Mathematical Biosciences and Engineering

MBE, 16(6): 6672–6682.
DOI: 10.3934/mbe.2019332
Received: 02 April 2019
Accepted: 19 June 2019
Published: 22 July 2019

http://www.aimspress.com/journal/MBE

Research article

A location-aware feature extraction algorithm for image recognition in mobile edge computing

Tianjun Lu[1,2,*], Xian Zhong[1], Luo Zhong[1] and RuiqiLuo[1]

[1] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
[2] School of Software, Nanyang Institute of Technology, Nanyang, Henan, 473004, China

* **Correspondence:** Email: lvtianjun@whut.edu.cn.

Abstract: With the explosive growth of mobile devices, it is feasible to deploy image recognition applications on mobile devices to provide image recognition services. However, traditional mobile cloud computing architecture cannot meet the demands of real time response and high accuracy since users require to upload raw images to the remote central cloud servers. The emerging architecture, Mobile Edge Computing (MEC) deploys small scale servers at the edge of the network, which can provide computing and storage resources for image recognition applications. To this end, in this paper, we aim to use the MEC architecture to provide image recognition service. Moreover, in order to guarantee the real time response and high accuracy, we also provide a feature extraction algorithm to extract discriminative features from the raw image to improve the accuracy of the image recognition applications. In doing so, the response time can be further reduced and the accuracy can be improved. The experimental results show that the combination between MEC architecture and the proposed feature extraction algorithm not only can greatly reduce the response time, but also improve the accuracy of the image recognition applications.

Keywords: feature extraction; mobile edge computing; image recognition; mobile cloud computing

1. Introduction

With the rapid growth of mobile devices, it is feasible and urgent-demand to deploy image recognition applications on mobile devices to provide image recognition services. However, since the constraint computing and storage resource, as well as energy resources, it is difficult to perform all image recognition applications on mobile devices. In recent years, a popular solution is to offload

the image recognition tasks to the remote cloud servers [1,2]. That is, the image recognition applications deployed on mobile devices, they only responsible for collecting images. Then, the mobile devices upload the images to the cloud servers to perform recognition tasks. Although this solution can save the computing and storage resources, as well as the energy resources of mobile devices. In doing so, mobile devices can provide long-lasting image recognition services for mobile users. However, in 5G networks, if hundreds of thousands of mobile users upload images to the cloud servers at the same time, the core network can be overload and even incurs network congestion, it is very likely resulting in long transmission delays. Therefore, traditional cloud computing solution cannot meet the requirements of real time response.

Mobile Edge Computing (MEC), as an emerging architecture, it is possible to solve the traditional computing solution problems that cannot meet the users real time demands [3–6]. In MEC architecture, many small-scale edge servers are deploying at the edge of the network. These edge servers can provide computing and storage resources for image recognition applications. Thus, mobile users can offload the image recognition tasks to the edge servers rather than the cloud servers. Since users close to the edge servers, generally, one hope. Therefore, the transmission delays can be reduced. In traditional cloud computing scheme, running deep neural network models on the cloud servers can achieve good performance. The reason is that cloud servers are regarded as rich computing and storage resources, as well as stored a large number of trainable images. It is well known that using a large number of trainable images to train a very deep neural networks can achieve good performance. However, in MEC architecture, a single edge server has a small range so that it only can collect a small number of images. Therefore, it is inappropriate to use deep neural network models to run the image recognition tasks.

To address this problem, in this paper, we propose a location-ware feature extraction algorithm for image recognition, called DAG_DNE_P. In DAG_DNE_P, we employ DAG_DNE_P to construct two adjacency graphs to preserve the intra-class information and the inter-class information that make every samples linked to its homogeneous and heterogeneous neighbors respectively and also, we introduce a heat kernel function as weight when construct matrix of the intra-class and inter-class. Thus, DAG_DNE_P could keep the geometric structure of the given data that find an optimal projection matrix. Experimental results validate the effectiveness of our DAG_DNE_P in comparison with DAG_DNE algorithm.

The reminder of this paper is organized as follows. Section 2 introduces the related work, which includes mobile edge computing and feature extraction algorithm. In Section 3, we introduce the proposed feature extraction algorithm. In section, we discuss the combination between MEC architecture and the proposed algorithms. The experimental results are presented in Section 4. Finally, we provide the concluding remarks in Section 5.

2. Related work

2.1. Mobile edge computing

Mobile Edge Computing (MEC), which can provide computing and storage resources for image recognition applications by deploying some small-scale servers at the edge of the network [3–6]. MEC mainly solves the problems of traditional mobile cloud computing that directly upload the raw images to the cloud servers that incurs long response time [7]. With the explosive growth of mobile

traffic, if hundreds of thousands of mobile users upload the image data to cloud servers at the same time, it may incur the core network congestion, resulting longer transmission delays and longer response time. The architecture of MEC is as shown in Figure 1, which consists of three layers of components, mobile devices, edge servers and cloud servers.

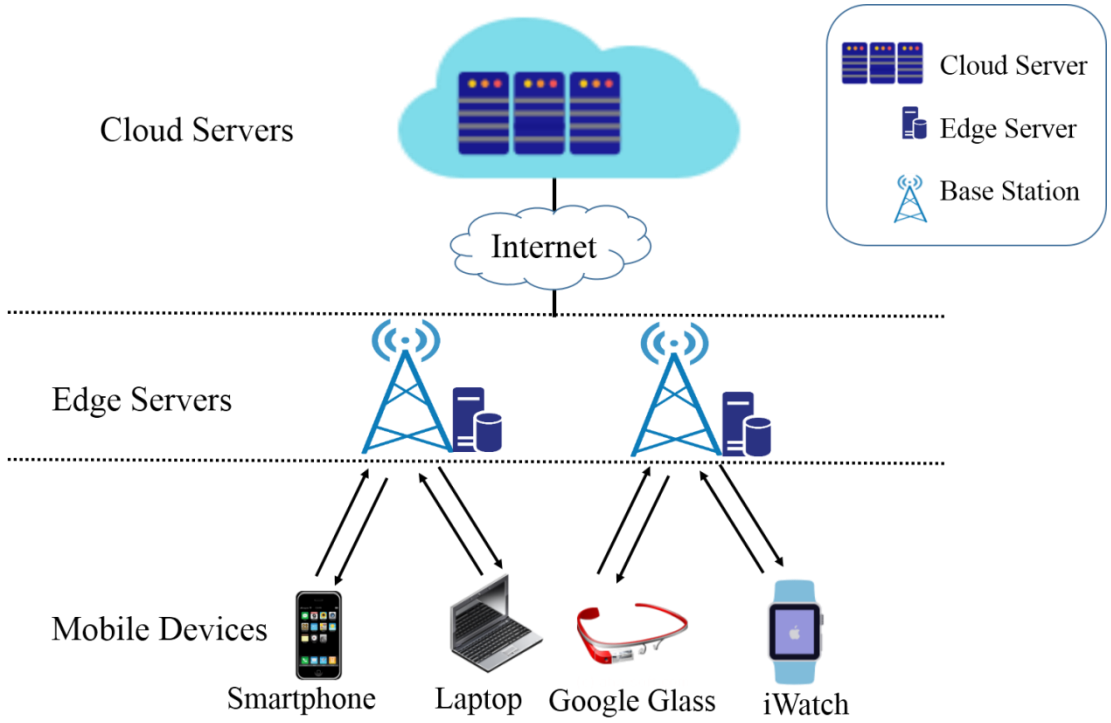


Figure 1. The architecture of MEC.

Mobile Devices: Mobile devices are the front-end devices, such as smartphone, laptop, iWatch, etc. Mobile devices are utilized to install image recognition applications. Since the constraint computing and storage resources of mobile devices, we only use them to capture images and receive results, as well as show results. *Edge Servers*: Edge servers are small scale servers deploying at the edge of the network. Edge servers are typically performing single function with limited resources, such as cache servers and specialized servers. Since edge servers close to users, using edge servers to provide computing and storage resources for image recognition applications can reduce the transmission delay. This is because mobile users only require upload the captured image to the edge servers instead of cloud servers. *Cloud Servers*: Cloud servers are regarded as having rich computing and storage resources. In general, the cloud servers are very far away from users.

2.2. .Feature extraction algorithm

Many feature extraction algorithms are proposed in recent years, such as discriminant neighborhood embedding [8], marginal Fisher analysis [9,10], local features discriminant projection [7], Appropriate points choosing based DAG_DNE [11], double adjacency graphs-based discriminant neighborhood embedding [12].

For example, Zhang et al. [8] proposed discriminant neighborhood embedding, which supposes

that multi-class data points in high-dimensional space tend to move due to local intra-class attraction or inter-class repulsion, and the optimal embedding from the point of view of classification is discovered consequently. Yan et al. [9,10] proposed marginal fisher analysis, which adopts two adjacency graphs to preserving the geometric structure. This approach first constructs two adjacency graphs, the intrinsic graph and the penalty graph, of which the intrinsic graph characterizes the intra-class compactness and connects each data point with its neighboring points of the same class, while the penalty graph connects the marginal points and characterizes the inter-class separability. Ding and Zhang [12] proposed double adjacency graphs-based discriminant neighborhood embedding, which let each sample be respectively linked to its homogeneous and heterogeneous neighbors by constructing two adjacency graphs. As a consequence, balance links are produced, neighbors belonging to the same class are compact while neighbors belonging to different classes become separable in the subspace. Thus, DAG_DNE could keep the local structure of a given data and find a good projection matrix.

However, these algorithms ignore the location information. Thus, these algorithms just preserve the local structure but ignore the location information. Nevertheless, in the task of clustering or classification, the location information is very important.

3. The location-aware feature extraction algorithm

In this section, we will introduce the location-aware feature extraction algorithm, called DAG_DNE_P. Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ be a set of training samples, where $\mathbf{x}_i \in R^d$ and $y_i \in \{1,2,...,C\}$. DAG_DNE_P aims to find a projection matrix $\mathbf{P}$, being able to extract the discriminative features from the raw image data. The extracted discriminative features have the characteristic that images with the same class label are compact, while images with different class labels are separable.

Similar to DAG_DNE algorithm, DAG_DNE_P requires to construct two adjacency graphs. Let $\mathbf{F}^w$ and $\mathbf{F}^b$ be the intra-class and inter-class adjacency matrices, respectively. For an image $\mathbf{x}_i$, $NH_k^w(\mathbf{x}_i)$ and $NH_k^b(\mathbf{x}_i)$ denote its K homogeneous set and heterogeneous set, respectively.

The intra-class adjacency matrix $\mathbf{F}^w$ and inter-class adjacency matrix $\mathbf{F}^b$ are defined as

$$F_{ij}^w = \begin{cases} \exp(\frac{-\| \mathbf{x}_i - \mathbf{x}_j \|^2}{\beta}), \ \mathrm{x}_i \in NH_k^w(\mathbf{x}_j) \ or \ \mathrm{x}_j \in NH_k^w(\mathbf{x}_i) \\ 0 \ , \ otherwise \end{cases} \tag{1}$$

$$F_{ij}^b = \begin{cases} \exp(\frac{-\| \mathbf{x}_i - \mathbf{x}_j \|^2}{\beta}),, \ \mathrm{x}_i \in NH_k^b(\mathbf{x}_j) \ or \ \mathrm{x}_j \in NH_k^b(\mathbf{x}_i) \\ 0 \ , \ otherwise \end{cases} \tag{2}$$

where $\beta \in (0,1]$, which controls the scale of weights.

The intra-class scatter $\Phi(\mathbf{P})$ and inter-class scatter $\Psi(\mathbf{P})$ are defined as follows:

$$\Phi(\mathbf{P}) = \sum_{i,j} \| \mathbf{P}^T\mathbf{x}_i - \mathbf{P}^T\mathbf{x}_j \|^2 F_{ij}^w = 2tr\{\mathbf{P}^T\mathbf{X}(\mathbf{D}^w - \mathbf{F}^w)\mathbf{X}^T\mathbf{P}\} \tag{3}$$

$$\Psi(\mathbf{P}) = \sum_{i,j} \| \mathbf{P}^T\mathbf{x}_i - \mathbf{P}^T\mathbf{x}_j \|^2 F_{ij}^b = 2tr\{\mathbf{P}^T\mathbf{X}(\mathbf{D}^b - \mathbf{F}^b)\mathbf{X}^T\mathbf{P}\} \tag{4}$$

where $\mathbf{D}^w$ and $\mathbf{D}^b$ are diagonal matrices and their entries are column sum of $\mathbf{F}^w$ and $\mathbf{F}^b$, respectively.

In order to achieve impressive accuracy, the extracted features need to satisfy that features from the same class labels are compact while features from different class labels become separable. To this end, we need to maximize the margin of total inter-class scatter and total intra-class scatter, i.e.,

$$\Theta(\mathbf{P}) = \Psi(\mathbf{P}) - \Phi(\mathbf{P}) \tag{5}$$

DAG_DNE_P aims to find a feature extractor $\mathbf{P}$ by solving the objective function (5). The complete derivation and theoretical justifications is similar to DAG_DNE, so the detail of the derivation and theoretical justification can be tracked back to [12]

$$\begin{cases} \max\limits_{\mathbf{P}} \ tr\{\mathbf{P}^T\mathbf{X}\mathbf{S}\mathbf{X}^T\mathbf{P}\} \\ s.t. \quad \mathbf{P}^T\mathbf{P} = \mathbf{I} \end{cases} \tag{6}$$

Where $\mathbf{S} = \mathbf{D}^b - \mathbf{F}^b - \mathbf{D}^w + \mathbf{F}^w$. The projection matrix $\mathbf{P}$ can be found by solving the generalized eigenvalue problem as follows:

$$\mathbf{X}\mathbf{S}\mathbf{X}^T\mathbf{P} = \lambda\mathbf{P} \tag{7}$$

Thus, $\mathbf{P}$ is composed of the optimal r projection vectors corresponding to the r largest eigenvalues.

The details of DAG_DNE_P is given in Algorithm 1.

Algorithm 1. Location-aware Feature Extraction (DAG_DNE_P)

Input: A training image set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, and the number of features r;

Output: Feature Extractor: $\mathbf{P}$;

Step 1. Compute the intra-class scatter matrix $\mathbf{F}^w$ and the inter-class scatter matrix $\mathbf{F}^b$ according to (1) and (2), respectively.

Step 2. Eigenvalue decomposition of $\mathbf{X}\mathbf{S}\mathbf{X}^T$. Let eigenvalues be $\lambda_i, i = 1,...,d$ and their corresponding eigenvectors be $\mathbf{p}_i (i = 1,...,d)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$.

Step 3. Choose the first r largest eigenvalues so that return $\mathbf{P} = [\mathbf{p}_i, \cdots, \mathbf{p}_r]$

The DAG_DNE_P algorithm is an improved version of the DAG_DNE algorithm. DAG_DNE could keep the local intrinsic structure for the raw image data through the extracted features by constructing two adjacency graphs. However, DAG_DNE just gives the weight value +1 when construct intra-class adjacency graph and inter-class adjacency graph. That just give +1 cannot

preserve the geometric structure of the given data and the geometric structure plays a different role in the classification task. Therefore, when extracting the discriminative features from the raw image data, some more important discriminative features may miss. DAG_DNE_P regulates heat kernel function as the weight that can preserve the geometric structure among data. As a result, DAG_DNE_P can achieve a good performance.

4. Discussion

Since a single edge server only collects a small amount of images, it is inappropriate to use deep neural network models to perform image recognition tasks. The reason is that using deep neural network models cannot achieve good performance due to the small number of images stored on a single edge server. In the contrast, the proposed feature extraction algorithm can well fit the small number of images and achieve good performance. In this sense, Combining the MEC architecture [13–17] and the proposed feature extraction algorithm to provide image recognition services is advisable.

5. Experiments and analysis

We have conducted experiments on two datasets, which are publicly available, UMIST and ORL datasets. Wherein the UMIST dataset contains 564 face images of 20 individuals and the ORL dataset contains 400 face images of 40 individuals, with 10 images for each individual.

DAG_DNE and the proposed algorithm are implemented in MATLAB 2015b, and are conducted on an i5 Intel ® Core CPU 2.50 GHz machine with 4G bytes of memory. In our experiment, which requires the nearest neighbor parameter K for constructing adjacency graphs. For simplicity, the nearest neighbor classifier is used for classifying the test images

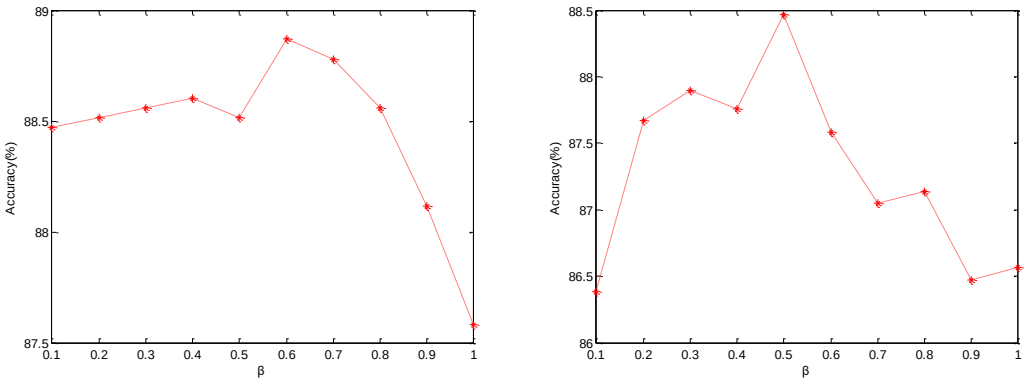
To evaluate the effectiveness and the correctness of DAG_DNE_P, experiments are carried out on UMIST and ORL databases, and the results are compared with DAG_DNE.

In the experiment, the parameter β is selected to be several different sets of values so as to observe their effect on the recognition rate. The whole validation set, and the value of parameter β is selected based on the training result on the validation set.

5.1. Results with UMIST dataset

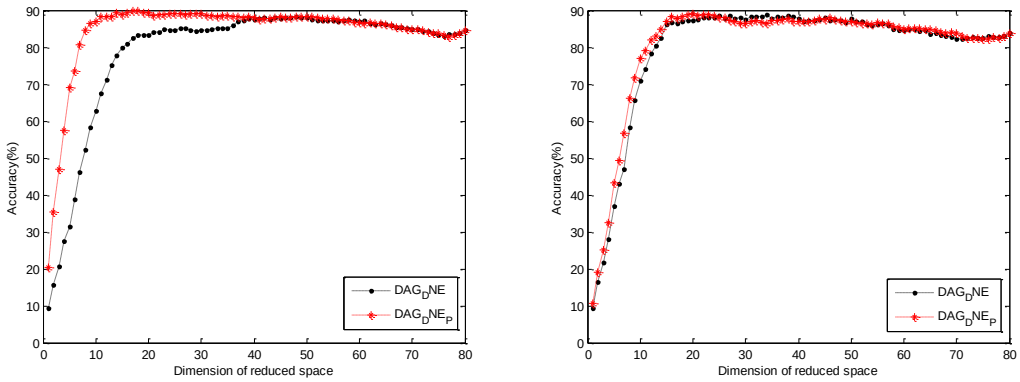
The UMIST dataset consists of 564 images of 20 individuals, taking into account race, sex and appearance. Each subject is taken in a range of poses from profile to frontal views. The pre-cropped dataset is used and the size of each image is 112×92 pixels, with 256 gray levels per pixel. For UMIST datasets, we randomly select 20% images from the database as training samples, the remaining 80% as test samples. Figure 2 shows some image samples in the UMIST dataset. We repeat 20 runs and report the average results.

Figure 2. Face images from the UMIST database.



(a) $K=1$ (b) $K=3$

Figure 3. Average recognition rates vs. β .



(a) $K=1$ (b) $K=3$

Figure 4. Recognition accuracy for different parameters on UMIST database.

First, we consider parameter choose, the nearest neighbor parameter K is selected in the set $\{1,3\}$. Figure 3 illustrates the relationships of accuracy and the value of β. From Figure 3 we know that different values of β has different accuracy, which β as a tuning parameter that preserve the geometric structure of the given data. So we know that the geometric structure information plays a great role in the classification task.

Figure 4 (a) and (b) show the accuracy of two algorithms vs. dimensionality of subspace with different K. From Figure 4 (a) and (b), the classification accuracy of DAG_DNE_P and DAG_DNE algorithms are all increase rapidly, however, DAG_DNE_P is more rapidly than DAG_DNE.

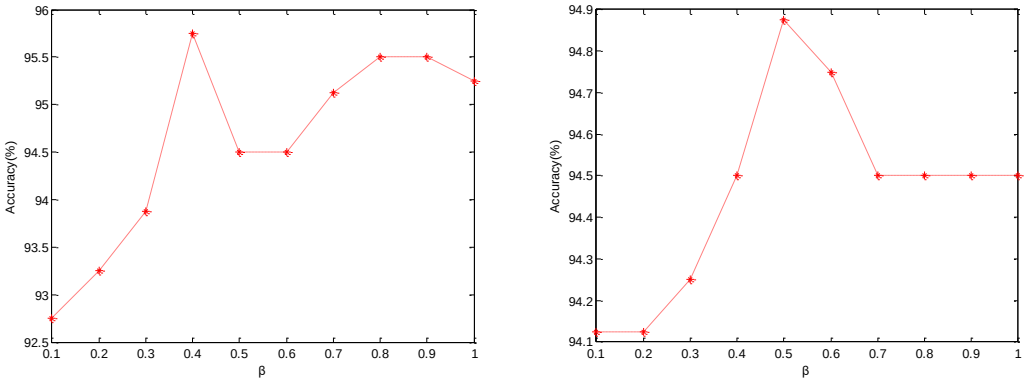
5.2. Results with ORL dataset

The ORL dataset is composed of 40 distinct subjects, each of which contains 10 different gray-scale images. In ORL dataset, images for each subject were taken by varying the lighting, facial expressions or facial details at different times, and all were taken against a dark homogenous background in an upright and frontal position. The size of each image is 112×92 pixels, with 256 gray levels per pixel. For ORL dataset, we randomly select 60% images from the database as training samples, the remaining 40% as test samples. Figure 5 shows some image samples in the ORL dataset. There are 240 training samples and 160 test ones. Figure 6 illustrate the relationship β and recognition rate with $K=1$ and $K=3$, respectively.

Figure 5. Sample face images from the ORL database.

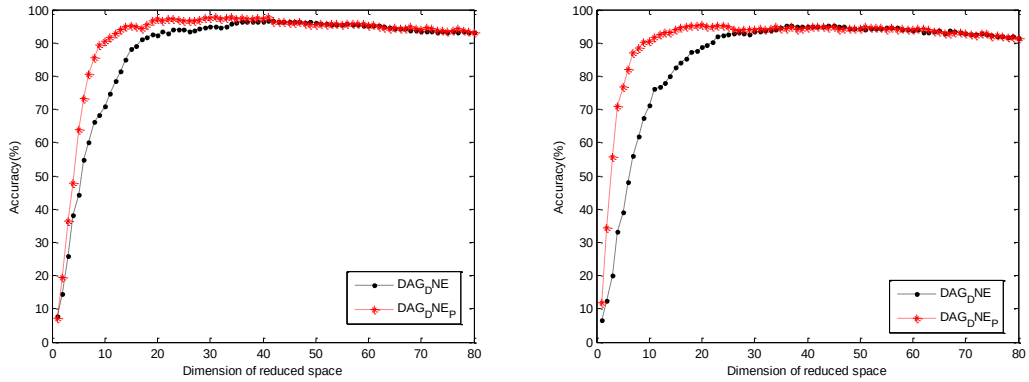
In this experiment, we reduce dimensionality with two times so as to get a high running speed. When choose parameter, similar to UMIST dataset, the neighborhood parameter K in DAG_DNE and DAG_DNE_P is set to be 1 and 3, respectively.

Figure 7 shows the accuracy of DAG_DNE and DAG_DNE_P algorithms vs. dimensionality of subspace with different K. We can see that DAG_DNE_P can obtain good performance and more rapidly than DAG_DNE to get the best performance. Compared with DAG_DNE, DAG_DNE_P has a better recognition rate and its optimal discriminative subspace has a relatively low dimensionality so as to reduce the complexity of calculation.



(a) $K=1$ (b) $K=3$

Figure 6. Average recognition rates vs. β.



(a) $K=1$ (b) $K=3$

Figure 7. Recognition accuracy for different parameters on ORL database.

6. Conclusion

In this paper, we proposed a location-aware feature extraction algorithm to fit the image recognition in the MEC environment. By considering that in MEC architecture, a single edge server only collects a small number of images, we propose a location-aware feature extraction algorithm, which can achieve good performance when the trainable images are very few. Moreover, the proposed feature extraction algorithm considers the location information of the images, compared with traditional feature extraction algorithms, it achieves higher accuracy. Thus, by combining the MEC architecture and the proposed feature extraction algorithm, it is possible to support the mobile devices to provide long-lasting real time response and high accuracy image recognition services. Finally, on two publicly datasets, we demonstrate the effectiveness of our proposed feature extraction algorithm.

Acknowledgment

The work is supported by National Natural Science Foundation of China (No.61003100), Hubei Provincial Natural Science Foundation of China (No. 2017CFA012), National Natural Science Foundation of China (No. 61572012), National Natural Science Foundation of Hubei Province (No.2015CFB525) and Wuhan science and technology plan innovation team project (No.201307020402005).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. S. Bera, S. Misra and J. J. P. C. Rodrigues, Cloud computing applications for smart grid: A survey, *IEEE T. Parall. Distr.*, **26** (2015), 1477–1494.
2. C. Esposito, A. Castiglione, B. Martini, et al., Cloud manufacturing: security, privacy, and forensic concerns, *IEEE Cloud Comput.*, **3** (2016), 16–22.
3. Y. C. Hu, M. Patel, D. Sabella, et al., Mobile edge computing: A key technology towards 5G, *ETSI white paper*, **11** (2015), 1–16.
4. M. T. Liu, F. R. Yu, Y. L. Teng, et al., Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing, *IEEE T. Wirel. Commun.*, **5** (2019), 695–708.
5. S. Wang, Y. Zhao, J. Xu, et al., Edge Server Placement in Mobile Edge Computing, Journal of Parallel and Distributed Computing, 2018. Available from: https://www.sciencedirect.com/science/article/pii/S0743731518304398.
6. S. Wang, Y. Zhao, L. Huang, et al., QoS Prediction for Service Recommendations in Mobile Edge Computing, Journal of Parallel and Distributed Computing, 2017. Available from: http://www.sciencedirect.com/science/article/pii/S074373151730268X.
7. H. T. Zhao, S. Y. Sun, Z. L. Jing, et al., Local structure based supervised feature extraction, *Pattern Recognition*, **39** (2005), 1546–1550.
8. W. Zhang, X. Y. Xue, H. Lu, et al., Discriminant neighborhood embedding for classification, *Pattern Recognition*, **39** (2006), 2240–2243.
9. S. C. Yan, D. Xu, B. Y. Zhang, et al., Graph embedding: a general framework for dimensionality reduction, In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2005), 830–837.
10. S. C. Yan, D. Xu, B. Y. Zhang, et al., Graph embedding and extensions: a general framework for dimensionality reduction, *IEEE T. Pattern Anal.*, **29** (2007), 40–51.
11. C. T. Ding and S. G. Wang, Appropriate points choosing for subspace learning over image classification, *J. Supercomput.*, **75** (2018), 688–703.
12. C. T. Ding and L. Zhang, Double adjacency graphs-based discriminant neighborhood embedding, *Pattern Recognition*, **48** (2015), 1734–1742.
13. Y. C. Hu, M. Patel, D. Sabella, et al., Mobile edge computing: A key technology towards 5G, *ETSI white paper*, **11** (2015), 1–16.

14. N. Abbas, Y. Zhang, A. Taherkordi, et al., Mobile edge computing: a survey, *IEEE Internet Things*, **5** (2018), 450–465.
15. E. Ahmed, A. Naveed, A. Gani, et al., Process state synchronization-based application execution management for mobile edge/cloud computing, *Future Gener. Comp. Sy.*, **91** (2019), 579–589.
16. Y. M. Zhang, X. L. Lan, Y. Li, et al., Efficient Computation Resource Management in Mobile Edge-Cloud Computing, *IEEE Internet Things*, **6** (2019), 3455–3466.
17. J. Zhang, L. Zhou, Q. Tang, et al., Stochastic Computation Offloading and Trajectory Scheduling for UAV-Assisted Mobile Edge Computing, *IEEE Internet Things*, **6** (2019), 3688–3699.

AIMS Press