Resource-aware Feature Extraction in Mobile Edge Computing

Chuntao Ding, Ao Zhou, Xiulong Liu, Xiao Ma, Shangguang Wang, *Senior Member, IEEE*

Abstract—Mobile image recognition services, which provide people with image recognition services through the cameras of mobile devices, are revolutionizing our lives. However, most existing cloud/edge-based approaches suffer from two major limitations, (i) Low recognition accuracy and high network bandwidth pressure, and (ii) Not easy to extract features based on currently available resources of mobile devices. In this paper, we propose a resource-aware feature extraction framework for mobile image recognition services. The proposed framework consists of discriminative feature extraction (DFE) and NestDFE algorithms. The DFE algorithm can generate an extractor $\mathbf{E}$ to extract discriminative features from the image data set on the edge server and images on mobile devices. Thus, the proposed framework can achieve higher recognition accuracy and require mobile devices to upload less feature data to the edge server. The NestDFE algorithm generates a single multi-capacity extractor that acts as a series of sub-extractors and enables mobile devices to dynamically select sub-extractors. Experimental results show that the proposed framework improves recognition accuracy by about 23% and reduces network traffic by about 76% compared with existing approaches.

Index Terms—Mobile edge computing, resource-aware, cloud computing, feature extraction.

1 Introduction

1.1 Motivation & Problem Formulation

Mobile image recognition services greatly facilitate our lives by providing various cognitive assistance [1]–[5]. For example, Aipoly and TapTapSee can meet the needs of visually impaired users, and CalorieMama snaps a picture of your meal and get all the nutritional information you need to stay fit and healthy. The most popular mobile image recognition services solution is based on cloud computing technology [6]–[9]. Recently, mobile edge computing provides high bandwidth and low latency potential by deploying edge servers near mobile devices [10]–[17]. However, with the rapid development of the Internet of Everything, the network bandwidth is still a bottleneck for offloading a large amount of data to cloud/edge servers. To alleviate this predicament, many approaches first pre-process the captured image data and then upload it to the edge server. Thus, they can reduce network traffic and lessen the pressure of the network bandwidth.

In addition, the discriminative features of image data are important for recognition tasks. However, it is difficult to extract discriminative features from images on mobile devices. This is because mobile devices do not have image label information, which is necessary to extract discriminative features. Furthermore, the amount of available resources of the mobile device may change frequently because it often runs multiple applications at the same time, and we may frequently launch new applications and close existing ones.

- *Chuntao Ding, Ao Zhou, Xiao Ma and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China 100876. E-mail: {ctding;aozhou;maxiao18;sgwang}@bupt.edu.cn.*
- *Xiulong Liu is with College of Intelligence and Computing, Tianjin University, Tianjin, China. E-mail: xiulongliudut@gmail.com. (Corresponding author: Ao Zhou.)*

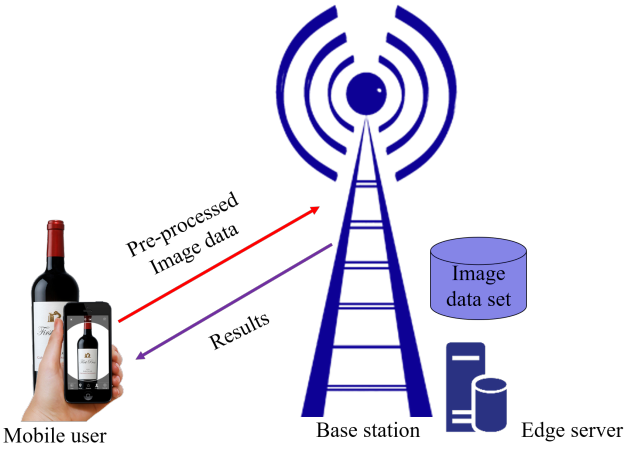


Fig. 1. System architecture for edge-based mobile image recognition.

Therefore, it is important to study how to extract the discriminative features of images based on currently available resources of mobile devices to reduce network traffic while ensuring high recognition accuracy.

This paper studies the problem of mobile image recognition services in the mobile edge computing context. As illustrated in Fig. 1, the system architecture consists of two types of components: mobile devices and edge servers. Mobile devices communicate with edge servers through base stations and connect to the base stations through LTE. We assume that an image data set is stored on the edge server and is represented as $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where N is the number of images, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}^K$, d is the dimension of $\mathbf{x}_i$, and K is the number of class labels. A mobile user uploads the pre-processed image data to the edge server to launch a request. After receiving the pre-processed image data, the edge server processes it and sends back the label information of the most similar image to the mobile user.

1.2 Limitations of Prior Art

Prior related mobile image recognition approaches [18]–[20] have two major drawbacks. First, they suffer from low recognition accuracy and high network bandwidth pressure because they aim to extract numerous features to preserve the intrinsic structure of the image data, rather than extracting effective discriminative features. Second, they are not easily aware of the available resources of the mobile device, and cannot dynamically determine the number of features.

1.3 Proposed Approach

In this paper, we propose a resource-aware feature extraction framework, which includes discriminative feature extraction (DFE) and NestDFE algorithms. In the proposed framework, we first propose the DFE algorithm to generate an extractor $\mathbf{E}$ on the edge server by using the information of the image data set. Then, we propose the NestDFE algorithm to divide $\mathbf{E}$ into multiple sub-extractors and form these sub-extractors into a single multi-capacity extractor. Follow that, we use the multi-capacity extractor to extract discriminative features from the image data set to form multiple feature sets. In addition, we send the multi-capacity extractor to the mobile device. When capturing an image, the mobile device pre-processes it and selects the appropriate sub-extractor from the multi-capacity extractor based on its currently available resources to extract discriminative features. Then, the mobile device uploads the extracted feature data to the edge server. After receiving the feature data, the edge server matches it with the feature data on the feature set and finds the most similar feature data. Finally, the edge server sends back the label of the most similar feature data to the mobile user. The label is regarded as the recognition result. The key idea of the proposed framework is to dynamically extract discriminative features from images based on the currently available resources of the mobile device.

1.4 Challenges and Proposed Solutions

The first key challenge is to generate the extractor $\mathbf{E}$ to extract effective discriminative features. Discriminative features are important because they determine the quality of services (QoS) of mobile image recognition, which includes recognition accuracy and response time.

To address this challenge, we propose the DFE algorithm to generate the extractor $\mathbf{E}$ to extract a small amount of effective discriminative features from the image data set and images on mobile devices, which can not only achieve high recognition accuracy, but also reduce the response time by reducing the network traffic and the number of matching features. To achieve this purpose, we first build a novel similarity function to preserve intra-class and inter-class structures of the image data set. Then, we divide intra-class and inter-class structures into four substructures and introduce trade-off parameters to control their weights. Thus, the DFE algorithm can generate an effective $\mathbf{E}$ to extract effective discriminative features.

The second key challenge is to dynamically select extractors based on the current resources of mobile devices. The amount of available resources of mobile devices changes dynamically because we may frequently launch new applications and close existing ones. If there are not enough available resources in the mobile device to support the feature extraction operation, the extractor $\mathbf{E}$ cannot work. To adapt the dynamic changes of available resources on mobile devices, a straightforward solution is to divide $\mathbf{E}$ into multiple sub-extractors and store all these sub-extractors. By doing so, mobile devices can dynamically select appropriate sub-extractors based on their available resources. However, it is not wise to store all sub-extractors, because they take up a lot of memory space.

To address this challenge, we propose the NestDFE algorithm to generate a single multi-capacity extractor. In the NestDFE algorithm, we divide $\mathbf{E}$ into multiple sub-extractors, and these sub-extractors form a single multi-capacity extractor. It is worth noting that the multi-capacity extractor takes up the same amount of memory space as $\mathbf{E}$. This is because the sub-extractor with a smaller capacity shares all its parameters to the sub-extractor with a larger capacity, and nests itself in the latter so as not to occupy additional memory space. Hence, the multi-capacity extractor can save a lot of memory space and enable mobile devices to dynamically select sub-extractors.

1.5 Novelty and Advantages over Prior Art

The key technical novelty of this paper is in proposing a resource-aware feature extraction framework, which consists of DFE and NestDFE algorithms. The key technical depth of this paper is in generating the extractor to extract effective discriminative features and nesting multiple sub-extractors into a single multi-capacity extractor. The key advantages of the proposed framework over the previous approaches are two-fold: (i) It has higher recognition accuracy and less network traffic because it uses the extractor $\mathbf{E}$ to extract a small amount of effective discriminative features; (ii) It can dynamically determine the number of features extracted from the image based on the available resources of mobile devices. Extensive experimental results show that, compared with state-of-the-art approaches, the proposed framework improves recognition accuracy by about 23%, reduces the network traffic by about 76%.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 describes the proposed framework. Section 4 presents our evaluation results and analysis. Finally, we conclude this paper in Section 5.

2 Related Work

Mobile image recognition mainly includes feature extraction and feature matching. Feature matching is to match the extracted features with the features on the feature set. Feature extraction aims to extract features from images. Many classic feature extraction algorithms have been proposed, such as scale invariant feature transform (SIFT) [21], local binary pattern (LBP) [22] and principle component analysis (PCA) [23]. The SIFT [21] has excellent noise immunity, illumination, partial occlusion and geometric transformation robustness. The LBP [22] uses a local neighborhood around each pixel, thresholds the pixels of the neighborhood to the value of the central pixel, and uses the resulting binary

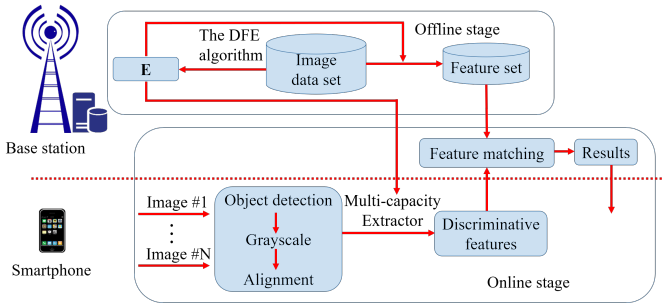


Fig. 2. The proposed framework overview.

image block as the local image descriptor. The PCA [23] preserves global information of the image data. However, they aim to preserve the intrinsic structure of the image data, rather than extracting discriminative features. To address this problem, numerous algorithms have been proposed [24]–[27]. For example, joint global and local structure discriminant analysis [25] generates the extractor by integrating global and local geometrical structures into the objective function of linear discriminant analysis. Locality adaptive discriminant analysis [27] generates the extractor by investigating the geometry of the local data structure. However, they do not consider the distance gap between the inter-class structure and intra-class structure, and ignore the importance of the two structures when generating the extractor. This affects the ability of the extractor they generate to extract discriminative features.

There are also a lot of mobile image recognition approaches based on mobile edge computing [18]–[20], [28], [29]. For instance, Li *et al.* [18] propose an online decision approach for determining the level of pre-processing on mobile devices. Liu *et al.* [28] propose a deep learning-based food recognition system for dietary assessment in the edge computing service infrastructure. Hu *et al.* [19] propose a resolution framework based on fog computing, which transfers some computing overhead from the cloud to network edge devices to improve processing efficiency and reduce network transmission. Drolia *et al.* [20] propose a system that uses offline analysis of applications and online estimation of network conditions to adaptive balance the load between the edge and the cloud, thereby minimizing latency. Soyata *et al.* [29] study how to perform task division from mobile devices to the cloud in order to minimize response time given various communication delays and server computing power. These approaches benefit from the mobile edge computing architecture in terms of network traffic and response time. However, they do not consider the impact of dynamic changes in mobile device resources on applications.

3 Detailed Design of Proposed Framework

3.1 Overview

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

In the offline stage, we first develop the DEF algorithm to generate the extractor $\mathbf{E}$. Then, we develop the NestDFE algorithm to divide $\mathbf{E}$ into multiple sub-extractors and form these sub-extractors into a single multi-capacity extractor. we use the multi-capacity extractor to extract discriminative features from the image data set and form multiple feature sets. In addition, we send the multi-capacity extractor to the mobile device. Because $\mathbf{E}$ has explored the distribution of the discriminative features of the image data set, and $\mathbf{E}$ is equivalent to the multi-capacity extractor, the multi-capacity extractor can extract discriminative features. Although the size of the multi-capacity extractor is larger than the size of the extracted features, the proposed framework can reduce network traffic. This is because mobile users can frequently use the multi-capacity extractor to extract discriminative features. Note that, $\mathbf{E}'s$ update and NestDFE's operation are both offline.

In the online stage, after capturing an image, the mobile device first pre-processes it. For instance, performing the objection detection algorithm (*e.g.*, histograms of oriented gradient [30]) to obtain objects, graying the obtained objects, and aligning the dimension of the object to the dimension of $\mathbf{E}$. Then, the mobile device selects an appropriate sub-extractor to extract discriminative features from the pre-processed image data. Because a larger sub-extractor can extract more effective discriminative features, thereby obtaining higher recognition accuracy. To achieve the highest recognition accuracy in the current situation, the mobile device should select the maximum capacity sub-extractor it can currently support. Thus, the mobile device can dynamically select appropriate sub-extractors to extract discriminative features based on its available resources. Next, the mobile device uploads the extracted feature data to the edge server. After receiving the feature data, the edge server performs feature matching to obtain results and sends the results back to the mobile user. It is worth noting that feature matching can save a considerable amount of time because the dimensions of the feature set are much lower than the dimensions of the image data set.

3.2 DFE Algorithm

The goal of the DFE algorithm is to generate the extractor $\mathbf{E}$ to extract effective discriminative features from the image data set on the edge server and images on mobile devices. However, we cannot test the performance of $\mathbf{E}$ directly on images on the mobile device. Based on the empirical risk minimization [31], we can measure the performance of $\mathbf{E}$ on the image data set. One of the most important performance metric of $\mathbf{E}$ is whether it can extract effective discriminative features from the image data, which is reflected by the recognition accuracy in this paper. If the extracted discriminative features can be easily distinguished, it means that these discriminative features are effective, and using them can obtain higher recognition accuracy. Otherwise, the extracted discriminative features are not effective, and using them will obtain lower recognition accuracy. Following this idea, we first design the DFE algorithm, which consists of two phases:

3.2.1 Constructing intra-class and inter-class structures

The DFE algorithm should contain intra-class and inter-class structures, as they all contribute to generating $\mathbf{E}$. Fig. 3 illustrates the intra-class and inter-class structures and divides them into four types of substructures. The left

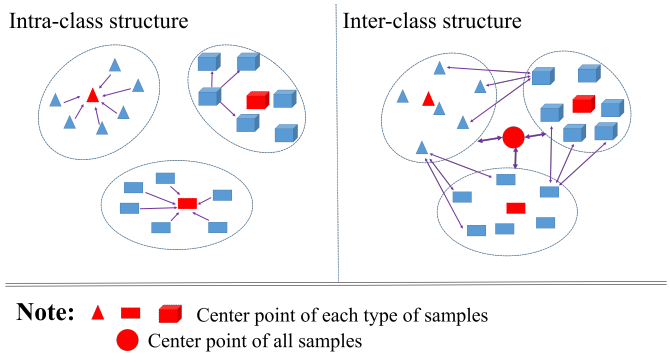


Fig. 3. Illustration of intra-class and inter-class structures.

side of Fig. 3 illustrates that the intra-class structure contains two types of substructures. One is the substructure between the sample (*e.g.*, blue triangles) and the center of samples (*e.g.*, the red triangle). The other is the substructure between samples with the same label (*e.g.*, blue cubes). The right side of Fig. 3 illustrates that the inter-class structure also contains two types of substructures. One is the substructure between the sample centers (*e.g.*, the red triangle, the red rectangle, and the red cube). The other is the substructure between samples with different labels (*e.g.*, the blue triangle and blue rectangles). However, most existing feature extraction algorithms only contain a portion of these substructures [23], [27], [32], [33], which affects the ability of the extractors they generate to extract discriminative features.

To address this problem, the DFE algorithm includes all of these substructures. In addition, it is not appropriate to directly use the Euclidean Distance (EC) [34] to measure the substructure information of image data. This is because, in general, the EC between samples from the same class label is small and the EC between samples from different class labels is large. The distance gap between the two types of samples leads to optimization difficulties. Inspired by the idea of [35] that residual networks are easier to optimize, we aim to narrow the distance gap between the two types of samples by designing a similarity function. In addition, the relationship between samples from the same class label is more important for generating efficient extractors. Therefore, we construct a novel similarity function to expand the distance between samples from the same class label and shrink the distance between samples from different class labels. We define the similarity function as follows:

$$S(i,j) = \begin{cases} d_n * exp(d_n + 1), y_i = y_j \\ d_n * exp(-d_n - 1), y_i \neq y_j \end{cases}, \tag{1}$$

where $d_n = \frac{d(i,j)-d(i,j)_{min}}{d(i,j)_{max}-d(i,j)_{min}}$, $d(i,j) = \sqrt{\sum_{l=1}^{d}(\mathbf{x}_i^l - \mathbf{x}_j^l)^2}$. $d(i,j)_{min}$ and $d(i,j)_{max}$ represent the minimum and maximum of the distance between samples $\mathbf{x}_i$ and $\mathbf{x}_j$. We normalize the distance between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ because their distances vary greatly and are difficult to optimize.

Fig. 4 illustrates the similarity function. As shown, the zoom range for these two types of distance is different. Specifically, the similarity function expands the distance between samples from the same class label to $[0, e^2]$ and shrinks the distance between samples from different class labels to $[0, 1]$. Thus, we can easily optimize them. Based on

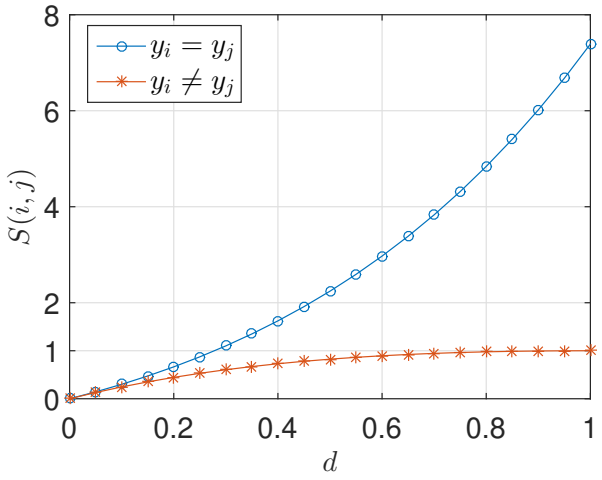


Fig. 4. The similarity function.

the similarity function and Fig. 3, we define and quantify these substructures as follows:

The global intra-class substructure represents the structure between samples and the sample center of the same class label. It is quantified as follows:

$$\mathcal{O}_{gw} = \sum_{k=1}^{K}\sum_{i=1}^{N_k} \mathbf{E}^T(\mathbf{x}_i^k - \frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i)(\mathbf{x}_i^k - \frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i)^T\mathbf{E}, \tag{2}$$

where N_k indicates the number of samples whose class label is k. $\mathbf{x}_i^k$ is the i-th sample in class k.

The local intra-class substructure represents the structure between samples with the same class label. It is quantified as follows:

$$\mathcal{O}_{lw} = \sum_{ij}^{N} ||\mathbf{E}^T\mathbf{x}_i - \mathbf{E}^T\mathbf{x}_j||^2 W_{ij}^w, \tag{3}$$

where $W_{ij}^w = S(i,j)$, if and only if $i \in \mho_{k_1}^w(j)$ or $j \in \mho_{k_1}^w(i)$; otherwise, $W_{ij}^w = 0$. $\mho_{k_1}^w(i)$ is calculated by Eq. (1) and is the index set of the k_1 nearest neighbors of sample $\mathbf{x}_i$ with the same class label.

The global inter-class substructure represents the structure between the center sample of each class of sample and the center sample of all samples. It is quantified as follows:

$$\mathcal{O}_{gb} = \sum_{k=1}^{K} N_k \mathbf{E}^T(\frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i)(\frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i)^T\mathbf{E}. \tag{4}$$

The local inter-class substructure represents the structure between samples with different class labels. It is quantified as follows:

$$\mathcal{O}_{lb} = \sum_{ij}^{N} ||\mathbf{E}^T\mathbf{x}_i - \mathbf{E}^T\mathbf{x}_j||^2 W_{ij}^b, \tag{5}$$

where $W_{ij}^b = S(i,j)$, if and only if $i \in \mho_{k_2}^b(j)$ or $j \in \mho_{k_2}^b(i)$; otherwise $W_{ij}^b = 0$. $\mho_{k_2}^b(i)$ is calculated by Eq. (1) and is the index set of k_2 nearest neighbors of sample $\mathbf{x}_i$ with different class labels.

3.2.2 Optimization

In order for the extractor $\mathbf{E}$ to extract effective discriminative features, we minimize $\mathcal{O}_{gw}$ and $\mathcal{O}_{lw}$, and maximize $\mathcal{O}_{gb}$ and $\mathcal{O}_{lb}$. The purpose of minimizing $\mathcal{O}_{gw}$ and $\mathcal{O}_{lw}$ is to make the distance between the discriminative features extracted from the images of the same class close. The purpose of maximizing $\mathcal{O}_{gb}$ and $\mathcal{O}_{lb}$ is to make the distance between the discriminative features extracted from the images of different classes far. By simultaneously minimizing the intra-class substructures and maximizing the inter-class substructures, we can obtain discriminative features that are easily distinguishable.

However, the contribution of different substructures may be quite different to generate the extractor when dealing with different image data sets. To circumvent this problem, we introduce three trade-off parameters α, β and γ to control the importance between $\mathcal{O}_{gw}$ and $\mathcal{O}_{lw}$, $\mathcal{O}_{gb}$ and $\mathcal{O}_{lb}$, and $\alpha\mathcal{O}_{gw}$+$(1-\alpha)\mathcal{O}_{lw}$ and $\beta\mathcal{O}_{gb}$+$(1-\beta)\mathcal{O}_{lb}$, respectively. We define the objective function as follows:

$$\begin{aligned}\mathcal{F}_{DFE} = \max_{\mathbf{E}}\{\gamma[\beta\mathcal{O}_{gb} + (1-\beta)\mathcal{O}_{lb}] \\ - (1-\gamma)[\alpha\mathcal{O}_{gw} + (1-\alpha)\mathcal{O}_{lw}]\}, \\ s.t.\ \mathbf{E}^T\mathbf{E} = \mathbf{I}\end{aligned} \tag{6}$$

where $\alpha, \beta, \gamma \in [0, 1]$. To gain more insight, Eq. (6) can be rewritten as:

$$\mathcal{O}_{DFE} = \arg\max_{\mathbf{E}} tr(\mathbf{E}^T\mathbf{\Phi}\mathbf{E}) \ \ s.t.\ \mathbf{E}^T\mathbf{E} = \mathbf{I}, \tag{7}$$

where

$$\begin{aligned}\mathbf{\Phi} = \gamma\beta\sum_{k=1}^{c}\sum_{i=1}^{N_k}(\mathbf{x}_i^k - \frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i)(\mathbf{x}_i^k - \frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i)^T - (1-\gamma) \\ \alpha\sum_{k=1}^{c}N_k(\frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i)(\frac{1}{N_k}\sum_{i=1}^{N_k}\mathbf{x}_i - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i)^T \\ + \sum_{ij}^{N}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T[\gamma(1-\beta)W_{ij}^b - (1-\alpha)(1-\gamma)W_{ij}^w]\end{aligned} \tag{8}$$

The extractor $\mathbf{E}$ can extract effective discriminative features because the DFE algorithm includes all substructures and adapts each substructure by introducing three trade-off parameters. In addition, the $\mathbf{E}$ is easy to be generated because the DFE algorithm narrows the distance gap between the two types of samples, as shown in Eq. (8).

Based on [32], [36], we switch Eq. (7) to a simple eigenvalue and eigenvector problem for matrix $\mathbf{\Phi}$. Based on the Lagrange multiplier [37], Eq. (7) forms the Lagrangian function: $\zeta(\mathbf{E}, \mathbf{\Lambda}) = tr(\mathbf{E}^T\mathbf{\Phi}\mathbf{E}) - tr(\mathbf{\Lambda}(\mathbf{E}^T\mathbf{E} - \mathbf{I}))$, where $\mathbf{\Lambda} = [\lambda_1, \ldots, \lambda_n]$. We have $\mathbf{\Phi}\mathbf{e}_i = \lambda_i\mathbf{e}_i$ by setting $\frac{\partial\zeta(\mathbf{E},\mathbf{\Lambda})}{\partial\mathbf{E}} = 0$. Thus, Eq. (7) can be rewritten as:

$$\mathcal{O}_{DFE} = \arg\max_{\mathbf{e}_i}\sum_{i=1}^{d}\mathbf{e}_i^T\mathbf{\Phi}\mathbf{e}_i = \arg\max_{\mathbf{e}_i}\sum_{i=1}^{d}\lambda_i. \tag{9}$$

From Eq. (8), $\mathbf{\Phi}$ is a non-positive real symmetric matrix and its eigenvalues can be positive, zero, or negative. To optimize Eq. (9), we choose all positive eigenvalues of $\mathbf{\Phi}$. Assuming that the number of positive eigenvalues of $\mathbf{\Phi}$ is r, the solution of Eq. (7) is $\mathbf{E} = [\mathbf{e}_1, \cdots, \mathbf{e}_r]$.

The extractor $\mathbf{E}$ consists of the eigenvectors corresponding to the top r positive eigenvalues of $\mathbf{\Phi}$. Fig. 5 and Fig. 6 illustrate the relationship between recognition accuracy, number of eigenvectors and eigenvalues on COIL20 and UMIST data sets. As shown, when $\mathbf{E}$ consists of eigenvectors corresponding to the r top positive eigenvalues, the DFE algorithm achieves the highest recognition accuracy. This indicates that $\mathbf{E}$ can extract effective discriminative features. However, the recognition accuracy of the DFE algorithm tends to be stable or even decreased when the eigenvectors corresponding to the zero or negative eigenvalues are selected. Hence, the dimension of $\mathbf{E}$ can be estimated as being equal to the number of positive eigenvalues of $\mathbf{\Phi}$.

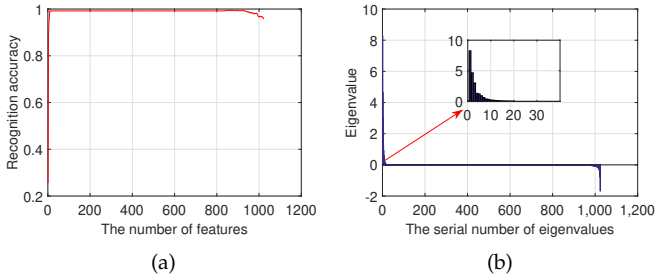


Fig. 5. Relationship between recognition accuracy, number of eigenvectors and eigenvalues on the COIL20 data set.

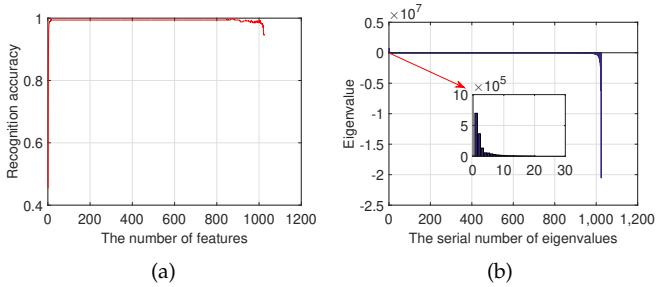


Fig. 6. Relationship between recognition accuracy, number of eigenvectors and eigenvalues on the UMIST data set.

In addition, Fig. 5 and Fig. 6 indicate that there are only a few positive eigenvalues. This means that the dimension of $\mathbf{E}$ is low because it only consists of eigenvectors corresponding to positive eigenvalues. Hence, the $\mathbf{E}$ extracts a few discriminative features from the image data set on the edge server and images on the mobile device. This has two advantages: (i) Network traffic is reduced because the mobile device only needs to upload a small amount of feature data to the edge server. (ii) Feature matching time is reduced because we only need to match a small number of discriminative features. In addition, the DFE algorithm saves a lot of manpower costs because it estimates the optimal dimension of the feature set.

3.3 NestDFE Algorithm

In real-world scenarios, the amount of available resources of the mobile device dynamically changes because the mobile device typically runs multiple applications simultaneously, and the mobile user may frequently launch new applications and close existing ones. Note that, the available resources are RAM resources, and we can use some tools (*e.g.*, memorymoniter) to easily measure the currently available resources of mobile devices. If, for a period of time, the mobile

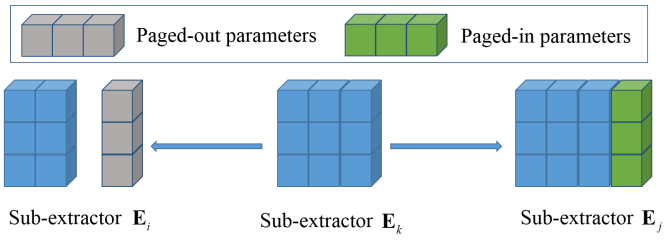


Fig. 7. Illustration of the multi-capacity extractor switching. The order of the capacity of these sub-extractors are $\mathbf{E}_i < \mathbf{E}_k < \mathbf{E}_j$.

device does not have sufficient resources to support the extractor operation, the extractor $\mathbf{E}$ cannot work and cause the mobile image recognition application to be inoperable, thereby affecting the user experience.

As introduced in Section 3.2, the optimal extractor $\mathbf{E}$ consists of $[\mathbf{e}_1, \cdots, \mathbf{e}_r]$. Inspired by this, a straightforward solution is to divide the extractor $\mathbf{E}$ into multiple possible sub-extractors and store all these sub-extractors. For example, we divide extractor $\mathbf{E}$ into a series of sub-extractors, such as $\mathbf{E}_1 = [\mathbf{e}_1]$, $\mathbf{E}_2 = [\mathbf{e}_1, \mathbf{e}_2]$, $\mathbf{E}_3 = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$, $\cdots$, $\mathbf{E}_r = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_r]$, and store them on the mobile device. Then, we deploy the memorymoniter to monitor the currently available resources of mobile devices and map each sub-extractor to the resources it needs. Therefore, the mobile device can dynamically select appropriate sub-extractor based on its available resources. Note that since the image has been pre-processed when using the extractor to extract discriminative features, the RAM resources required to extract features are only related to the capacity of the extractor. We can use the detection tools offline to measure the RAM resources required by each extractor. In addition, since Section 3.2 reveals the relationship between the number of discriminative features and the recognition accuracy, we can obtain the relationship between the required resources and the recognition accuracy. Therefore, we can use a table to record the mapping between each extractor, the required RAM resources, and the recognition accuracy.

However, it is not wise to store all of these sub-extractors because they take up a considerable amount of memory space. In addition, running multiple extractor operations on the mobile device may cause a considerable amount of switching overhead because the mobile device needs to switch the entire sub-extractor. For example, assume that the mobile image application continues to use $\mathbf{E}_2$ to extract discriminative features. If a new application with a higher priority is suddenly launched, the available resources of the mobile device may no longer meet the needs of $\mathbf{E}_2$. In order to accommodate the available resources, the mobile device needs to page out $\mathbf{E}_2$ and page in $\mathbf{E}_1$.

To circumvent this problem, we propose a NestDFE algorithm for generating a single multi-capacity extractor that is equivalent to storing all sub-extractors. As analyzed above, these sub-extractors are not independent and the sub-extractor with a smaller capacity shares all its parameters to the sub-extractor with a larger capacity, *i.e.*, $\mathbf{E}_i \subset \mathbf{E}_j$, where $1 \leq i < k < j \leq r$. Thus, the sub-extractor $\mathbf{E}_i$ can nest itself within the sub-extractor $\mathbf{E}_j$ without occupying additional memory space. Inspired by this, in the NestDFE algorithm, the extractor $\mathbf{E}$ is divided into r sub-extractors, *i.e.*, $\mathbf{E}_1 = [\mathbf{e}_1]$, $\mathbf{E}_2 = [\mathbf{e}_1, \mathbf{e}_2]$, $\mathbf{E}_3 = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$, $\cdots$, $\mathbf{E}_r = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_r]$. Then, the NestDFE enables the sub-extractors with smaller capacity to be nested into the sub-extractors with larger capacity. Therefore, we only need to store $\mathbf{E}_r$ on the mobile device. This is because all other sub-extractors are nested in $\mathbf{E}_r$. $\mathbf{E}_r$ plays the role of the multi-capacity extractor, which can realize all the functions of other sub-extractors.

Each sub-extractor nested in $\mathbf{E}_r$ can extract different numbers of discriminative features, so as to obtain the corresponding recognition accuracy. When selecting a sub-extractor with a larger capacity, we can use it to extract more effective discriminative features, and more discriminative features mean higher recognition accuracy. Therefore, the highest recognition accuracy in the current situation can be obtained by selecting the maximum capacity sub-extractor that the mobile device's currently available resource can support. When capturing an image, the mobile device first pre-processes it. Then, the mobile device selects the maximum capacity sub-extractor that can be supported by its currently available resources to extract discriminative features from the pre-processed image data. Follow that, the mobile device uploads the extracted feature data to the edge server to perform feature matching. Finally, the edge server sends the results back to the mobile user.

As shown in Fig. 7, it is assumed that the mobile image application uses extractor $\mathbf{E}_k$ to extract discriminative features. At some point, the available resources of the mobile device may not be able to meet the needs of extractor $\mathbf{E}_k$ because of the launch of some new applications with higher priority. The mobile device has to switch $\mathbf{E}_k$ to $\mathbf{E}_i$ to adapt its available resources. Thus, the mobile device incurs zero page-in overhead, and only needs to page out the parameters that $\mathbf{E}_i$ does not have (marked as gray squares). On the contrary, at some point, the available resources of the mobile device are enough because some applications are closed. The mobile device can switch $\mathbf{E}_k$ to $\mathbf{E}_j$. Thus, the mobile device incurs zero page-out overhead, and only needs to page in the parameters included in $\mathbf{E}_j$ (marked as green squares). Assume that the number of parameters of $\mathbf{E}_i$ is $P(\mathbf{E}_i)$, and the number of parameters of $\mathbf{E}_j$ is $P(\mathbf{E}_j)$. When switching from the smallest-capacity sub-extractor (*e.g.*, $\mathbf{E}_i$) to the largest-capacity sub-extractor (*e.g.*, $\mathbf{E}_j$), the upper-bound of the overhead of page-in parameters is $P(\mathbf{E}_j) - P(\mathbf{E}_i)$, and the overhead of page-out parameters is 0. When switching from $\mathbf{E}_j$ to $\mathbf{E}_i$, the upper-bound of the overhead of page-out parameters is $P(\mathbf{E}_j) - P(\mathbf{E}_i)$, and the overhead of page-in parameters is 0. Compared with page-in and page-out the entire sub-extractor, the multi-capacity extractor reduces the overhead of page-in and page-out sub-extractors. In addition, since the size of the multi-capacity extractor is smaller than the size of the accumulated sub-extractors, the former saves a lot of memory space.

In summary, by using $\mathbf{E}$ to extract discriminative features and deploying the multi-capacity extractor on the mobile device, the proposed framework can achieve higher recognition accuracy and shorter feature matching time, generate less network traffic and consume less memory space. In addition, the proposed framework can be generalized to support many other feature extraction algorithms, such as joint global and local structure discriminant analy-

TABLE 1
Description of benchmark data sets.

Data Sets	# Images	# Dimensions	# Classes
COIL20	1440	1024	20
UMIST	564	1024	20
YALE	165	1024	15
ORL	400	1024	40
USPS	1854	256	10

sis [25] and locality adaptive discriminant analysis [27].

4 EVALUATION

4.1 Experimental Setup

We conduct experiments on five benchmark data sets: COIL20 [38], UMIST [39], YALE [40], ORL [41], USPS [42]. Table 1 lists the details used in the experiment.

We compare the DFE algorithm with four algorithms, which represent the best performing algorithms currently in use. They are: local binary pattern (LBP) [22], principle component analysis (PCA) [23], joint global and local structure discriminant analysis (JGLDA) [25], and locality adaptive discriminant analysis (LADA) [27]. In addition, we evaluate the PCA in the case of edge guidance, namely EG-PCA. That is, we first perform the PCA on the training set to generate the extractor on the edge server. Then, we send the extractor to mobile device to extract features from the test set.

For our DFE algorithm, the trade-off parameters α, β and γ can be tuned as follows. Because α, β and γ all need to be tuned, we fix two of them to tune the remaining one. For instance, when tuning α, we first fix β and γ and set $\beta = \gamma = 0.5$. After obtaining α (assuming α=0.1), we fix α and γ and set α=0.1 and γ=0.5 to tune β. After obtaining β (assuming β = 0.9), we fix α and β and set α=0.1 and β=0.9 to tune γ. To maintain the recognition accuracy of LBP and PCA, we extract 128 features for LBP and extract 100 features for PCA. For DFE and JGLDA, we set $k_1 = k_2 = 1$ to preserve local intra-class and inter-class structures. In addition, we determine the number of features of JGLDA and LADA based on their recognition accuracy. That is, we first calculate all the recognition accuracy corresponding to their feature values. Then, we select the number of features with the highest recognition accuracy as the best dimension of the extractor. For our DFE algorithm, we choose the number of positive eigenvalues as the dimension of the extractor.

We randomly split the COIL20 data set at the ratio of 1:1 to form a training set and a test set. That is, the COIL20 data set contains 1440 images, we randomly select 720 images as the training set (36 images per class), and the remaining 720 images are used as the test set. We randomly split the UMIST and ORL data sets at the ratio of 3:2 to form the training set and the test set, and randomly split the YALE and USPS data sets at the ratio of 9:1 to form the training set and the test set. Finally, we use the nearest neighbor classifier to evaluate all algorithms and report the result of 20 averages .

4.2 System Implementation

We implement a prototype system. The experimental environment consists of two components: mobile device and mobile edge computing platform. A Huawei Honor 8 smartphone is regarded as the mobile device. The mobile edge computing platform consists of a base station and three edge servers, e.g., edge server A, edge server B and edge server C. Note that edge server A, B and C are three desktops. The edge server A is responsible for providing computing and memory resources. The base station is based on the Open Air Interface (OAI) [43], and is responsible for communicating with the mobile device. The base station consists of three components: radio-frequency signal generator, edge server B, and edge server C. The radio-frequency signal generator is equipped with USRP-B210. The edge server B is responsible for running the eNodeB. The radio-frequency signal generator and edge server B are connected via USB 3.0. The edge server C is responsible for running Home Subscriber Service (HSS), Mobility Management (MME), Serving Gateway (SGW), and Packet data network Gateway (PGW) [43]. The upload link rate of mobile devices connect to mobile edge computing platform is 1000 KB/s and its download link rate is 1.36 MB/s. Note that, the base station is next to the edge server.

We take the COIL20 data set as an example, and the system flow is: we first use the DFE algorithm to generate the extractor $\mathbf{E}$ on the edge server. Then, we run the NestDFE algorithm to turn $\mathbf{E}$ into a single multi-capacity extractor. Follow that, we use the single multi-capacity extractor to extract discriminative features from the COIL20 data set and form a discriminative feature set named DF-COIL20. Then, the edge server sends the single multi-capacity extractor to the mobile device. When capturing an image, the mobile device first pre-processes it and then selects an appropriate sub-extractor to extract discriminative features. Follow that, the mobile device uploads the extracted discriminative feature data to the edge server. Then, the edge server matches it with the DF-COIL20 and chooses the most similar feature data and returns its label. Finally, the edge server returns the label data back to the mobile device.

4.3 Experimental Results

We evaluate the proposed framework from the aspects of recognition accuracy, network traffic, feature matching time, memory space, and switching overhead. Recognition accuracy and network traffic are two commonly used indicators. Feature matching time affects the response time of mobile image recognition applications. Memory space and switching overhead affect the resource usage of mobile devices.

4.3.1 Recognition Accuracy

Our DFE algorithm outperforms other algorithms on all data sets. As shown in Table 2, on the YALE data set, DFE improves recognition accuracy by 22.65% compared with LBP; compared with LADA, DFE improves recognition accuracy by 2.2%. The reason is that DFE can extract effective discriminative features. It is worth noting that the recognition accuracy of LADA and JGLDA is higher than that of LBP and EG-PCA. This is because LBP and EG-PCA extract numerous features to preserve the intrinsic structure

TABLE 2
Recognition accuracy of different algorithms on different data sets.

	COIL20	UMIST	YALE	ORL	USPS
LBP	95.56	97.75	70.67	90.63	83.52
PCA	59.00	54.03	22.20	59.58	65.93
EG-PCA	**96.67**	**98.12**	**86.67**	**90.80**	**85.71**
JGLDA	98.61	98.20	90.67	91.87	87.91
LADA	97.64	98.65	91.12	95.00	89.56
DFE	**99.31**	**99.55**	**93.32**	**96.25**	**93.96**

of the image data, rather than extracting discriminative features. However, LADA and JGLDA extract discriminative features. In addition, our algorithm has a higher recognition accuracy than JGLDA and LADA. This is because our algorithm uses the similarity function to construct substructures and controls these substructures reasonably. Therefore, our algorithm can achieve the highest recognition accuracy.

It is important to use the image data set to guide the feature extraction of images. As shown in Table 2, on all data sets, the recognition accuracy of EG-PCA is higher than that of PCA. For example, the recognition accuracy of EG-PCA on the COIL20 and USPS data sets is 37.67% and 19.78% higher than that of PCA, respectively. This is because the extractor generated by EG-PCA explores the distribution of the features of the image data set. When using the extractor, it is easy to extract useful features. However, when using PCA, the extracted features from the image data set and images are separate, and it is difficult to extract features that are useful for recognition tasks. Hence, EG-PCA achieves higher recognition accuracy. In addition, it indicates that the guidance of the image data set is important.

In most cases, the recognition accuracy of JGLDA is lower than that of LADA. As shown in Table 2, on the ORL data set, JGLDA's recognition accuracy is 3.13% lower than LADA's recognition accuracy. Although the JGLDA consists of all substructures, its recognition accuracy is lower than that of the LADA. This is because the JGLDA cannot reasonably control the contribution of each substructure when generating the extractor. Thus, some features that are detrimental to recognition tasks are also extracted and result in low recognition accuracy. This indicates that proper control of each substructure is important to generate the extractor to extract effective discriminative features.

4.3.2 Network Traffic

Our algorithm has minimal network traffic in all data sets. Fig. 8 shows that on the YALE data set, the DFE reduces network traffic by 76.09% compared with LBP. The reason is that the DFE can extract fewer and effective discriminative features. Hence, when using the DFE to extract discriminative features from images, the mobile device generates a small amount of network traffic. This also indicates that under the guidance of the edge, the mobile device generates less network traffic.

LADA and JGLDA have less network traffic than LBP and PCA. Fig. 8 shows that on the COIL20 data set, LADA reduces network traffic by 42.71% compared with LBP, and LADA reduces network traffic by 30.96% compared with

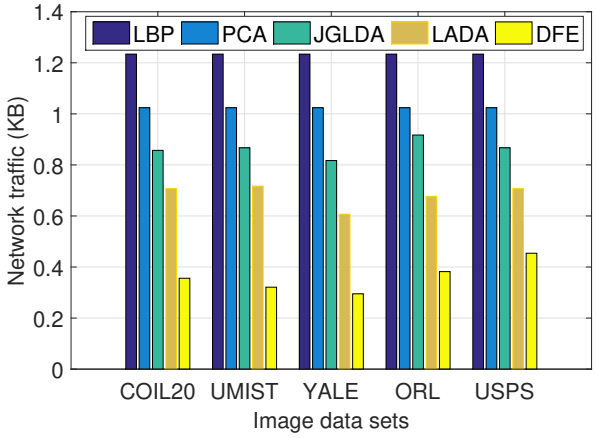
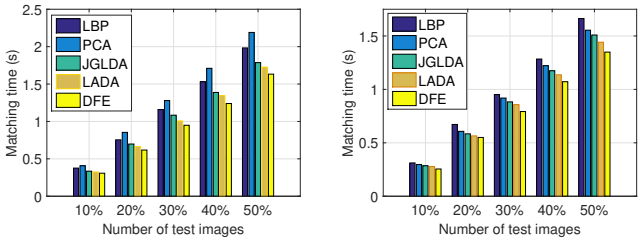


Fig. 8. Network traffic for different algorithms.



(a) The COIL20 data set (b) The USPS data set

Fig. 9. Matching time for different number of images.

PCA. The reason is that LBP and PCA extract numerous features to preserve the intrinsic structure of the image data. However, our algorithm, LADA, and JGLDA extract fewer discriminative features, thereby consuming less network traffic.

4.3.3 Feature Matching Time

Our DFE has the least feature matching time. This is because the DFE extracts a small number of features (as analyzed in Section 4.3.2). Therefore, with the same number of images, fewer features lead to less feature matching time. As shown in Fig. 9. In addition, LADA and JGLDA have less feature matching time than LBP and PCA. The reason is that LBP and PCA extract numerous features. The number of extracted discriminative features of LADA and JGLDA is less than the number of features extracted by LBP and PCA. Hence, LADA and JGLDA have less feature matching time.

4.3.4 Reduction on Memory Space

We divide the extractor of each data set into multiple sub-extractors based on the size of the extractor and the corresponding recognition accuracy, as shown in Table 3. In addition, COIL20* indicates that we divide the extractor on the COIL20 data set into all possible sub-extractors. We have the following observations.

The size of the multi-capacity extractor is smaller than the size of the accumulated extractors for each data set. As illustrated in Table 4, compared with the accumulated extractors, the multi-capacity extractor reduces memory space by 48.72% and 69.58% in the COIL20 and ORL data

TABLE 3
Sub-extractors of each data set. (num, vec, si, acc) denotes a sub-extractor, where num, vec, si, and acc represent the sequence number, feature number, size, and recognition accuracy.

Data Sets	Multiple Sub-extractors
COIL20	(1, 1:3, 23.2KB, 0.879); (2, 1:6, 46.2KB, 0.957) (3, 1:9, 69.3KB, 0.991); (4, 1:19, 146KB, 0.992)
COIL20*	(1, 1:1, 7.87KB, 0.256); (2, 1:2,15.5KB, 0.689) (3, 1:3, 23.2KB, 0.879); (4, 1:4, 30.8KB, 0.921) (5, 1:5, 38.5KB, 0.941); (6, 1:6, 46.2KB, 0.957) (7, 1:7, 53.9KB, 0.963); (8, 1:8, 61.6KB, 0.964) (9, 1:9, 69.3KB, 0.991); (10, 1:19, 146KB, 0.992)
UMIST	(1, 1:3, 23.2KB, 0.9045); (2, 1:6, 46.2KB, 0.9851) (3, 1:9, 69.3KB, 0.9881); (4, 1:12, 92.3KB, 0.991) (5, 1:15, 115KB, 0.997)
YALE	(1, 1:3, 23.2KB, 0.4889); (2, 1:6, 46.3KB, 0.7111) (3, 1:9, 69.3KB, 0.8222); (4, 1:12, 92.4KB, 0.9333)
ORL	(1, 1:5, 38.6KB, 0.85); (2, 1:10, 77KB, 0.9458) (3, 1:15, 118KB, 0.95); (4, 1:20, 153KB, 0.956) (5, 1:22, 169KB, 0.9667)
USPS	(1, 1:5, 9.8KB, 0.7857); (2, 1:10, 19.4KB, 0.9066) (3, 1:15, 29KB, 0.9231); (4, 1:20, 38.6KB, 0.9341) (5, 1:30, 57.9KB, 0.9560)

TABLE 4
Multi-capacity and accumulated extractors on memory space reduction.

Data Sets	Multi-capacity Extractor Size (KB)	Accumulated Extractor Size (KB)	Reduced Memory Space
COIL20	146	284.7	48.72%
COIL20*	146	492.77	70.37%
UMIST	115	346	66.76%
YALE	92.4	231.2	60.03%
ORL	169	555.6	69.58%
USPS	57.9	154.7	62.57%

sets, respectively. The reason is that in the multi-capacity extractor, the sub-extractor having a smaller capacity is nested in the sub-extractor having a larger capacity. Thus, the sub-extractor having a larger capacity shares memory space with the sub-extractor having a smaller capacity, and the size of the multi-capacity extractor is equal to the size of the sub-extractor having a larger capacity. However, the size of accumulated extractors is equal to the sum of the size of each extractor. Hence, the size of the multi-capacity extractor is much smaller.

The multi-capacity extractor saves more memory space when the optimal extractor divides more sub-extractors. As shown in Table 4, the memory space of the multi-capacity is reduced by 70.37% in the COIL20*. This is because, in the multi-capacity extractor, each sub-extractor is not independent. The size of all sub-extractors is equal to the single multi-capacity extractor. However, the size of these accumulated sub-extractors is larger because there are multiple sub-extractors and their memory space is independent.

TABLE 5
Multi-capacity and independent extractors on extractor switching (extractor upgrade) in terms of memory usage.

Data Sets	Multi-capacity Extractor Upgrade Overhead (KB)		Independent Extractor Upgrade Overhead (KB)	
	Page-in	Page-out	Page-in	Page-out
COIL20	122.8	0	146	23.2
COIL20*	7.7	0	69.3	61.6
UMIST	91.8	0	115	23.2
YALE	69.2	0	92.4	23.2
ORL	130.4	0	169	38.6
USPS	48.1	0	57.9	9.8

TABLE 6
Multi-capacity and independent extractors on extractor switching (extractor downgrade) in terms of memory usage.

Data Sets	Multi-capacity Extractor Downgrade Overhead (KB)		Independent Extractor Downgrade Overhead (KB)	
	Page-in	Page-out	Page-in	Page-out
COIL20	0	122.8	23.2	146
COIL20*	0	7.7	61.6	69.3
UMIST	0	91.8	23.2	115
YALE	0	69.2	23.2	92.4
ORL	0	130.4	38.6	169
USPS	0	48	9.8	57.8

4.3.5 *Reduction on Switching Overhead*

The average page-in and page-out memory usage of the multi-capacity extractor is less than independent extractors for each data set during extractor switching. Table 5 shows that on the COIL20 data set, when upgrading the sub-extractor, the multi-capacity extractor needs to page in 122.8KB and page out 0KB parameters. However, the independent sub-extractor needs to page in 146KB and page out 23.2KB parameters. Similar, Table 6 shows that on the COIL20 data set, when downgrading the sub-extractor, the multi-capacity extractor needs to page in 0KB and page out 122.8KB parameters. The independent extractor needs to page in 23.2KB and page out 146KB parameters. This is because during the extractor upgrade, the multi-capacity extractor only needs to page in the parameters included in the sub-extractor having a larger capacity. In addition, the multi-capacity extractor does not need to page out parameters because the parameters are used by the next sub-extractor. When the extractor is downgraded, the multi-capacity extractor only needs to page out the extra parameters that the sub-extractor with a smaller capacity does not have. Whether the extractor is upgrading or downgrading, the independent extractor needs to page in and page out the entire sub-extractor. Therefore, the multi-capacity can reduce the switching overhead.

The multi-capacity extractor saves more switching overhead when the extractor divides more sub-extractors. As shown in Table 5 and Table 6, when upgrading the extractor, the multi-capacity extractor only needs to page in 7.7KB parameters. When downgrading the extractor, the multi-

capacity extractor only needs to page out 7.7KB parameters. This is because when the the optimal extractor is divided into multiple sub-extractors, the adjacent sub-extractors have similar sizes. When upgrading or downgrading adjacent sub-extractors, the multi-capacity extractor only needs to page in and page out a few parameters. However, the independent extractor needs to page in and page out entire sub-extractors. Therefore, the independent extractor consumes more switching overhead. In addition, the multi-capacity extractor can save more switching overhead as the extractor switching frequency increases.

5 Conclusion

In this paper, we mainly make the following three contributions. First, we propose a resource-aware feature extraction framework. Second, we propose the DFE algorithm to generate the extractor to extract effective discriminative features. Third, we propose the NestDFE algorithm, which generates a single multi-capacity extractor that works equally well with a serials of sub-extractors. Experimental results show that the proposed framework outperforms several state-of-the-art algorithms in terms of recognition accuracy, network traffic, memory space and switching overhead.

In future work, we will study the frequency of updating the extractor. The extractor is important because it affects the recognition accuracy and network traffic of the proposed framework. Since the extractor is related to the amount of training data, we plan to study the relationship between the amount of training data and recognition accuracy to determine how often the extractor is updated.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61922017, 61902036), and the Funds for Creative Research Groups of China (61921003).

References

[1] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "Deepcache: Principled cache for mobile deep vision," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 129–144.

[2] L. Xie, J. Sun, Q. Cai, C. Wang, J. Wu, and S. Lu, "Tell me what i see: Recognize rfid tagged objects in augmented reality systems," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2016, pp. 916–927.

[3] B. Fang, X. Zeng, and M. Zhang, "Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proceedings of the ACM International Conference on Mobile Computing and Networking*, 2018, pp. 115–127.

[4] Y. Chen, J. Sun, X. Jin, T. Li, R. Zhang, and Y. Zhang, "Your face your heart: secure mobile face authentication with photoplethysmograms," in *Proceeding of the IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[5] W. Xu, Y. Shen, N. Bergmann, and W. Hu, "Sensor-assisted multi-view face recognition system on smart glass," *IEEE Transactions on Mobile Computing*, vol. 17, no. 1, pp. 197–210, 2018.

[6] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in *Proceedings of the IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[7] Z. Guan and T. Melodia, "The value of cooperation: Minimizing user costs in multi-broker mobile cloud computing networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 780–791, 2017.

[8] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2019.

[9] K.-C. Wu, W.-Y. Liu, and S.-Y. Wu, "Dynamic deployment and cost-sensitive provisioning for elastic mobile cloud services," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1326–1338, 2018.

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

[11] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2762–2773, 2018.

[12] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *Proceedings of the IEEE International Conference on Sensing, Communication, and Networking*, 2018, pp. 406–414.

[13] M. Ma, L. Zhang, J. Liu, Z. Wang, H. Pang, L. Sun, W. Li, G. Hou, and K. Chu, "Characterizing user behaviors in mobile personal livecast: Towards an edge computing-assisted paradigm," *ACM Transactions on Multimedia Computing, Communications and Application*, vol. 14, no. 3s, pp. 1–24, 2018.

[14] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[15] H. Wang, B. Kim, J. Xie, and Z. Han, "E-auto: A communication scheme for connected vehicles with edge-assisted autonomous driving," in *Proceedings of the IEEE International Conference on Communications*, 2019, pp. 1–6.

[16] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2018.

[17] M. Li, F. R. Yu, P. Si, and Y. Zhang, "Energy-efficient machine-to-machine (m2m) communications in virtualized cellular networks with mobile edge computing (mec)," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1541–1555, 2019.

[18] J. Li, Z. Peng, B. Xiao, and Y. Hua, "Make smartphones last a day: Pre-processing based computer vision application offloading," in *Proceedings of the IEEE International Conference on Sensing, Communication, and Networking*, 2015, pp. 462–470.

[19] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1910–1920, 2017.

[20] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan, "Cachier: Edge-caching for recognition applications," in *Proceedings of the IEEE International Conference on Distributed Computing Systems*, 2017, pp. 276–286.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[22] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[23] A. M. Martinez and A. C. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, 2001.

[24] M. Sugiyama, "Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1027–1061, 2007.

[25] Q. Gao, J. Liu, H. Zhang, X. Gao, and K. Li, "Joint global and local structure discriminant analysis," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 4, pp. 626–635, 2013.

[26] C. Ding and L. Zhang, "Double adjacency graphs-based discriminant neighborhood embedding," *Pattern Recognition*, vol. 48, no. 5, pp. 1734–1742, 2015.

[27] X. Li, M. Chen, F. Nie, and Q. Wang, "Locality adaptive discriminant analysis," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017, pp. 2201–2207.

[28] C. Liu, Y. Cao, Y. Luo, G. C. V. Kokkarane, Y. Ma, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure,"

IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 249–261, 2018.
[29] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. B. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proceedings of the IEEE Symposium on Computers and Communications*, 2012, pp. 59–66.
[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.
[31] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.
[32] W. Zhang, X. Xue, H. Lu, and Y.-F. Guo, "Discriminant neighborhood embedding for classification," *Pattern Recognition*, vol. 39, no. 11, pp. 2240–2243, 2006.
[33] S. Yan, D. Xu, B. Zhang, H. jiang Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40–51, 2007.
[34] T. E. Schouten and E. L. van den Broek, "Fast exact euclidean distance (feed): A new class of adaptable distance transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2159–2172, 2014.
[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*, 2016, pp. 770–778.
[36] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, USA: Johns Hopkins University Press, 1996.
[37] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge university press, 2004.
[38] "Columbia University Image Library," http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php, [Online; accessed 1-Jan-2020].
[39] "The Sheffield Face Database," https://www.sheffield.ac.uk/eee/research/iel/research/face, [Online; accessed 1-Jan-2020].
[40] "Yale Face Database," http://vision.ucsd.edu/content/yale-face-database, [Online; accessed 1-Jan-2020].
[41] "The Database of Faces," https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, [Online; accessed 1-Jan-2020].
[42] "Handwritten Digits USPS data set," https://www.kaggle.com/bistaumanga/usps-dataset, [Online; accessed 1-Jan-2020].
[43] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," *Computer Communication Review*, vol. 44, no. 5, pp. 33–38, 2014.

Chuntao Ding received the B.S. and M.S. degrees from SIAS International University in 2012 and Soochow University in 2015, respectively. He is currently a Ph.D. candidate at the State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include mobile edge computing, deep learning.

Ao Zhou received the Ph.D. degrees in Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an Associate Professor with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published 20+ research papers. She played a key role at many international conferences. Her research interests include Cloud Computing and Edge Computing.

Xiulong Liu is currently a professor in College of Intelligence and Computing, Tianjin University, China. Before that, he received the B.E. and Ph.D. degrees from Dalian University of Technology (China) in 2010 and 2016, respectively. He also worked as a visiting researcher in Aizu University, Japan; a postdoctoral fellow in The Hong Kong Polytechnic University, Hong Kong; and a postdoctoral fellow in the School of Computing Science, Simon Fraser University, Canada. His research interests include wireless sensing and communication, indoor localization, and networking, etc. His research papers were published in many prestigious journals and conferences including TON, TMC, TC, TPDS, TCOM, INFOCOM, and ICNP, etc. He received Best Paper Awards from ICA3PP 2014 and IEEE System Journal 2017. He is also the recipient of CCF Outstanding Doctoral Dissertation award 2017.

Xiao Ma received her Ph.D. degree in Department of Computer Science and Technology from Tsinghua University and B.S. degree in Telecommunication Engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2018 and 2013, respectively. She is currently a postdoctoral fellow at 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.

Shangguang Wang received his Ph.D degree at Beijing University of Posts and Telecommunications in 2011. He is Professor and Deputy Director at the State Key Laboratory of Networking and Switching Technology (BUPT). He has published more than 150 papers, and played a key role at many international conferences, such as general chair and PC chair. His research interests include service computing, cloud computing, and mobile edge computing. He is a senior member of the IEEE, and the Editor-in-Chief of the International Journal of Web Science.