs. The eigenvectors corresponding to negative eigenvalues are detrimental to extracting DFs. When $\mathbf{E}_{inter1}$ consists of eigenvectors corresponding to p positive eigenvalues, HDFE1 achieves the highest accuracy, e.g., 95%.

In addition, we observe that the accuracy rate increases rapidly at the beginning as the positive eigenvalue increases, and becomes slower later. This is because the eigenvalues are sorted and the absolute value of the eigenvalues is larger at the beginning. Thus, we can divide $\mathbf{E}$ in terms of the relationship between the recognition accuracy and the number of eigenvalues. As shown in Fig. 6, the red curve increases rapidly when the number of positive eigenvalues is between 0 and $p2$. The red curve increases slowly when the number of positive eigenvalues is between $p2$ and p. The access-level and district-level MEC platforms support latency-sensitive applications with almost real-time response. To this end, we choose $p1$ positive eigenvalues as the dimension of $\mathbf{E}1$, i.e., $\mathbf{E}1 = \mathbf{E}[\mathbf{e}_1, \ldots, \mathbf{e}_{p1}]$, where $p1 = \gamma * p2$, $\gamma \in (0, 1)$ is a positive real number. Lots of experimental results show that when γ is approximately equal to $1/2$, the HDFE1 can achieve low network traffic and guarantee recognition accuracy. Because the metro-regional-level MEC platform can tolerate larger latency, we choose $p2$ positive eigenvalues as the dimension of the sub-extractor $\mathbf{E}2$, i.e., $\mathbf{E}2 = \mathbf{E}[\mathbf{e}_1, \ldots, \mathbf{e}_{p2}]$. Compared with $\mathbf{E}1$ and $\mathbf{E}2$, $\mathbf{E}$ achieves the highest accuracy. Thus, we place $\mathbf{E}$ on the cloud server to serve high-accuracy applications, such as payment authentication.

In the Sub-ED algorithm, we first divide the extractor $\mathbf{E}$ into the sub-extractor $\mathbf{E}1 = \mathbf{E}[\mathbf{e}_1, \ldots, \mathbf{e}_{p1}]$ and sub-extractor $\mathbf{E}2 = \mathbf{E}[\mathbf{e}_1, \ldots, \mathbf{e}_{p2}]$. Then, we deploy $\mathbf{E}1$ in the access-level and district-level MEC platforms and deploy $\mathbf{E}2$ in the metro-regional-level MEC platform. In addition, we deploy $\mathbf{E}$ in the cloud server. Thus, extractors of different sizes are deployed on different MEC platforms to meet the diverse needs of users in terms of recognition accuracy and response time by extracting different numbers of DFs. In fact, the extractor can be divided into more fine-grained sub-extractors to cope with the different needs of IoT image recognition services.

In actual situations, the network conditions are dynamically changing, and fine-grained extractors are deployed so that users can select appropriate sub-extractors based on the current network status. For instance, when the current network condition is good and the user's response time requirements can be met, the user can select a larger sub-extractor to obtain higher recognition accuracy. Otherwise, the user can choose a smaller sub-extractor to get a faster response at the expense of slightly lower accuracy.

3.4 Summary

The proposed framework not only improves accuracy, but also meets different demands of IoT image recognition services. In the framework, HDFE1 can improve the accuracy by extracting effective DFs from the data set on the cloud servers and the data on MEC platforms. Extracting DFs from the data on MEC platforms can reduce network traffic and feature matching time. The Sub-ED algorithm can meet the different demands of users by dividing $\mathbf{E}$ into sub-extractors and deploying them into appropriate MEC platforms. This is because by extracting different numbers of DFs, network transmission delays and feature matching time and accuracy are affected.

In addition, because numerous IoT devices can use sub-extractors to extract DFs from the data on MEC platforms, the proposed framework can save network traffic. Assume that the pre-processed image data is 100 KB; the extracted DFs is 2 KB; the sub-extractor $\mathbf{E}2$ is 2 MB. When there are 10,000 users requesting the image recognition, the MEC platform upload the pre-processed image data to the cloud servers with the network traffic of 1,000,000 KB. The total network traffic consumption of the cloud server sending the sub-extractor $\mathbf{E}2$ to the MEC platform and uploading the DFs data to the cloud servers by the MEC platform is 22,048 KB. The results show that uploading the DFs data can reduce network traffic. In addition, sub-extractors $\mathbf{E}1$ and $\mathbf{E}2$ are not updated every time. When the new image accumulates a certain amount of data, we use the new image and HDFE1 algorithm to generate a new extractor, then use the Sub-ED algorithm to divide the new extractor into two sub-extractors and send them to the corresponding MEC platforms. We will leave it as our future work. As the number of IoT devices requesting image recognition increases, the proposed framework saves more network traffic.

4 Experiments

4.1 Datasets

Synthetic Data. We generate three classes of uniformly distributed samples. The first class samples are numbers between intervals [0, 1]. The second class samples are numbers between intervals [0.8, 1.8]. The third class samples are numbers between intervals [1.6, 2.6].

Yale [39]. The Yale data set contains 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject. The size of each image is resized to 32×32 pixels.

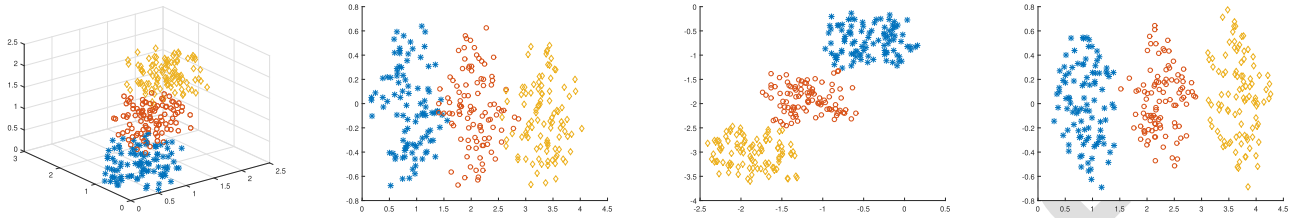
Leaves [40]. The Leaves data set contains 186 images of 3 species of leaves against cluttered different backgrounds. Each image is resized to 32×32 pixels.

USPS [41]. The USPS data set contains 1856 images of 10 individuals. The size of each image is 16×16 pixels.

COIL20 [42]. The Columbia Object Image Library (COIL-20) contains 1440 images of 20 objects. The objects were placed on a motorized turntable against a black background. The size of each image is 32×32 pixels.

UMIST [43]. The UMIST data set consists of 564 images of 20 individuals, taking into account race, sex and appearance. Each image is resized to 32×32 pixels.

In addition, we compare the proposed algorithm with DAG-DNE [35], JGLDA [36] and WAPL [20] algorithms. DAG-DNE algorithm only includes local intra-class and inter-class structures. Although JGLDA algorithm includes all structures, they are treated equally when generating the extractor. WAPL not only includes all structures (i.e., global



(a) The original data. (b) WAPL (c) HDFE1 (d) HDFE2

Fig. 7. Visualization of the two-dimensional discriminative features.

and local intra-class structure, and global and local inter-class structure), but also introduces trade-off parameters (i.e., α, β and γ) to reasonably controls the contribution of each structure. When evaluating the performance of the Sub-ED algorithm, we compare HDFE1 with PCA [7], [8], because PCA is one of the most popular feature extractor generation algorithms and is widely used to extract features of data. PCA aims to find a small number of features from a large number of potentially relevant features that retain as much information as possible.

4.2 Performance of Visualization

4.2.1 Experimental Setup

To evaluate the extracted DFs, we compare the proposed algorithms with the WAPL algorithm, because WAPL should be able to generate the extractor that can extract more effective DFs than those generated by DAG-DNE and JGLDA, as discussed above. In the experiment, parameters α, β, and γ in WAPL are 0.7, 0.5, and 0.2, respectively. The parameters α and β in the HDFE1 are 0.9 and 0.2, respectively. The parameters α and β in the HDFE2 are 0.9 and 0.8, respectively.

4.2.2 Synthetic Data Set Visualization

The extractors generated by HDFE1 and HDFE2 can extract more effective DFs from the data set. Fig. 7 shows that the DFs extracted by extractors generated by HDFE1 and HDFE2 have the characteristics: if they come from data with the same class label, they are aggregated; otherwise, they are separable. However, the capabilities of the extractors generated by HDFE1 and HDFE2 are more efficient than the capabilities of the extractor generated by WAPL. As shown in Fig. 7. Fig. 7b shows some overlap between DFs of different class labels of data. Figs. 7c and 7d show that the DFs of different class labels of data are separate. The reason is that WAPL generates the extractor by simultaneously optimizing the structural information of the data. Although WAPL controls the contribution of each structure through trade-off parameters, these data still interfere with each other. HDFE1 and HDFE2 generate extractors by hierarchically optimizing the structural information of the data, thus avoiding mutual interference between the data, so that the generated extractors can extract more effective DFs.

The capability of the extractor generated by HDFE1 is much higher than that of the extractor generated by HDFE2. As illustrated in Figs. 7c and 7d, DFs from the same class samples are more aggregated. This is because, if the inter-class structure is optimized first, the distribution of the samples will become confusing because the samples of different classes push each other, and even the linearly separable samples become linearly inseparable, making samples more difficult to separate. However, if the intra-class structure is optimized first, then the same class samples will be pulled together and will be aggregated, even the linearly inseparable samples will be linearly separable. So, the extractor generated by HDFE1 can extract more effective DFs.

4.3 Performance of Recognition Accuracy

4.3.1 Experimental Setup

We randomly split each data set into a training set $\mathbf{X}_{tr}$ and a test set $\mathbf{X}_{te}$. $\mathbf{X}_{tr}$ is randomly split into a training set $\mathbf{X}_{tr1}$ and a validation set $\mathbf{X}_{te1}$. We use the training set $\mathbf{X}_{tr1}$ to tune parameters and use the validation set $\mathbf{X}_{te1}$ to validate them. For WAPL, we use the training set $\mathbf{X}_{tr1}$ to tune α, β, and γ. When training one parameter, the other two parameters are set to 0.5. For HDFE1 and HDFE2, we also use the training set $\mathbf{X}_{tr1}$ to tune α and β and use the validation set $\mathbf{X}_{te1}$ to validate them. k_1 and k_2 indicate the number of neighbors selected when preserving the local intra-class and inter-class structures, we shows the results when $k_1 = k_2 = 1$, $k_1 = k_2 = 3$ and $k_1 = k_2 = 5$. We compare the proposed HDFE1 and HDFE2 algorithms with DAG-DNE, JGLDA and WAPL algorithms. When comparing all algorithms, their training set and test set are the same.

Finally, the nearest neighbor classifier [44] is performed on the test set $\mathbf{X}_{te}$. We report the best mean accuracy and standard deviation over 30 random splits for each data set.

4.3.2 Improvement of Recognition Accuracy

HDFE1 and HDFE2 achieve higher recognition accuracy. Table 1 shows that HDFE1 and HDFE2 achieve the highest recognition accuracy on all data sets. As shown, on Yale data set, when $k = 1$, the recognition accuracy of HDFE1 is 6.44% higher than that of WAPL. The reason is that HDFE1 and HDFE2 hierarchically explore the structural information of the image data, avoiding the interference between image data. Although JGLDA incorporates all sub-structures, it treats them equally. However, different sub-structures have different contributions to generating the extractor. Furthermore, it optimizes these sub-structures simultaneously and makes them interfere with each other. The DAG-DNE cannot generate an extractor that extracts effective DFs because it incorporates only a subset of these sub-structures (e.g., local intra-class and inter-class sub-structures). Although WAPL contains all sub-structures and processes them reasonably, it cannot generate an extractor for extracting DFs

TABLE 1
Image Retrieval Accuracy (% ± std)

Data Sets	$k = k_1 = k_2$	Recognition Accuracy (%)				
		JGLDA	DAG-DNE	WAPL	HDFE1	HDFE2
Yale	k=1	72.67±0.57	70.67±0.43	75.56±0.83	**82.00±0.22**	77.67±0.87
	k=3	80.00±0.97	80.00±0.62	82.22±0.47	**84.44±0.47**	82.67±0.52
	k=5	86.67±0.35	77.78±0.45	86.67±0.85	**88.89±0.16**	**88.89±0.53**
Leaves	k=1	75.83±0.46	75.83±0.89	79.44±0.68	**80.28±0.11**	80.00±0.77
	k=3	77.22±0.72	75.56±0.63	75.83±0.37	**78.33±0.39**	77,22±0.49
	k=5	76.94±0.69	76.67±0.46	77.94±0.59	**79.72±0.06**	78.33±0.23
USPS	k=1	93.96±0.23	95.79±0.54	96.00±0.28	**97.80±0.18**	97.52±0.42
	k=3	93.59±0.30	95.60±0.58	96.00±0.87	**97.25±0.36**	96.70±0.26
	k=5	94.32±0.91	95.97±0.34	95.99±0.45	96.15±0.61	**97.25±0.32**
COIL20	k=1	84.40±0.50	84.74±0.39	85.67±0.45	**88.83±0.41**	87.58±0.27
	k=3	86.77±0.69	87.45±0.41	88.21±0.73	**89.14±0.24**	**89.14±0.51**
	k=5	84.24±0.73	85.76±0.58	86.12±0.26	**87.11±0.13**	86.20±0.37
UMIST	k=1	97.22±0.55	97.32±0.26	98.06±0.82	**98.63±0.41**	98.35±0.63
	k=3	98.35±0.78	97.64±0.34	98.25±0.79	**98.87±0.33**	**98.87±0.47**
	k=5	98.10±0.64	97.87±0.54	98.46±0.55	**98.57±0.29**	98.56±0.31

because it optimizes these sub-structures at the same time, which leads to mutual interference between image data. Therefore, HDFE1 and HDFE2 can generate extractors that can extract more effectively DFs compared with JGLDA, DAG-DNE, and WAPL.

The recognition accuracy of HDFE1 is higher than that of HDFE2 in most cases. As shown in Table 1, except on the USPS data set, when $k = 5$, the recognition accuracy of HDFE2 (i.e., 97.25%) is higher than the recognition accuracy of the HDFE1 (i.e., 96.15%), and the others are HDFE1 with a higher recognition accuracy. This is because that HDFE1 first minimizes the intra-class structure, and the same class images will be pulled together, which makes the subsequent different classes of image data easier to separate. However, HDFE2 first maximizes the inter-class structure, and image data of different classes will be pushed away, which can confuse the image data. Therefore, HDFE1 can generate an extractor that can extract more effective DFs and achieves higher recognition accuracy.

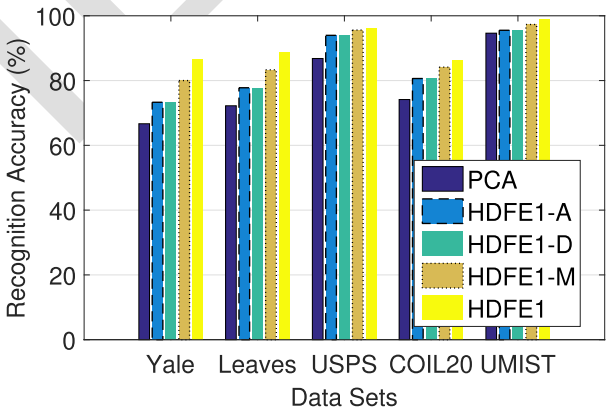


Fig. 8. Recognition accuracy of different MEC platforms. HDFE1-A, HDFE1-D, and HDFE1-M refer to sub-extractors on access-level MEC platform, district-level MEC platform, and metro-regional-level MEC platform, respectively.

4.4 Performance of the Sub-ED Algorithm

4.4.1 Experimental Setup

We use 10% (15), 10% (18), 10% (182), 10% (1280), and 50% (284) images of the Yale, Leaves, USPS, COIL20, and UMIST data set as test images. Note that the number in the parentheses is the number of test samples. The parameters of α and β are turned as in Section 4.3.1. We set $k_1 = 1$ and $k_2 = 1$. In the Yale data set, $\gamma = 8/15$. In the Leaves data set, $\gamma = 10/19$. In the USPS data set, $\gamma = 16/37$. In the COIL20 data set, $\gamma = 5/11$. In the UMIST data set, $\gamma = 5/8$. Without any knowledge, we can set $\gamma = 1/2$.

4.4.2 Results

HDFE1 improves recognition accuracy across all data sets on all MEC platforms. As illustrated in Fig. 8, compared with PCA, the HDFE1 improves recognition accuracy by 23.08% on the Yale data set. For the access-level and district-level MEC platform, compared with PCA, the HDFE1 improves recognition accuracy by 9.05% on the Yale data set. For the metro-regional-level MEC platform, compared with PCA, the HDFE1 improves recognition accuracy by 11.88% on the COIL20 data set. This is because HDFE1 can generate an extractor that extracts effective DFs from image data set. However, PCA aims to extract features that can preserve the global structural information of the image data, rather than DFs that facilitate recognition or classification tasks. Therefore, compared with PCA, HDFE1 can achieve higher recognition accuracy by extracting effectively DFs.

HDFE1 reduces network traffic and feature matching time on all MEC platforms across. As illustrated in Fig. 9, for the access-level and district-level MEC platforms, HDFE1 reduces network traffic by 94.97% on the COIL20 data set compared with PCA. Fig. 10 shows that, for the access-level and district-level MEC platforms, HDFE1 reduces feature matching time by 26.83% on the COIL20 data set compared with PCA. The reason is that HDFE1

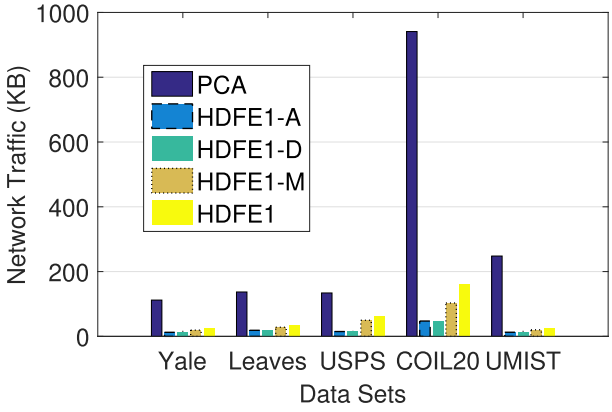


Fig. 9. Network traffic of different MEC platforms. For the meanings of HDFE1-A, HDFE1-D, and HDFE1-M, please refer to Fig. 8.

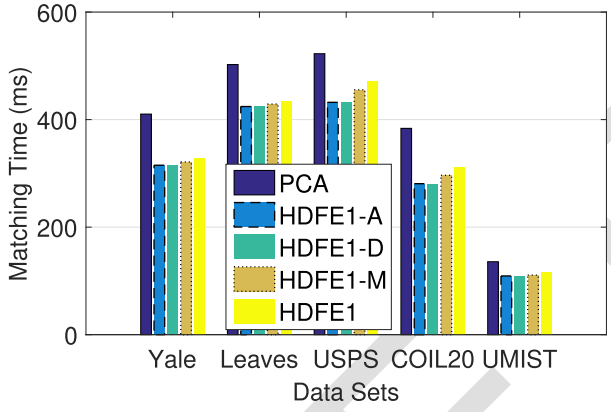


Fig. 10. Matching time of different MEC platforms. For the meanings of HDFE1-A, HDFE1-D, and HDFE1-M, please refer to Fig. 8.

can extract a small number of DFs from the image data. Thus, the corresponding MEC platforms only need to upload a small amount of DFs data to the cloud server. A small amount of DFs data consumes a small amount of network transmission traffic, which also has a short network transmission delay. A small number of DFs also indicates a shorter feature matching time, because there are fewer DFs that need to be matched. Therefore, HDFE1 can reduce network traffic and feature matching time. This also shows to a certain extent that HDFE1 can reduce the response time of image recognition services, because the response time mainly includes network transmission time and feature matching time.

The more images that are processed, the more network traffic and feature matching time saved. As illustrated in Fig. 9, compared with PCA, HDFE1-A reduces network traffic by 88.88% on the USPS data set. Compared with PCA, HDFE1-A reduces network traffic by 86.35% on the Leaves data set. Fig. 10 shows that HDFE1-A reduces feature matching time by 17.31% compared with PCA on the USPS data set. Compared with PCA, HDFE1-A reduces feature matching time by 15.53% on the Leaves data set. This is because a large number of images are processed by the extractor on the USPS data set. For instance, if one image can save 1 KB, then two images can save 2 KB. Similarly, if one image can save 100 milliseconds, then two images can save 200 milliseconds. Therefore, as the number of images processed by the extractor increases, more network transmission traffic and feature matching time can be saved.

5 Conclusion

In this paper, we mainly make the following three contributions. First, we propose an IoT services framework to adapt the hierarchical MEC environment. Second, we propose the HDFE algorithm to generate an extractor to extract DFs from the image data set on cloud servers and images on MEC platforms. In addition, we analyze two version of HDFE: HDFE1 and HDFE2. Third, we propose the Sub-ED algorithm to divide the extractor into sub-extractors and deploy them on different MEC platforms to meet the diverse demands of IoT services. Experimental results show that the proposed framework can improve recognition accuracy, reduce network traffic and feature matching time, and meet diverse demands of IoT services. In future work, we will deploy the proposed framework in the real MEC environment.

Acknowledgments

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B010164002 and in part by the NSFC under Grants 61922017 and 61921003.

References

[1] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 129–144.

[2] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2114–2127, Sep. 2018.

[3] S. Deng *et al.*, "Dynamical resource allocation in edge for trustable internet-of-things systems: A reinforcement learning method," *IEEE Trans. Ind. Inform.*, vol. 16, no. 9, pp. 6103–6113, Sep. 2020.

[4] D. Yao, C. Yu, L. T. Yang, and H. Jin, "Using crowdsourcing to provide QoS for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 344–356, Apr.–Jun. 2019.

[5] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 164–174, Feb. 2018.

[6] C. Ding, A. Zhou, X. Ma, and S. Wang, "Cognitive service in mobile edge computing," in *Proc. IEEE Int. Conf. Web Serv.*, Oct. 2020, pp. 181–188.

[7] Y. Sun, X. Tao, Y. Li, and J. Lu, "Robust 2D principal component analysis: A structured sparsity regularized approach," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2515–2526, Aug. 2015.

[8] C. Kim and D. Klabjan, "A simple and fast algorithm for L1-norm kernel PCA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 1842–1855, Aug. 2020.

[9] C. Papageorgiou, M. Oren, and T. A. Poggio, "A general framework for object detection," in *Proc. Int. Conf. Comput. Vis.*, 1998, pp. 555–562.

[10] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[11] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1657–1681, Jul.–Sep. 2017.

[12] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[13] H. Huang and S. Guo, "Proactive failure recovery for NFV in distributed edge computing," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 131–137, May 2019.
[14] S. Deng *et al.*, "Optimal application deployment in resource constrained distributed edges," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1907–1923, May 2021.
[15] S. Dahlfort, A. D. Gregorio, G. Fiaschi, S. Khan, J. Rosenberg, and T. Thyni, "Enabling intelligent transport in 5G networks," 2018. [Online]. Available: https://www.ericsson.com/en/ericsson-technology-review/archive/2018/enabling-intelligent-transport-in-5g-networks
[16] A. Reznik *et al.*, "Cloud RAN and MEC: A perfect pairing," 2018. [Online]. Avaiable: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp23_MEC_and_CRAN_ed1_FINAL.pdf
[17] J. Wang, C. Jiang, Z. Han, Y. Ren, and L. Hanzo, "Network association strategies for an energy harvesting aided super-WiFi network relying on measured solar activity," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3785–3797, Dec. 2016.
[18] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Trans. Ind. Inform.*, vol. 13, no. 4, pp. 1910–1920, Aug. 2017.
[19] J. Li, Z. Peng, B. Xiao, and Y. Hua, "Make smartphones last a day: Pre-processing based computer vision application offloading," in *Proc. 12th Annu. IEEE Int. Conf. Sens., Commun., Netw.*, 2015, pp. 462–470.
[20] S. Wang *et al.*, "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 292–305, Feb. 2021.
[21] X. Sun and N. Ansari, "Green cloudlet network: A sustainable platform for mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 180–192, Jan.–Mar. 2020.
[22] N. Fernando, S. W. Loke, and W. Rahayu, "Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 329–343, Apr.–Jun. 2019.
[23] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "FogFlow: Easy programming of IoT services over cloud and edges for smart cities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 696–707, Apr. 2018.
[24] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proc. 9th Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.
[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[26] Z. Guo, X. Wang, J. Zhou, and J. You, "Robust texture image representation by scale selective local binary patterns," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 687–699, Feb. 2016.
[27] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Pull-in time dynamics as a measure of absolute pressure," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 276–286.
[28] J. Ye, R. Janardan, C. H. Park, and H. Park, "An optimization criterion for generalized discriminant analysis on undersampled problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 982–994, Aug. 2004.
[29] B. Li, L. Lei, and X.-P. Zhang, "Constrained discriminant neighborhood embedding for high dimensional data feature extraction," *Neurocomputing*, vol. 173, pp. 137–144, 2016.
[30] X. Li, M. Chen, F. Nie, and Q. Wang, "Locality adaptive discriminant analysis," in *Proc. 26th Int. Conf. Artif. Intell.*, Aug. 2017, pp. 2201–2207.
[31] B. Li, C.-H. Zheng, and D.-S. Huang, "Locally linear discriminant embedding: An efficient method for face recognition," *Pattern Recognit.*, vol. 41, no. 12, pp. 3813–3821, 2008.
[32] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, 2007.
[33] W. Zhang, X. Xue, H. Lu, and Y.-F. Guo, "Discriminant neighborhood embedding for classification," *Pattern Recognit.*, vol. 39, no. 11, pp. 2240–2243, 2006.
[34] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[35] C. Ding and L. Zhang, "Double adjacency graphs-based discriminant neighborhood embedding," *Pattern Recognit.*, vol. 48, no. 5, pp. 1734–1742, 2015.
[36] Q. Gao, J. Liu, H. Zhang, X. Gao, and K. li, "Joint global and local structure discriminant analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 4, pp. 626–635, Apr. 2013.
[37] M. Shafi *et al.*, "5G: A tutorial overview of s… lenges, deployment, and practice," *IEEE J.* … vol. 35, no. 6, pp. 1201–1221, Jun. 2017.
[38] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, …
[39] The Yale Dataset. [Online]. Available: ht… cvc/projects/yalefaces/yalefaces.html
[40] The Leaves Dataset. [Online]. Available: http://www.vision.caltech.edu/archive.html
[41] The USPS Dataset. [Online]. Available: … roweis/data.html
[42] The COIL20 Dataset. [Online] Available:http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[43] The UMIST Dataset. [Online] Available:ht… ac.uk/eee/research/iel/research/face
[44] B. Liao, L. H. U, M. L. Yiu, and Z. Gong, … latency kNN search on commodity machine… *Data Eng.*, vol. 27, no. 10, pp. 2618–2631, Oct.

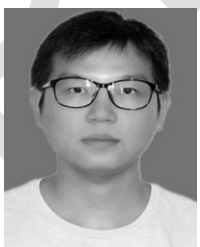
accessed: 9-Sep-2021.

accessed: 9-Sep-2021.

accessed: 9-Sep-2021.

The COIL20 Dataset. [Online] Available: https://www.kaggle.com/bistaumanga/usps-dataset, accessed: 9-Sep-2021.

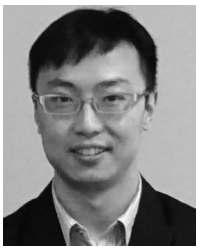
The UMIST Dataset. [Online] Available: https://www.visioneng.org.uk/datasets/, accessed: 9-Sep-2021.



Chuntao Ding received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2021. He is currently a lecturer with Beijing Jiaotong University. He was a research assistant … technic University and a … Michigan State University. … coauthored many researc… published in many prestig… ferences including the *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Cloud Computing*, and IEEE ICWS. His research interests include edge computing and machine learning.

Ao Zhou received the PhD degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an associate professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has authored or coauthored more than 20 research papers. Her research interests include cloud computing and mobile edge computing. She also played a key role at many international conferences.

Xiao Ma received the BS degree in telecommunication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2013, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University in 2018. She is currently a postdoctoral fellow with the State Key Laboratory of Networking and Switching Technology, BUPT. Her research interests include task scheduling and allocation in mobile cloud computing and mobile edge computing.



Ning Zhang (Member, IEEE) received the PhD degree from the University of Waterloo, Canada, in 2015. His research interests include wireless communication and networking, mobile edge computing, machine learning, and physical layer security. He is currently an associate editor for the *IEEE Internet of Things Journal*, *IEEE Transactions on Cognitive Communications and Networking*, *IEEE Access* and *IET Communications*, and an area editor of the *Encyclopedia of Wireless Networks* (Springer) and Cambridge Scholars.

Ching-Hsien Hsu (Senior Member, IEEE) is currently the chair professor and dean of the College of Information and Electrical Engineering, Asia University, Taiwan. He has authored or coauthored more than 200 papers in top journals such as the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *ACM Transactions on Multimedia Computing, Communications, and Applications*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Emerging Topics in Computing*, *IEEE Systems Journal*, *IEEE Network Magazine*, top conference proceedings, and book chapters in his areas of research, which include high-performance computing, cloud computing, parallel and distributed systems, big data analytics, and ubiquitous/pervasive computing and intelligence. He is a fellow of the IET (IEE), the vice chair of the IEEE Technical Committee on Cloud Computing (TCCLD), and IEEE Technical Committee on Scalable Computing (TCSC).

Shangguang Wang (Senior Member, IEEE) received the PhD degree from the Beijing University of Posts and Telecommunications (BUPT), China, in 2011. He is currently a professor with the Beiyou Shenzhen Research Institute and the School of Computing, BUPT. He has authored or coauthored more than 200 papers. His research interests include service computing, cloud computing, and mobile edge computing. He was the general chair or TPC chair of the IEEE EDGE 2020, IEEE CLOUD 2020, IEEE SAGC 2020, IEEE EDGE 2018, and IEEE ICFCE 2017. He is currently the Chair of IEEE Technical Committee on Services Computing (2022–2023) and the vice-chair of IEEE Technical Committee on Cloud Computing.