Multitask-Guided Deep Clustering With Boundary Adaptation

Xiaobo Zhang, *Member, IEEE*, Tao Wang, Xiaole Zhao, *Member, IEEE*, Dengmin Wen, and Donghai Zhai

***Abstract*— Multitask learning uses external knowledge to improve internal clustering and single-task learning. Existing multitask learning algorithms mostly use shallow-level correlation to aid judgment, and the boundary factors on high-dimensional datasets often lead algorithms to poor performance. The initial parameters of these algorithms cause the border samples to fall into a local optimal solution. In this study, a multitask-guided deep clustering (DC) with boundary adaptation (MTDC-BA) based on a convolutional neural network autoencoder (CNN-AE) is proposed. In the first stage, dubbed multitask pretraining (M-train), we construct an autoencoder (AE) named CNN-AE using the DenseNet-like structure, which performs deep feature extraction and stores captured multitask knowledge into model parameters. In the second phase, the parameters of the M-train are shared for CNN-AE, and clustering results are obtained by deep features, which is termed as single-task fitting (S-fit). To eliminate the boundary effect, we use data augmentation and improved self-paced learning to construct the boundary adaptation. We integrate boundary adaptors into the M-train and S-fit stages appropriately. The interpretability of MTDC-BA is accomplished by data transformation. The model relies on the principle that features become important as the reconfiguration loss decreases. Experiments on a series of typical datasets confirm the performance of the proposed MTDC-BA. Compared with other traditional clustering methods, including single-task DC algorithms and the latest multitask clustering algorithms, our MTDC-BA achieves better clustering performance with higher computational efficiency. Deep features clustering results demonstrate the stability of MTDC-BA by visualization and convergence verification. Through the visualization experiment, we explain and analyze the whole model data input and the middle characteristic layer. Further understanding of the principle of MTDC-BA. Through additional experiments, we know that the proposed MTDC-BA is efficient in the use of multitask knowledge. Finally, we carry out sensitivity experiments on the hyper-parameters to verify their optimal performance.**

Manuscript received 31 August 2022; revised 20 February 2023 and 6 June 2023; accepted 13 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61976247 and Grant 62102330, in part by the Key Research and Development Project in Sichuan Province of China under Grant 2023YFS0404, in part by the Natural Science Foundation of Sichuan Province under Grant 2022NSFSC0947, in part by the Fundamental Research Funds for the Central Universities of China under Grant 2682022KJ045, and in part by the Open Research Fund Program of Data Recovery Key Laboratory of Sichuan Province under Grant DRN2203. *(Corresponding author: Xiaole Zhao.)*

The authors are with the School of Computing and Artificial Intelligence and the National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Southwest Jiaotong University, Chengdu 611756, China, and also with the Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, Chengdu 611756, China (e-mail: zhangxb@swjtu.edu.cn; wangt@my.swjtu.edu.cn; zxlation@foxmail.com; dmwen@swjtu.edu.cn; dhzhai@swjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3307126.

Digital Object Identifier 10.1109/TNNLS.2023.3307126

***Index Terms*— Boundary adaptation, clustering, deep learning, explainable method, multitask learning.**

NOMENCLATURE

m	Number of multitask datasets.
$X = \{X^1, X^2, \ldots, X^m\}$	Datasets of multitask.
$X^j = \{x_i^j\} \in \mathbb{R}^{d \times N^j}$	jth dataset contains N^j samples.
N^j	Number of samples contained in the jth dataset.
Y	Multitask dataset used for the M-train.
$Z^j = \{z_i^j\} \in \mathbb{R}^{r \times N^j}$	Deep feature set extracted from the jth dataset.
r	Dimension of the deep features.
d	Dimension of the single data sample.
$S^j = \{s_i^j\} \in \mathbb{R}^{N^j}$	Cluster assignment for the jth dataset.
k	Number of clustering.
$C = \{c_1, c_2, \ldots, c_k\}$	Clustering centers of all clusters.
$h_u(\cdot)$	Decoder mapping.
$f_w(\cdot)$	Encoder mapping.
L_{c}^M and L_{r}^M	Clustering loss and net loss of M-train.
L_{c}^S and L_{r}^S	Clustering loss and net loss of S-fit.
L^S	Final loss of S-fit.
$\alpha = \gamma\ (\cdot)$	Data enhancement extension mapping.

I. INTRODUCTION

CLUTERING is an active topic in the field of machine learning, which has been widely used in pattern recognition, computer vision, and data mining. Traditional clustering methods, such as K-means (KM) [1], Gaussian mixture model [2], and hierarchical clustering [3], usually group samples by the similarity implicitly contained in the data. Other methods, e.g., spectral clustering [4] and kernel KM [5]. In essence, these methods conduct shallow learning via extracting shallow features from samples, which fails to fully explore the deep correlation between different samples and tasks.

In recent years, clustering methods based on deep-learning techniques [6] have made remarkable progress due to the

powerful representational capacity of deep neural networks (DNNs) for large-scale data. DNNs are used to transform the data into more clustering-friendly representations with the guidance of clustering objectives. Therefore, deep-learning models encode the original data in deep feature spaces. Moreover, DNNs can also play a crucial role in dimension reduction and alleviate the problem of the "dimension curse." Both performance and efficiency of clustering methods have been greatly facilitated in the context of deep clustering (DC).

However, both traditional methods and DC methods are rarely considered to promote clustering on the target task. In general, the same data keeps implicit heterogeneity for different tasks, while different data also maintain implicit homogeneity for the same tasks. In multitask learning [7], the common knowledge shared by multiple tasks is extracted and transferred in a general manner.

Most methods do not consider the influence of boundary samples on class centers. In unsupervised learning, the initial parameters of DNNs often determine the final result of boundary samples, which makes it impossible to effectively learn boundary samples. Actually, ignoring boundary effects also means excluding data enhancement, which will lead to weak generalization of the model. This indicates that the robust features for clustering cannot be learned. We put forward a way of thinking to solve this problem and incorporate it into our proposed approach.

In this article, to explore the interpretability of the DC method, we propose a multitask-guided DC with boundary adaptation (MTDC-BA) algorithm, which consists of two stages: multitask pretraining (M-train) and single-task fitting (S-fit). In the M-train phase, a convolutional neural network autoencoder (CNN-AE) was built that uses DenseNet as the encoder and five deconvolution layers as the decoder. We use CNN-AE to extract fuzzy knowledge from all task data. In the subsequent S-fit phase, the extracted fuzzy knowledge will help us improve the single-task clustering results. It is worth mentioning that in the overall framework, we use self-paced and data argumentation techniques to construct the boundary adapters to eliminate the negative effects of boundary effects and improve the clustering efficiency. Extensive experiments on several representative datasets illustrate the advantages of our method over other state-of-the-art works. The main contributions of this article are highlighted below.

1) Multitask learning is integrated into DC, which makes them benefit each other to some extent. The powerful representation capacity of deep features helps the model adequately excavate the deep association between multiple tasks. On the other hand, deep features of the data can be effectively acquired with the aid of the shared knowledge among multiple tasks. The performance of the target clustering can be significantly improved by the shared knowledge extracted from multiple tasks by our CNN-AE.
2) A DenseNet-like CNN is built to extract deep features of samples. Improved by data augmentation and boundary adaptation, the clustering effect of depth features is more obvious. The visualization experiment of the feature layer in the model verifies the validity of the method in the interpretability of the method.
3) For intractable ambiguous samples, we propose an improved strategy of boundary adaptation based on data augmentation and self-paced learning. The strategy is designed to eliminate the uncertainty of ambiguous samples and promote the clustering performance of the proposed method on the boundaries of deep features. The working principle of boundary adaptation is explained visually by the feature layer.
4) Extensive experiments are implemented to verify the promotion of multitask learning on the target clustering and the effectiveness of boundary adaption. Also, comparative experiments are presented to illustrate the superiority of the proposed method over other state-of-the-art clustering methods.

The rest of this article is organized as follows. Section II presents a review of related work. Section III introduces the proposed MTDC-BA algorithm. In Section IV, we conducted extensive experiments on the constructed learning model. It includes the interpretation of the meaning of the network framework hierarchy. In Section V, we present several variants by changing the structure of the network. Section V presents the conclusion and prospects for future work.

II. Related Work

The proposed framework for multitask clustering integrates deep feature extraction, self-paced learning, and data augmentation. Therefore, in this section, we only review and discuss the existing work related to this article.

A. Multitask Clustering

Multitask clustering is a type of unsupervised multitask learning, which improves the clustering performance of a single task by considering the correlation between tasks. Three knowledge transfer methods are usually used to reuse knowledge, which are feature transfer, instance transfer, and model parameter transfer.

Feature transfer looks for an expression method that can represent the features of multiple tasks, but will reduce the distribution difference of the tasks. At present, more popular learning methods are finding a new subspace for tasks with the same centroid, and the subspace contains all the data of all tasks. For example, learning the shared subspace for multitask clustering (LSSMTC) [8] finds a new kernel space for all tasks with similar distributions. The transfer learning-based maximum entropy clustering (TL_MEC) [9] learns a feature association matrix for each task such as multitask information bottleneck co-clustering (MIBC) [10].

Instance transfer improves the clustering process of the main task by transferring the instance samples of the auxiliary task. Multitask information bottleneck (MTIB) [11] quantifies the correlation of different task distributions by designing a minimum loss function, multihop balanced clustering (MBC) [12] uses the exchange of cluster centroids. Learning the correlation between tasks, self-adapted multitask clustering (SAMTC) [13] constructs a common subtask for multiple tasks with shared cluster labels, and learns the correlation through

the correlation of subtasks knowledge. manifold regularized coding multitask clustering (MRCMTC) [13] is to alternately learn a low-dimensional space, it satisfies the data in the space and completely replaces the data in the source task.

The parameter transfer is also often used in deep-learning models. Multitask embedding learning (MTEL) [14] constructed an adjacency matrix of two task data, and achieved the purpose of clustering the target node by predicting the interval of each pair of nodes in the adjacency matrix. The regularization model of two tasks in the algorithm shares parameters. The advantage of the parameter is that it performs well in attribute graphs, but the disadvantage is also obvious, and it can only be used for knowledge transfer learning between two tasks. We solved this issue with an end-to-end approach.

B. Deep Clustering

DC is a clustering algorithm that adopts a DNN to learn cluster-oriented features. Depending on the type of network, we are able to classify DC algorithms into four types: autoencoder (AE)-based, variational AE (VAE)-based, generative adversarial network (GAN)-based, and convolutional neural networks (CNNs)-based [15]. AE-based models are currently the most popular, with associated structures used and showing good performance in a large number of experiments [16], [17], [18], [19], [20]. In the past five years, Guo et al. [21] proposed to ignore the network loss and only use the clustering loss to replace the loss of the entire model. Not only that, the study also incorporated self-expression features into the middle layer of the fully connected AE.

The advantage of VAE-based can predefine a distribution that conforms to the cluster structure. Shahin et al. [22] proposed the predefined maximum and lower bound distribution, and a Gaussian mixture model with higher performance is obtained. Shu et al. [23] performed prefitting on the density, and its experiment results are more in line with the distribution curve of the real clustering results. However, they all have a major defect, the complexity is too high, especially in image data, and the complexity shows a geometric growth trend.

GAN- and VAE-based methods are much the same. Wang et al. [24] learned the common low-dimensional features, which can make up the missing view data and capture a better structure for clustering. Liu et al. [25] beside two typical parts in GAN, i.e., generator (G-Net) and discriminator (D-Net). There is a third party named TaskNet (T-Net) in the task-oriented GAN. However, GAN-based also has the disadvantage of hard convergence and model collapse, just like GANs [26].

The last one is called CNN-based approaches, which only appeared in recent years and belongs to the latest development research of DC. Reference [27] is an early CNN-based research, a novel machine-learning model combining CNN with KM clustering is proposed. In [28], CNN-long–short-term memory (CNN-LSTM) is constructed. What's more, Yu et al. [29] showed that CNN-based has better data processing and feature extraction ability on large-scale cluster data. We improve the CNN-based by introducing the AE-based model, which retains the excellent features of the CNN-based and shows strong clustering robust feature learning ability.

C. Self-Paced Learning and Data Augmentation

Many existing studies have ignored the influence of boundary factors, Bai et al. [30] showed that samples far away from the cluster center have a negative effect on the training of the whole DNN, we solved this problem with self-paced learning and data enhancement.

Self-paced learning simulates the law of human learning and follows the basic logic from simple to difficult. Zheng et al. [31] confirmed that adding self-paced learning to unsupervised learning can enhance the generalization of the overall model. Yu et al. [32] applied self-paced learning to KM clustering. Compared with the scheme without self-paced learning, the excellent results confirm the feasibility of self-paced clustering learning.

Data augmentation is an effective way to expand samples. We can expand data by rotation, offset, and clipping to prevent model overfitting. The importance of data augmentation is clearly demonstrated and verified in [33], [34], and [35]. However, data argumentation is rarely used because of its instability.

MTDC-BA combines self-paced learning and data argumentation to eliminate the negative influence of boundary factors and enhance the stability of the whole model. The experiment results on various datasets verify the validity of our solution.

III. Multitask-Guided DC With Boundary Adaptation

In this section, we introduce the proposed MTDC-BA, which explores the relevance of multiple tasks and cluster assignments for each task. In Section III-A, we present the overall framework of MTDC-BA, a two-phase popular structure of M-train and S-fit. In Section III-B, we describe the detailed processes and principles of the two phases, highlighting their workflow, and the relationship between the two phases. In Section III-C, after a two-stage process is established, we improve learning efficiency by eliminating the negative effects of boundary factors through data enhancement and self-paced learning. Section III-D presents a summary of our algorithm and tries to analyze interpretability, including the analysis of time complexity and real-time analysis.

A. Network Architecture

In this section, we introduce the network architecture of the proposed MTDC-BA. We need to summarize the definitions and notations. The most frequently used symbols in this article are shown in Nomenclature.

Definition 1 (MTDC-BA): Suppose there are m tasks for clustering, each of which corresponds to a dataset X^j, $j = 1, 2, \ldots, m$. These datasets contain common samples $Y = \{x_i\}_{i=1}^{n}$ for different tasks. $X^j = \{x_i^j\} \in \mathbb{R}^{d \times N^j}$ is the jth dataset, where x_i^j is the ith sample in the jth dataset, and N^j is the amount of data contained in the dataset, and d is the dimension of a single data sample. MTDC-BA is designed to

take advantage of the powerful abstractions of deep learning to extract important features that can represent metadata $Z^j = \{z_i^j\} \in \mathbb{R}^{r\times N^j}$, where z_i^j is deep features of x_i^j, and r is the dimension of z_i^j. MTDC-BA utilizes the deep features to improve the final cluster allocation $S^j = \{s_i^j\} \in \mathbb{R}^{N^j}, s_i^j \in \{1, 2, \ldots, n\}$, where n is the number of cluster classes given in advance.

As shown in Fig. 1, MTDC-BA consists of two parts: the M-train and the S-fit. The M-train stage is responsible for capturing and maintaining the shared knowledge and interrelationship between these tasks. The S-fit phase is built to draw lessons from the shared knowledge and improve the clustering performance of the target task.

M-train and S-fit adopt the parameter transfers which is one of the multitask knowledge transfer modes to learn the whole multitask. Both M-train and S-fit depend on a CNN-AE. CNN-AE inherits DenseNet's properties due to the construction of dense connections, which greatly improves the ability of feature extraction and knowledge capture. The decoder uses five levels of deconvolution and two levels of full connection, like [36], and the decoder has excellent learning capabilities. In addition, the boundary adaptation produces the effect in M-train and S-fit and eliminates the boundary influence. Specifically, the boundary adaptation removes the boundary samples during the iteration of M-train and S-fit so that they are not involved in knowledge capture. The samples near the cluster center have a higher influence. Then, the credibility has been greatly enhanced. Next, to explore the learning process in more detail, we begin by showing how multitask knowledge is captured and used by M-train and S-fit.

B. Two-Phase Learning for Multitask Relationship

In the scenario of multitask clustering, the common method to capture multitask knowledge is to find the same distribution of different tasks. Specifically, the different tasks T_i and T_j or their subsets are associated. In detail, different subsets have the same class label, and these subsets follow a similar density distribution. We need to find these subsets before moving into the M-train phase and leverage them to improve performance.

In order to facilitate the search, we first transform the data of all tasks into a uniform representation, so that the data has the same feature dimension. We then designed a D-Net that distinguishes data samples with multiple tags. The D-Net works as follows:

$$Y = \cup_{k=1}^{T} \cap_{j=1}^{m} g(X^j, t_k) \tag{1}$$

where t represents the total number of all datasets and all labels, t_k represents the kth label, $g(X^j, t_k)$ means to look for a sample with the t_k tag in the dataset X^j, and $\cup_{k=1}^{T} \cap_{j=1}^{m} g(X^j, t_k)$ represents the intersection of samples containing t_k tags in m datasets.

Then, we use the resulting dataset Y correlated to multiple tasks for shared knowledge learning. During the M-train stage, we will get low-dimensional features through CNN-AE, which is trained by minimizing the L_r^M loss

$$L_r^M = \frac{1}{n}\sum_{i=1}^{n} ||x_i - h_u(f_w(x_i))||^2 \tag{2}$$

where L_r^M represents the reconstruction loss (M stands for M-train and r indicates the reconstruction loss), and x_i is the ith sample in Y. $f_w(*)$ corresponds to the function of the encoder and $h_u(*)$ is the decoder, where u and w is the network parameters for these two structures. n is the total number of samples in Y. The target is to obtain the intermediate codes of the CNN-AE

$$Z = \{f_w(x_i)\}_{i=1}^n = \{z_i\}_{i=1}^n \in \mathbb{R}^{r\times n} \tag{3}$$

where Z is the result of dimension reduction of CNN-AE and r is the dimension of z_i. During the M-train phase, CNN-AE is trained by minimizing the following objectives:

$$\min \frac{1}{n}\sum_{i=1}^{n} ||h_u(z_i) - x_i||^2. \tag{4}$$

The goal of the M-train stage is obtaining suitable parameters u and w for the S-fit phage. Therefore, it is critical to judge when the training of CNN-AE should be ended. We performed a cursory review of the intermediate code z_i, depending in part on our clustering loss L_c^M

$$L_c^M = \frac{1}{n}\sum_{i=1}^{n} ||x_i - c_i||^2 \tag{5}$$

where c_i is the cluster center corresponding to the sample x_i. Next, the clustering assignment S represents the cluster center to which the ith sample belongs. The $S = \{s_i\} \in \mathbb{R}^n$ can obtained by clustering Z

$$C = \pi(Z) \tag{6}$$

where $\pi(Z)$ means to cluster Z using KM or other basic clustering methods such as spectral clustering or hierarchical clustering. We set up L_c^M as the loss to help determine when the M-train phase has been completed. We take the derivative of L_c^M with respect to the code z_i, i.e., $\partial L_c^M/\partial z_i$, as the criteria. When $\partial L_c^M/\partial z_i < 0$, then the optimization of L_c^M starts to change the direction of searching. If the M-train phase continues, the network parameters u and w might change to be unsuitable for knowledge sharing of multitask clustering, then the M-train phase should be terminated.

Our MTDC-BA is a streamlined structure, and the end of the M-train is the beginning of the S-fit. The multitask knowledge captured in the M-train phase is stored in the model parameters u and w. We use u and w to initialize CNN-AE in the S-fit phase. The initial activation process is shown in Fig. 2.

The S-fit phase is designed for single-task clustering. During this stage, the overall loss can be written as

$$L^S = L_c^S + L_r^S \tag{7}$$

where the L_r^S represents the reconstruction loss resulting from the S-fit phase for single-task data fitting, the L_c^S represents the loss to single-task data clustering during the S-fit phase, and the principle of L_r^S and L_c^S is the same as (2) and (5).

We use the L_c^S in the (7) as the overall loss during the S-fit phase, which is the gain

$$L^S = \frac{1}{n_j}\sum_{i}^{n_j} \left\|x_i^j - s_i^j\right\|^2 \tag{8}$$

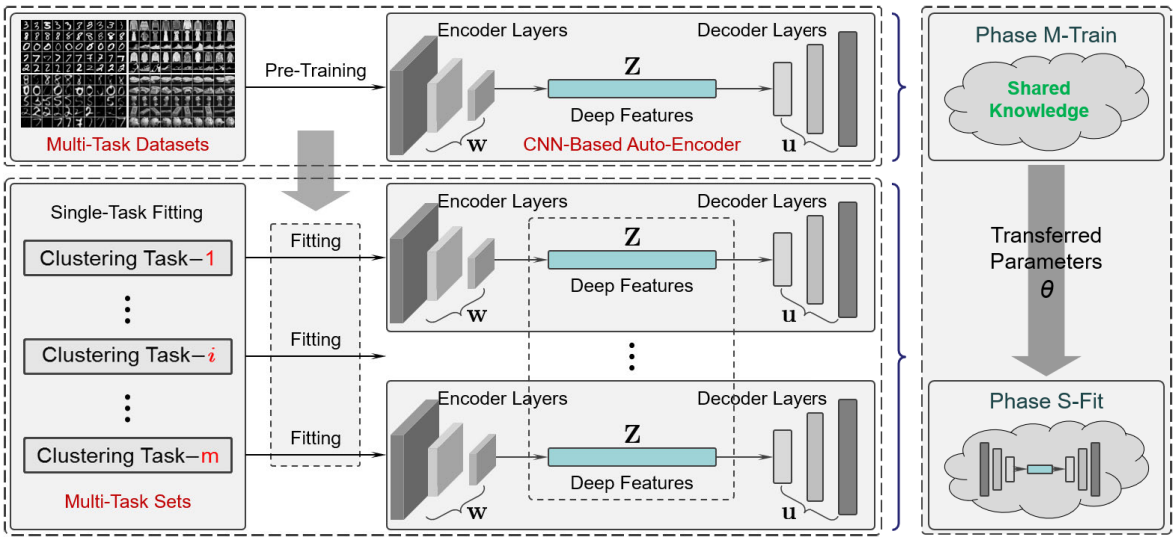


Fig. 1. Overall architecture of MDTC-BA. The multitask dataset completes the initial M-train phase through CNN-AE, which inherits the CNN-AE model parameters from the M-train in the S-fit and gets the results from clustering the generated deep features. It is worth noting that the two steps use different datasets, the first of which contains multitasking data. Multitask knowledge is extracted into the parameter θ. In S-fit, the target single task uses the extracted knowledge to obtain high-performance results.

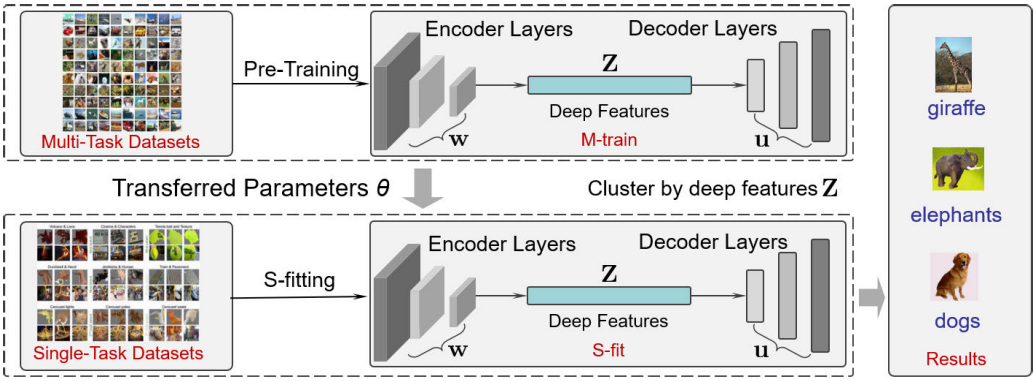


Fig. 2. S-fit initialized with the M-train parameters, and the single-task data obtained through S-fit to calculate deep features, then the cluster division is obtained. Note that the parameters obtained by training in the M-train phase are inherited in the S-fit phase, and the input becomes the target task data in the S-fit phase.

where x_i^j is the ith sample of the jth task, s_i^j is the cluster center of the cluster partition about x_i^j, and n_j is the total number of samples in jth task. The s_i^j is obtained by clustering the low-dimensional feature set $Z^j = \{z_1^j, z_2^j, \ldots, z_{n_j}^j\}$ after each iteration.

During the S-fit phase, the entire clustering process can be understood as Fig. 2. We put the single-task dataset that currently needs to complete clustering into the CNN-AE first. Depending on the resulting middle deep feature Z, we performed the entire clustering, and the training of CNN-AE depends on the value of the loss that we define. When L^S tends to be stable, we can determine that the distribution of clusters tends to be completed. The CNN-AE training process for the entire S-fit phase can be expressed as

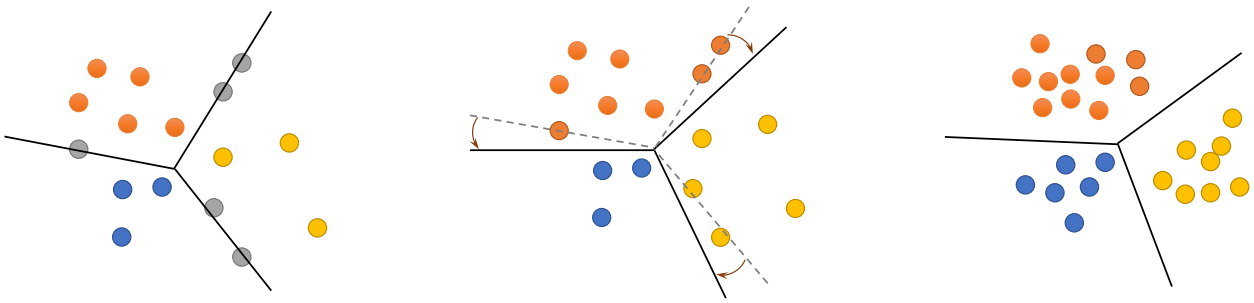
$$\min_{u,w} \frac{1}{n} \sum_{i=1}^{n} \left\| h_u\left(f_w\left(x_i^j\right)\right) - x_i^j \right\|^2$$
$$\text{s.t. } L^S = \frac{1}{n_j} \sum_{i}^{n_j} \left\| x_i^j - s_i^j \right\|^2. \tag{9}$$

However, the whole model does not eliminate the influence of the boundary factor, so it often falls into the local optimal trap during the training process.

C. Boundary Adaption

The boundary samples and the initial boundary parameters of the model often lead to the failure of the whole model. In MTDC-BA, the boundary factor is eliminated by the boundary adaption. The boundary adaption is conducted by self-paced learning and data augmentation, which is shown in Fig. 3. Because the data augmentation may bring the problem of data imbalance, we need to delimit the offset range when we carry on the data augmentation. The data imbalance problem is solved when we perform only nonoverlapping angular rotation. But at this time the data expansion is not sufficient, we rely on self-paced learning to further mine knowledge.

After an iteration loop, CNN-AE will incorporate all the samples into the training, including the boundary factor, which obviously can not be avoided. We use the method of discarding the boundary samples so that the samples learned after each



(a) (b) (c)

Fig. 3. (a) Boundary adaptation improves the clustering of ambiguous samples from two aspects. (b) Self-paced learning fine-tunes ambiguous samples so that they move away from cluster boundaries, which is equivalent to optimizing the boundaries in a fine-tuning manner. (c) Data augmentation smoothes the manifold where the training samples reside, making the samples easier to aggregate in the high-dimensional space.

iteration are reliable samples near the center of the class. In order from simple to difficult, we learn the samples near the center of the cluster first, then learn the samples far from the center, which means the boundary samples. In general, giving the training sample set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and the learning model f with parameter w, the goal of traditional self-paced learning is

$$\min_{w} \sum_{i=1}^{n} v_i L(f_w(x_i), y_i) + g(\lambda, v_i) \quad \text{s.t. } v_i \in [0, 1] \tag{10}$$

where $v = [v_1, v_2, \ldots, v_n]^T$ is the weight of the sample and $g(\lambda, v_i)$ is called the self-paced regularization term. λ is the valve to determine the degree of learning difficulty.

We use (10) instead of (4) to obtain a loss function in conjunction with self-paced learning

$$L_r = \frac{1}{n} \sum_{i=1}^{n} v_i ||f_w(\mathrm{x}_i) - y_i||^2 - \lambda v_i \quad \text{s.t. } v_i \in [0, 1]. \tag{11}$$

Traditional self-paced learning consists of two hyper-parameters: λ controls the learning speed and another one controls the step size, but it is difficult to choose the appropriate step size because our loss gradually decreases during training. Based on the above analysis, we optimize the steps of self-paced learning and propose a new learning speed λ

$$\lambda = \mu(L^t) + \frac{t}{T}\sigma(L^t) \tag{12}$$

where L^t is the total loss during training, T is the maximum number of iterations trained, t is the number of current iterations, and $\mu(*)$ and $\sigma(*)$ are the mean and variance of the loss. At this point according to (12), the loss value L^t is deterministic during the training because of the deterministic model, while T and t are deterministic according to the model, λ becomes a value independent of any hyper-parameters. v_i can be obtained in closed form

$$v_i = \begin{cases} 1, & \text{if } ||x_i - c_i||^2 < \lambda \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Taking the defined self-paced learning rate into the overall model, we can infer the goals of the M-train phase and the S-fit phase

$$\min_{w,u} \frac{1}{n} \sum_{i=1}^{n} v_i ||h_u(z_i) - x_i|| - \lambda v_i \quad \text{s.t. } v_i \in [0, 1] \tag{14}$$

and

$$\min_{L^S} \frac{1}{n} \sum_{i=1}^{n} v_i \left\| h_u\left(f_w\left(x_i^j\right)\right) - x_i^j \right\|^2 - \lambda v_i \tag{15}$$

where come from (12).

In order to improve the learning efficiency of boundary samples, we use data augmentation and define a transformation equation

$$\alpha = \gamma(x) \tag{16}$$

where $\gamma(*)$ stands for a random combination of rotation, affine transformation, clipping, flipping, and other operations.

By introducing data augmentation into the M-train and the S-fit, we can get the goal of two phases, which can be expressed as

$$\min_{w,u} \frac{1}{n} \sum_{i=1}^{n} v_i ||h_u(f_w(\alpha_i)) - \alpha_i|| - \lambda v_i \quad \text{s.t. } v_i \in [0, 1] \tag{17}$$

and

$$\min_{L^S} \frac{1}{n} \sum_{i=1}^{n} v_i \left\| h_u\left(f_w\left(\alpha_i^j\right)\right) - \alpha_i^j \right\|^2 - \lambda v_i. \tag{18}$$

Explain boundary adaptation works in depth, with self-paced learning and data augmentation, we deflect the edge samples in the original space so that the edge samples tend to the correct class center. After each iteration, a new batch of edge samples is judged by self-paced learning. The boundary adaptation constructed by self-paced learning and data argumentation can effectively enhance the learning of boundary factors.

D. Logical Interpretability and Associated Code

A method of interpreting data in which the original data passes through an encoder and becomes a depth feature. The decoder is restored according to the depth feature. The smaller the loss value of the whole network is, the more important the value of the feature layer is.

For the multitask frame interpretation, the M-train stage identifies the density distribution of tasks and finds out a similar part of the probability distribution. The S-fit phase is specific to the base task, which differentiates it from other tasks at the micro level.

The explanation of boundary adaptation lies in the reinforcement of training boundary samples, which can get better knowledge. We also put restrictions on the transformation of the boundary adapters to prevent data clutter. The logic of the method as a whole can be explained as we get unsupervised data clustering by extracting representative important features and strengthening the whole process.

Logically, MTDC-BA follows a sequential structure, with M-train as the first stage and S-fit as the second. In the first stage of the M-train, we can decide whether the pretraining should be finished or not by the given loss derivative. In the second stage of S-fit, we rely on judgment to see if the sample classification continues to change to determine whether it should end. Formally, the stopping criteria are

$$1 - \frac{1}{n}\sum\left(s_i^t - s_i^{t-1}\right) < \delta \tag{19}$$

where s_i^t represents the clustering assignment in the tth iteration. In the experiments, we set $\delta = 0.0001$. The whole MTDC-BA is summarized in Algorithm 1.

Algorithm 1 MTDC-BA

Input: Multitask dataset $X = \{X^1, X^2, \ldots X^m\}$, number of clusters k, stop parameter δ, maximum iteration T, data mapping $\gamma(*)$

Output: Cluster assignments C

```
Initialize the dataset according to Eqn. (1)
Augment the original samples with Eqn. (16)
Initializes the cluster center C according to Eqn. (6)
Update the u and w as Eqn. (17)
for 0 ≤ t ≤ T do
    Update the cluster center C according to Eqn. (8)
    Update the edge samples V according to Eqn. (19)
    if 1 − (1/n) Σ(c_i^t − c_i^{t−1}) < 0.0001 then
        Break
    end if
end for
```

IV. Experiments

We conducted numerous experiments to verify the effectiveness of the proposed MTDC-BA. After completing the basic experiment settings, we compared them with the latest clustering methods. In order to explore the relevance of multitask data, the interpretability study was carried out based on the experiment results. Finally, the sensitivity of some hyper-parameters is analyzed.

A. Datasets

To verify the performance of the proposed method, we constructed four multitask image datasets and one nonimage dataset to conduct the experiments. These datasets are derived from NUS-wide [37], Caltech-256 [38], Cifar-10 [39], Cifar-100 [40], and MHRC [41]. Details of the datasets are shown in Table I, which includes the dataset names, the task of datasets, the category names, and the sample quantity of every class in every task. The number of samples is marked in parentheses.

B. Baseline Methods

To verify the performance of our proposed MTDC-BA, we performed horizontal and vertical comparisons. Horizontal compared with the latest multitask DC methods to determine that our method has a better performance, vertical compared with single-task DC methods to determine that the multitask learning method does improve. We compare the proposed MTDC-BA method with the following three baseline methods.

1) Typical single-task clustering: KM [42] and affinity propagation (AP) [43].
2) DC methods: DC [44], deep embedded clustering (DEC) [45], and deep adaptive clustering (DAC) [46].
3) Multitask clustering methods: LSSMTC [8], multitask Bregman clustering (MBC) [47], multitask multiview clustering (MTMVC) [48], multitask image clustering through correlation propagation (MICCP) [49], and deep correlation mining for multitask image clustering (DMTC) [49].

C. Experiment Setup

The network architecture used in this article is an AE of a CNN. The encoder adopts the DenseNet121 structure, and the output is the deep feature layer [50] with full connection and ReLU activation. The batch size is fixed to 256. In the M-train phase, we set the dimension of deep features to 50 and used the SGD optimizer with a learning rate of 1.0 and momentum of 0.9 to train the autoencoder with a maximum of 500 iterations. During the S-fit, we set the maximum number of iterations to 100 and set up the Adam Optimizer with an initial learning rate of 0.0001. The stopping criterion is designed δ less than 0.001. In the boundary adaptation, the transformation function is a random rotation of up to 20 pixels in each direction and a random shift of up to 0.2 pixels. Table II shows a summary of all the hyperparameter settings.

We ran each method 20 times and evaluated all clustering methods by clustering accuracy (ACC) and normalized mutual information (NMI). The ACC is defined as the best match between ground truth y and predicted label c

$$\mathrm{ACC}(y, c) = \arg\max \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i - g(c_i)\}}{n} \tag{20}$$

where y_i and c_i is the ground truth and predicted label of sample x_i and $g(*)$ is the mapping from predicted label to ground truth label. Through the Hungarian algorithm [51], we can get the best mapping. The NMI is defined as

$$\mathrm{NMI}(y, c) = \frac{2I(y, c)}{H(y) + H(c)} \tag{21}$$

TABLE I
SUMMARY ON THE DETAILS OF DATASETS

Datasets	Tasks	Class1	Class2	Class3	Class4	Class5	Class6
Caltech-256	Task1	Anamials.Giraffe(84)	Food.hotdog(85)	Ball.bowling(104)	Instruments.jean(92)	Appliance.TV(107)	Traffic.carriage(97)
	Task2	Anamials.Elephant(131)	Food.hamburger(86)	Ball.golf(98)	Instruments.piano(95)	Appliance.washing(84)	Traffic.fire(118)
	Task3	Anamials.Dog(102)	Food.pizza(104)	Ball.football(174)	Instruments.guzheng(100)	Appliance.refrigerator(84)	Traffic.bicycle(82)
		Class1	Class2	Class3			
Cifar-10	Task1	Aircraft.plane(1000)	Car.truck(1000)	Dog.border(1000)			
	Task2	Aircraft.fighter(5000)	Car.van(5000)	Dog.teddy(5000)			
		Class1	Class2	Class3	Class4	Class5	
Cifar-100	Task1	Mouse.hamster(400)	Tree.oak(400)	Cat.leopard(400)	Fish.dolphin(400)	Insect.bee(400)	
	Task2	Mouse.mouse(400)	Tree.palm(400)	Cat.lion(400)	Fish.whale(400)	Insect.beetle(400)	
	Task3	Mouse.squirrel(400)	Tree.pine(400)	Cat.tiger(400)	Fish.shark(400)	Insect.cockroach(400)	
		Class1	Class2	Class3	Class4		
NUS-wide	Task1	Anamial.cow(159)	Car.motorcycle(229)	Movement.ice_skating(223)	Natural.desert(236)		
	Task2	Anamial.coyote(247)	Car.truck(156)	Movement.basketball(193)	Natural.waterfalls(245)		
		Class1	Class2	Class3	Class4		
MHRC	Task1	Personal anxiety	Learning anxiety	Lonely	Guilt		
	Task2	sensitive	Physical symptoms	Fear	Impulse		

TABLE II
SUMMARY OF KEY HYPER-PARAMETER

Parameter	Value
maximum number of iterations	100
stopping letter criterion δ	0.0001
mapping γ	20/0.2
Batch_size	256
SGD_lr	1.0
SGD_momentum	0.9
deep_feature_dim	50

TABLE III
CLUSTERING PERFORMANCES OF DIFFERENT ALGORITHMS IN TERMS OF ACC AND NMI ON CALTECH-256

Method	Task1		Task2		Task3	
	ACC	NMI	ACC	NMI	ACC	NMI
KM	0.2561	0.1416	0.2523	0.0815	0.328	0.1425
AP	0.1998	0.1666	0.1881	0.1661	0.2407	0.1719
DC	0.2602	0.2231	0.2763	0.2223	0.2998	0.2716
DEC	0.2523	0.1876	0.2464	0.2908	0.3102	0.2264
DAC	0.2449	0.1392	0.2324	0.1487	0.2569	0.1411
LSSMTC	0.2869	0.1326	0.3592	0.1527	0.3676	0.1527
MBC	0.3123	0.1416	0.3388	0.1533	0.361	0.1655
MTMVC	0.3437	0.1875	0.321	0.1524	0.4043	0.2531
MICCP	0.5877	0.3024	0.5834	0.2993	0.4602	0.2624
DMTC	0.6778	0.5209	0.6059	0.4512	0.5643	**0.5723**
MTDC-BA	**0.8049**	**0.7064**	**0.7563**	**0.7302**	**0.5855**	0.5464

where $I(*)$ and $H(*)$ are mutual information and entropy, respectively. The scope of ACC and NMI is between 0 and 1. The value of NMI closer to 1 indicates that the model performs better.

TABLE IV
CLUSTERING PERFORMANCES OF DIFFERENT ALGORITHMS IN TERMS OF ACC AND NMI ON CIFAR-10

Method	Task1		Task2	
	ACC	NMI	ACC	NMI
KM	0.3254	0.0595	0.223	0.0442
AP	0.1226	0.0952	0.3225	0.0963
DC	0.1522	0.1114	0.0956	0.0997
DEC	0.3652	0.1801	0.4123	0.1617
DAC	0.3545	0.1795	0.4042	0.0979
LSSMTC	0.3661	0.121	0.3426	0.1523
MBC	0.3056	0.0236	0.2924	0.089
MTMVC	0.4523	0.1704	0.2953	0.1356
MICCP	0.2263	0.0149	0.1534	0.0198
DMTC	0.5461	0.1805	0.5125	0.1523
MTDC-BA	**0.5927**	**0.1808**	**0.5936**	**0.1735**

TABLE V
CLUSTERING PERFORMANCES OF DIFFERENT ALGORITHMS IN TERMS OF ACC AND NMI ON CIFAR-100

Method	Task1		Task2		Task3	
	ACC	NMI	ACC	NMI	ACC	NMI
KM	0.3115	0.0694	0.2855	0.0311	0.2665	0.0253
AP	0.1012	0.2526	0.0931	0.1964	0.0925	0.2719
DC	0.3298	0.2898	0.2317	0.1296	0.2818	0.2795
DEC	0.4331	0.1793	0.3642	0.1564	0.4243	0.2615
DAC	0.3152	0.1712	0.2832	0.1437	0.2895	0.1694
LSSMTC	0.2272	0.0271	0.2309	0.0161	0.2176	0.0212
MBC	0.2056	0.0148	0.2216	0.0203	0.246	0.0216
MTMVC	0.3437	0.1875	0.321	0.1524	0.4043	0.2531
MICCP	0.2406	0.0242	0.2367	0.0303	0.2415	0.0206
DMTC	0.6717	0.4085	0.4925	0.2859	0.5662	0.3148
MTDC-BA	**0.7105**	**0.4712**	**0.5748**	**0.3223**	**0.6315**	**0.4207**

D. Experiment Results

Tables III–VII present the quantitative results of ACC and NMI for the testing datasets described in Section IV-A.

1) Compared With Traditional Clustering Methods: In this section, the proposed MTDC-BA is compared with the classical clustering algorithms KM and AP. In preparation, we compare the performance of MTDC-BA on five datasets with the best results obtained using these classic algorithms on different tasks. For example, the experiment results on the NUS-wide show that the ACC and NMI of MTDC-BA are higher than classical algorithms, as shown in Table VI. There are two main reasons for such a gap: 1) traditional algorithms do not rely on the deep features of high-level abstract semantics, and are prone to fall into the trap of local optimization; and 2) the traditional algorithm does not make full use of the external/shared information of other clustering

TABLE VI
CLUSTERING PERFORMANCES OF DIFFERENT ALGORITHMS IN TERMS OF ACC AND NMI ON NUS-WIDE

Method	Task1		Task2	
	ACC	NMI	ACC	NMI
KM	0.369	0.0409	0.3149	0.0455
AP	0.2479	0.0401	0.2751	0.0355
DC	0.2083	0.1652	0.2651	0.2084
DEC	0.1747	0.124	0.2025	0.1552
DAC	0.3304	0.1723	0.2973	0.1815
LSSMTC	0.3405	0.0237	0.3281	0.0244
MBC	0.2895	0.0392	0.3272	0.0387
MTMVC	0.3546	0.0392	0.3574	0.0707
MICCP	0.5877	0.1954	0.5665	0.1859
DMTC	0.5903	0.2597	0.6038	0.2767
MTDC-BA	**0.8198**	**0.6437**	**0.9608**	**0.8823**

TABLE VII
CLUSTERING PERFORMANCES OF DIFFERENT ALGORITHMS IN TERMS OF ACC AND NMI ON MHRC

Method	Task1		Task2	
	ACC	NMI	ACC	NMI
KM	0.4543	0.1523	0.4739	0.2205
AP	0.1566	0.0567	0.1447	0.0981
DC	0.4062	0.2241	0.3969	0.1597
DEC	0.3925	0.1582	0.3809	0.0959
DAC	0.3655	0.225	0.3091	0.2673
LSSMTC	0.4096	0.1121	0.5574	0.2441
MBC	0.4148	0.2564	0.4238	0.3145
MTMVC	0.5124	0.1552	0.5532	0.1578
MICCP	0.5872	0.1752	0.4224	0.1752
DMTC	0.5446	0.2975	0.6142	0.2453
MTDC-BA	**0.6324**	**0.3254**	**0.7571**	**0.3511**

tasks, and the ability of feature extraction is not as strong as MTDC-BA.

2) Compared With DC Methods: Since the proposed MTDC-BA is an extension/upgrade of a deep cluster algorithm, it is necessary to compare it with other deep cluster methods to determine the advanced nature of MTDC-BA. We compare MTDC-BA with the popular DC methods. In short, the datasets for single tasks are divided, and running relevant code or referencing data from published papers to get the best ACC and NMI comparison. From the corresponding results, we can see whether the performance of MTDC-BA has been improved.

As shown in Tables III–VII, the latest and mainstream DC algorithms are selected, but their results are not as good as MTDC-BA. The network of MTDC-BA uses the DenseNet format, which can better extract the deep features of the data. What's more, MTDC-BA considers the influence of boundary factors and learns boundary samples better through boundary adapters. In addition, MTDC-BA has introduced multitasking to improve performance, and we will conduct further experiments in Section IV-G to confirm that this improvement does exist.

3) Compared With Multitask DC Methods: Since the proposed method introduces multitask learning for performance boosting, typical methods for multitask clustering are compared. As can be seen from Tables III to VII, the MTDC-BA is better than the existing multitask DC algorithms. Specifically, the performance of LSSMTC and MTMVC is low because the two methods assume that tasks are completely related, and there will be negative migration on the real dataset due to incomplete correlation, which will affect the performance of the algorithm. The result of MBC is not ideal, because it only uses distance to measure task correlation, and less knowledge is gained, so the promotion of single-task performance is low. The final performance of DMTC was also inferior to the MTDC-BA because DMTC used AlexNet, an earlier network framework. Compared with MTDC-BA, the ability of DMTC to extract representative features is slightly insufficient.

E. Visualization Results

To verify the performance of MTDC-BA in efficiently extracting deep features from image data, we conduct the experiment for visualizing the clustering results. The experiment used t-distributed stochastic neighbor embedding (t-SNE) to visualize the extracted deep features in 2-D space and labeled the final clustering results [52]. The experiment results are shown in Fig. 4. Easily observed, MTDC-BA works best on NUS-wide, which means that the result has the clearest clustering allocation. On the contrary, there is some overlap in the image mapping of each cluster class, which is similar to the actual ACC and NMI in Section IV-D.

Comparing the results of four datasets, the final results are related to the dataset containing the cluster class. In detail, the more categories a dataset contains that are related to each other, the denser the clustering in the visualization results. In extension, the MTDC-BA can construct the mapping of original samples in a new space through multitask knowledge. There is a higher density of clusters in the new data space, with more obvious characteristics. This also proves that MTDC-BA can make efficient use of multitasking knowledge, greatly improving the performance of the current single task. The interpretability of the method can be clearly concluded from this section. The data layer of the whole model can be divided into three layers, which are the original data layer, feature layer, and restore layer.

F. Convergence Study

The convergence curve of DMTC-BA-based four image datasets is shown in Fig. 5. Noting that although the convergence performance of the nonimage data MHRC is not included, in fact, the MTDC-BA can complete the convergence on the MHRC after the start of 20 iterations. It can be observed that the proposed MTDC-BA has good convergence in four different datasets. As can be seen, after 200 iterations, the overall model tends to be stable. MTDC-BA converges rapidly after 30 iterations for both Cifar-10 and Cifar-100 datasets. For NUS-wide, the MTDC-BA converges after 20 iterations.

Specifically, we found reasons for faster convergence on the last three datasets. The original feature dimensions of the latter three datasets are relatively small, and the DenseNet structure in MTDC-BA can efficiently mine features in images, especially moderate images like the latter three datasets. The Caltech-256 samples have more than 200 000 original dimensions, so the MTDC-BA needs more iterations to find the correct solution.

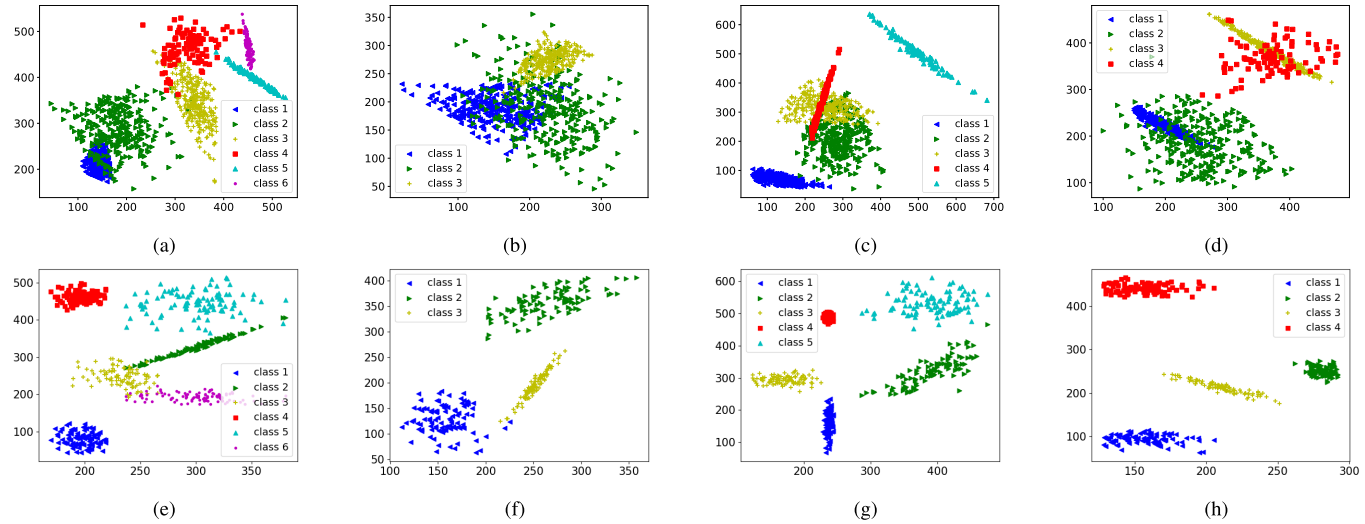


Fig. 4. Visualization of input and deep features on four image datasets. Note that the classes in the legend correspond to the classes in the dataset introduction. (a) Input feature of Caltech-256. (b) Input feature of Cifar-10. (c) Input feature of Cifar-100. (d) Input feature of NUW-wide. (e) Deep feature of Caltech-256. (f) Deep feature of Cifar-10. (g) Deep feature of Cifar-100. (h) Deep feature of NUW-wide.

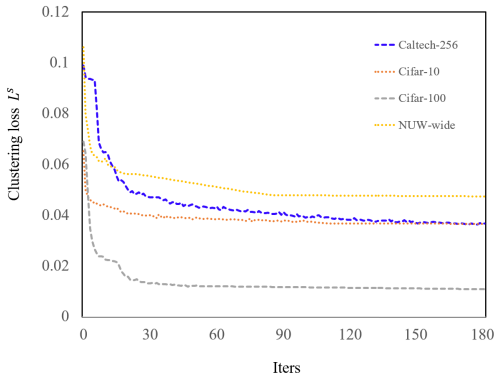


Fig. 5. Convergence curves of the proposed MTDC-BA on four image datasets. Obviously, the model converges in significantly different manners, which is related to the properties of the datasets.

G. Discovery of Multitask Learning

Compare the performance of the model in single-task and multitask situations to determine whether multitask knowledge is beneficial to improving single-task clustering to avoid the differences caused by the number of datasets. The random sampling is used to equalize the sample size and the multitask data and the single-task data have the same data size. The control variable method was used in the whole experiment, and the experiment parameters were kept the same.

The performance comparison of the model with and without multitask learning is shown in Fig. 6, from which we can see that the performance has been consistently improved with regard to ACC. Because multitask learning is assisted by multitask knowledge, MTDC-BA understands which part of the knowledge characteristics are more important. For these important features, MTDC-BA will be preferred.

In the absence of the guidance of multitask learning, the model will treat all features equally. Without any discrimination leads a balance for parameter resources of the whole model. It also means the model parameter resources assigned to important features will reduce the performance of ACC. From the point of view of the NMI index, the difference in NMI of each task before and after adding multitask learning is not big, which is because NMI represents the similarity within the cluster, and the main influencing factor is the overall framework of the algorithm. The hyperparameters remain unchanged in the experiment, which also leads to a small gap in NMI.

Failure cases also appeared in the experiment. We can find the NMI performed better before adopting multitask learning. This also means our multitasking learning is not omnipotent. Observing all the tasks that produce the failure cases, these NMI are not ideal and lower than 0.4 before applying multitask learning. Presumably, if there is less correlation between the samples in one dataset, the multitask learning is a failure. Multitask learning makes use of the relevance or similarity among the samples and draws lessons from the classification knowledge among the tasks. When the relevance is low, multitasking learning can not recognize the rule of data classification, resulting in negative learning.

In conclusion, our experiment established that multitasking learning can enhance the evaluation of indicators in certain situations. Certain situations refer to the exact existence of exploitable correlations in the dataset. The MTDC-BA borrows multitask learning to promote model performance.

H. Sensitivity of Hyper-Parameters

Fig. 7 shows how the ACC and NMI change over the five datasets as the hyper-parameters change. With the increase of the maximum number of iterations, ACC and NMI increase continuously before convergence. After the number of iterations reaches 100, the curves about the five datasets tend to be stable. Continuing to increase the number of iterations may result in a negative fit, after taking into account the performance of the five datasets and setting the maximum number of iterations as 100.

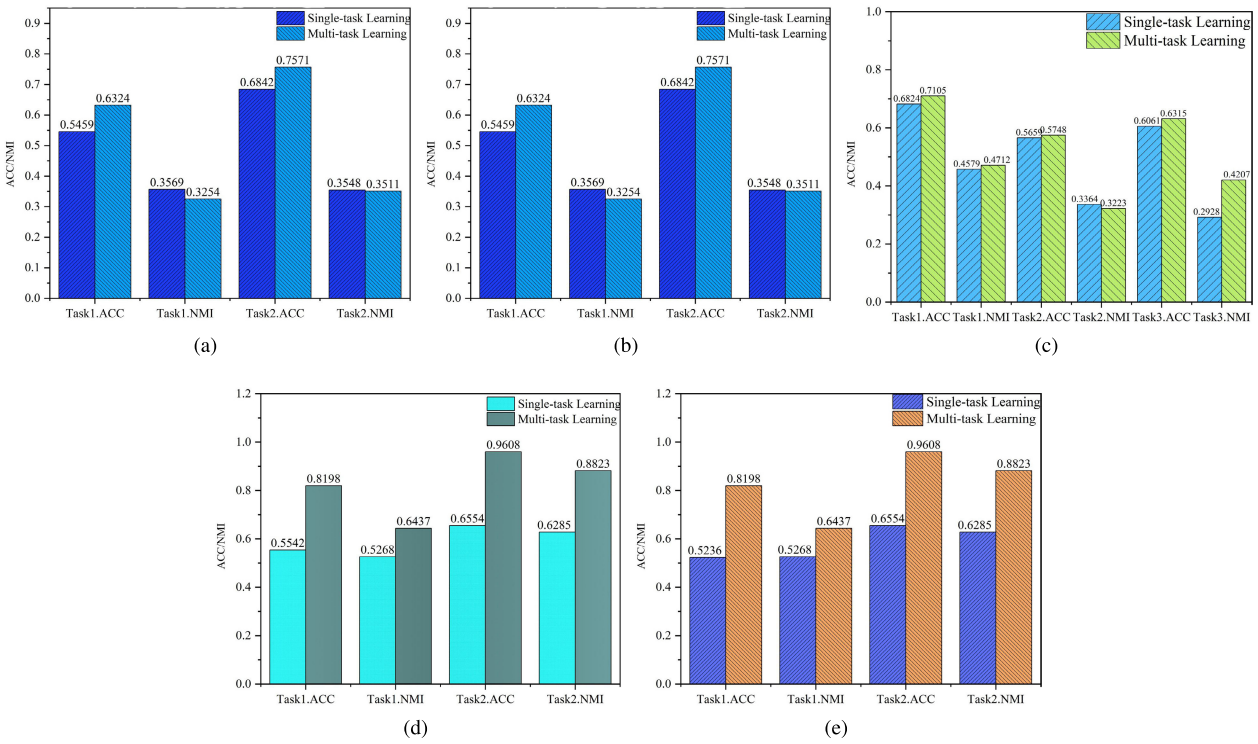


Fig. 6. Performance comparison with combined multitask learning-based datasets. (a) Based on Caltech-256. (b) Based on Cifar-10. (c) Based on Cifar-100. (d) Based on NUW-wide. (e) Based on MHRC.

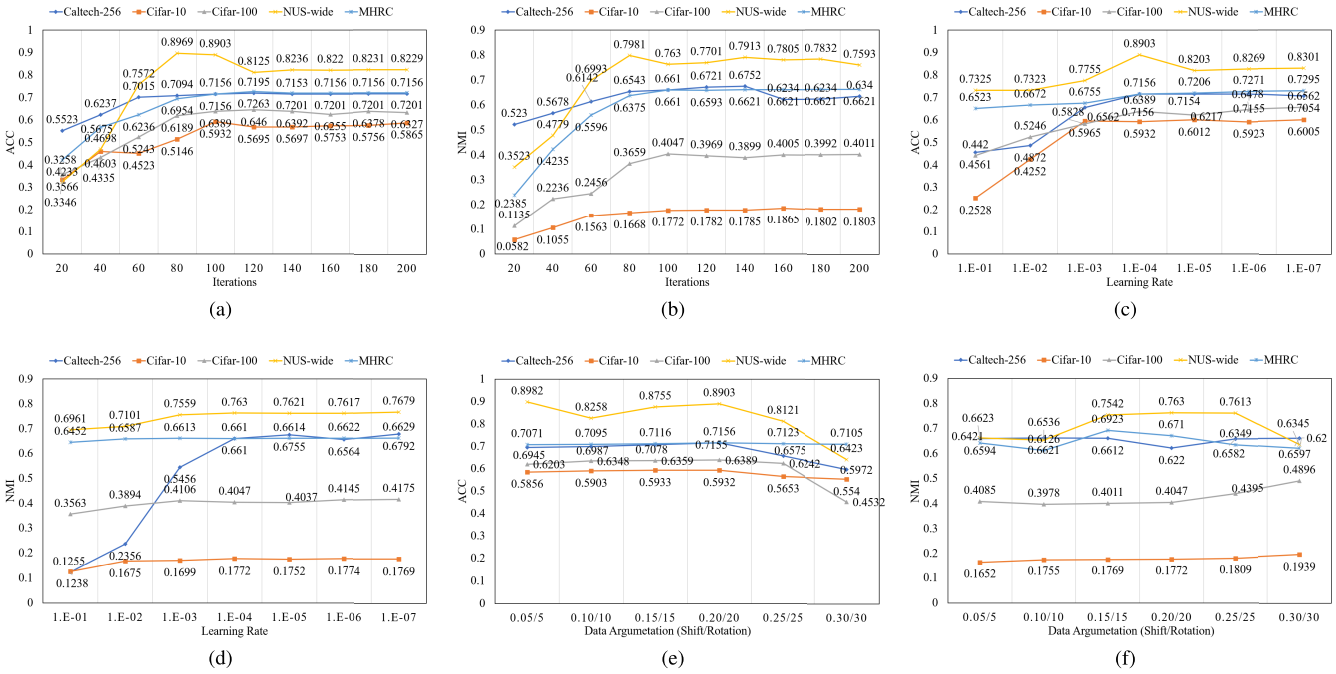


Fig. 7. Sensitivity analysis of a maximum number of iterations, optimizer Adam learning rate parameter, and data argumentation function γ. (a) Trend of ACC with iterations increases. (b) Trend of NMI with iterations increases. (c) Trend of ACC with learning rate. (d) Trend of NMI with learning rate. (e) ACC trend with different data mapping. (f) NMI trend with different data mapping.

The parameters of the Adam Optimizer LR are also part of the experiment. The ACC and NMI tend to be stable when LR after 0.0001. The learning rate has a larger effect on ACC and a smaller effect on NMI. The learning rate can affect the accuracy of model learning. The model easier to discover features and details with a low learning rate, but costs longer study time and more space. The NMI analysis shows that the model framework has a strong influence on learning rate.

Data enhancement techniques require the use of translation and cropping-related parameters. The final result shows that

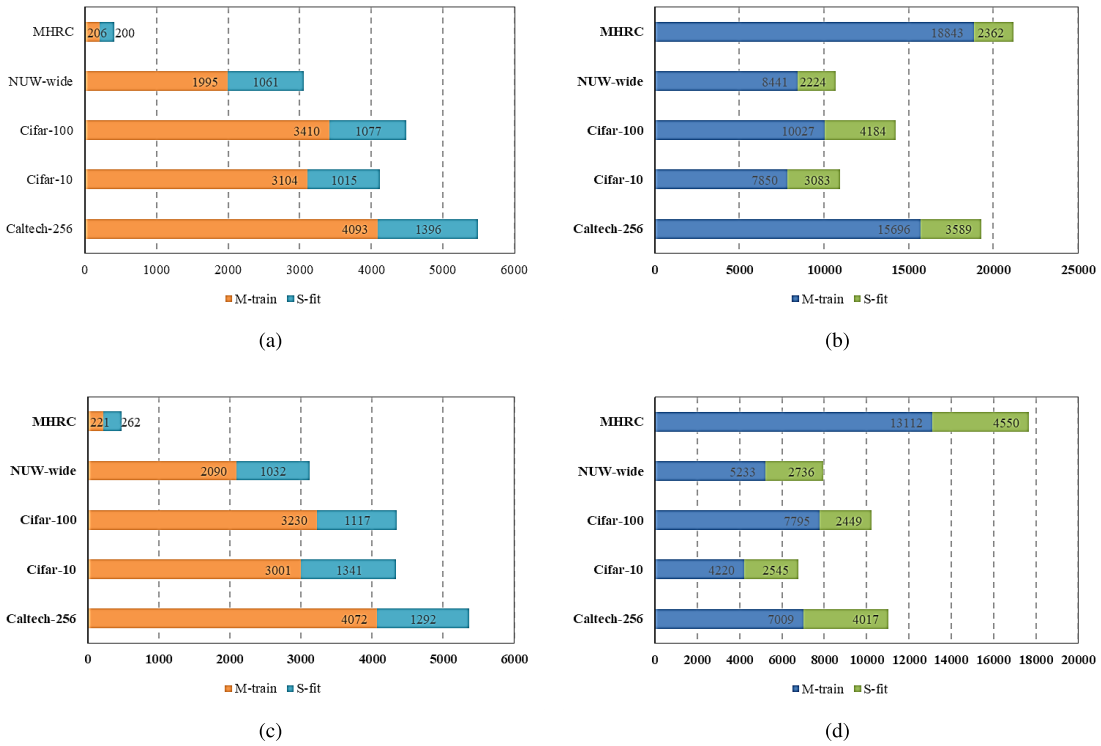


Fig. 8. Time and space consumption analysis on five datasets-based different task numbers. Note that the results are divided into two phases, taking the maximum consumption in each phase. The results have no units, and only represent the size, based on the minimum task and the smallest single data consumption of time and space consumption comparison. (a) Time consumption of all datasets-based two tasks. (b) Space consumption of all datasets-based two tasks. (c) Time consumption of all datasets-based one task. (d) Space consumption of all datasets-based one task.

the best performance of the model is obtained when the translation is 20 pixels and the rotation is 20 pixels. This is because the data obtained by small offset and rotation are not different from the original data and can not achieve the goal of enhancing learning. The larger rotation and migration make the expanded data not have the original features, and the algorithm model can not accurately learn from the data with large differences in features. Furthermore, the reasonable translation and rotation range of the image dataset is larger, nonimage data (ordinary text data) translation and rotation range is smaller. This is because the image itself has a higher dimension and a wider data enhancement scope. Text data features are few and easily influenced by external features, so the enhancement range is small.

I. Time and Space Consumption Analysis

In order to verify the stable rows of MTDC-BA, a time and space consumption analysis is performed in Fig. 8. In terms of time consumption, the performance of MTDC-BA is related to dataset size and feature dimensions. The first phase of the five datasets consumed more than the second, independent of the number of multitasking tasks. In terms of space consumption, the performance of MTDC-BA is related to the number of tasks and the space consumption in the first stage increases with the number of tasks.

The consumption situation based on the MHRC dataset is representative. From the time trumpet, M-train multitask knowledge mining runs in parallel with multiple threads. MTDC-BA gets the result very quickly but costs nearly ten times more memory space because the number of tasks increases. The number of multitasks affects the consumption of space and more time. The overall performance of MTDC-BA is stable, and consistent for time control, but it requires sufficient memory space support.

V. Conclusion

In this study, an MTDC-BA model is proposed, which can perform multitask association learning when datasets are related. MTDC-BA conducts two-phase learning: 1) the M-train works for capturing and storing the knowledge between the multitask datasets and 2) the S-fit is responsible for utilizing the stored multitask knowledge and performing the single-task clustering. Both phases eliminate boundary effects by boundary adaptors composed of data enhancement and self-learning. The results on five typical datasets show that the proposed MTDC-BA method is superior to the existing methods. The experiment on multitask learning proves MTDC-BA can effectively mine multitask knowledge, which aims to produce a more intuitive data space.

In the future, it is expected to integrate better mechanisms into the MTDC-BA framework, such as adversarial autoencoder. In terms of learning style, unsupervised learning for big data can incorporate more anthropomorphic approaches, such as active learning. The boundary adaptation can be extended to be an active learning machine that submits learning content for raw data. However, there are many challenges, active learning requires a lot of manual labor, which will lead to cost surges, out of control.

REFERENCES

[1] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020.

[2] L. Yang, W. Fan, and N. Bouguila, "Clustering analysis via deep generative models with mixture models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 1, pp. 340–350, Jan. 2022.

[3] K. Zeng, M. Ning, Y. Wang, and Y. Guo, "Hierarchical clustering with hard-batch triplet loss for person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13654–13662.

[4] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[5] X. Liu et al., "Localized simple multiple kernel k-means," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9273–9281.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–13.

[7] D. Le, Z. Aldeneh, and E. M. Provost, "Discretized continuous speech emotion recognition with multi-task deep recurrent neural network," in *Proc. Interspeech*, Aug. 2017, pp. 1108–1112.

[8] Q. Gu and J. Zhou, "Learning the shared subspace for multi-task clustering and transductive transfer classification," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 159–168.

[9] R. Gargees, J. M. Keller, and M. Popescu, "TLPCM: Transfer learning possibilistic C-means," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 4, pp. 940–952, Jun. 2020.

[10] R. Satkunasivam et al., "A phase II clinical trial of neoadjuvant sasanlimab and stereotactic body radiation therapy as an in situ vaccine for cisplatin-ineligible MIBC: The RAD VACCINE MIBC trial," *Future Oncol.*, vol. 18, no. 25, pp. 2771–2781, Aug. 2022.

[11] M. O. Omorogie, M. T. Agbadaola, A. M. Olatunde, B. Helmreich, and J. O. Babalola, "Surface equilibrium and dynamics for the adsorption of anionic dyes onto MnO_2/biomass micro-composite," *Green Chem. Lett. Rev.*, vol. 15, no. 1, pp. 51–60, Jan. 2022.

[12] S. Yaghoubi, R. Farnoosh, and M. H. Behzadi, "Model-based clustering (MBC) for road data via multivariate mixture of normal distributions and factor analysis (FA)," *J. Statist. Manage. Syst.*, vol. 2022, pp. 1–11, Jul. 2022.

[13] X. Zhang, X. Zhang, H. Liu, and X. Liu, "Partially related multi-task clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2367–2380, Dec. 2018.

[14] X. Zhang, H. Liu, X. Zhang, and X. Liu, "Attributed graph clustering with multi-task embedding learning," *Neural Netw.*, vol. 152, pp. 224–233, Aug. 2022.

[15] S. Zhou et al., "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," 2022, *arXiv:2206.07579*.

[16] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, J. Ruiz-Shulcloper and G. Sanniti di Baja, Eds. Berlin, Germany: Springer, 2013, pp. 117–124.

[17] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4061–4070.

[18] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham, Switzerland: Springer, 2017, pp. 373–382.

[19] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5747–5756.

[20] K.-L. Lim, X. Jiang, and C. Yi, "Deep clustering with variational autoencoder," *IEEE Signal Process. Lett.*, vol. 27, pp. 231–235, 2020.

[21] X. Guo et al., "Adaptive self-paced deep clustering with data augmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 9, pp. 1680–1693, Sep. 2020.

[22] I. Shahin, A. B. Nassif, and S. Hamsa, "Emotion recognition using hybrid Gaussian mixture model and deep neural network," *IEEE Access*, vol. 7, pp. 26777–26787, 2019.

[23] X. Shu, Y. Yang, and B. Wu, "A neighbor level set framework minimized with the split Bregman method for medical image segmentation," *Signal Process.*, vol. 189, Dec. 2021, Art. no. 108293.

[24] Q. Wang, Z. Ding, Z. Tao, Q. Gao, and Y. Fu, "Partial multi-view clustering via consistent GAN," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1290–1295.

[25] F. Liu, L. Jiao, and X. Tang, "Task-oriented GAN for PolSAR image classification and clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2707–2719, Sep. 2019.

[26] Z. Yang, J. Wen, and C. Davatzikos, "Smile-GANs: Semi-supervised clustering via GANs for dissecting brain disease heterogeneity from medical images," 2020, *arXiv:2006.15255*.

[27] X. Guo, Q. Zhao, D. Zheng, Y. Ning, and Y. Gao, "A short-term load forecasting model of multi-scale CNN-LSTM hybrid neural network considering the real-time electricity price," *Energy Rep.*, vol. 6, pp. 1046–1053, Dec. 2020.

[28] R. Yan, J. Liao, J. Yang, W. Sun, M. Nong, and F. Li, "Multi-hour and multi-site air quality index forecasting in Beijing using CNN, LSTM, CNN-LSTM, and spatiotemporal clustering," *Expert Syst. Appl.*, vol. 169, May 2021, Art. no. 114513.

[29] F. Yu, D. Wei, S. Zhang, and Y. Shao, "3D CNN-based accurate prediction for large-scale traffic flow," in *Proc. 4th Int. Conf. Intell. Transp. Eng. (ICITE)*, Sep. 2019, pp. 99–103.

[30] Y. Bai, X. Yan, Y. Jiang, S.-T. Xia, and Y. Wang, "Clustering effect of adversarial robust models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 29590–29601.

[31] W. Zheng, X. Zhu, G. Wen, Y. Zhu, H. Yu, and J. Gan, "Unsupervised feature selection by self-paced learning regularization," *Pattern Recognit. Lett.*, vol. 132, pp. 4–11, Apr. 2020.

[32] H. Yu, G. Wen, J. Gan, W. Zheng, and C. Lei, "Self-paced learning for k-means clustering algorithm," *Pattern Recognit. Lett.*, vol. 132, pp. 69–75, Apr. 2020.

[33] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[34] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A survey on data augmentation for text classification," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–39, Jul. 2023.

[35] J. Nalepa, M. Marcinkiewicz, and M. Kawulok, "Data augmentation for brain-tumor segmentation: A review," *Frontiers Comput. Neurosci.*, vol. 13, p. 83, Dec. 2019.

[36] S. Bidnur, D. Srikanth, and S. Gurugopinath, "Resource-conscious high-performance models for 2D-to-3D single-view reconstruction," in *Proc. IEEE Region 10 Conf. (TENCON)*, Dec. 2021, pp. 681–686.

[37] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world Web image database from National University of Singapore," in *Proc. ACM Int. Conf. Image Video Retr.*, Jul. 2009, pp. 1–6, doi: 10.1145/1646396.1646452.

[38] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object category dataset," California Inst. Technol., Tech. Rep. 7694, 2007.

[39] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, "CIFAR-10: KNN-based ensemble of classifiers," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2016, pp. 1192–1195.

[40] S. Singla, S. Singla, and S. Feizi, "Improved deterministic l_2 robustness on CIFAR-10 and CIFAR-100," in *Proc. Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://openreview.net/forum?id=tD7eCtaSkR

[41] H. Liu, Y. Shi, E. Auden, and S. Rozelle, "Anxiety in rural Chinese children and adolescents: Comparisons across provinces and among subgroups," *Int. J. Environ. Res. Public Health*, vol. 15, no. 10, p. 2087, 2018.

[42] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003.

[43] U. Bodenhofer, A. Kothmeier, and S. Hochreiter, "APCluster: An R package for affinity propagation clustering," *Bioinformatics*, vol. 27, no. 17, pp. 2463–2464, Sep. 2011.

[44] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 132–149.

[45] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[46] X. Chen, J. Z. Haung, F. Nie, R. Chen, and Q. Wu, "A self-balanced min-cut algorithm for image clustering," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2080–2088.

[47] J. Zhang and C. Zhang, "Multitask Bregman clustering," *Neurocomputing*, vol. 74, no. 10, pp. 1720–1734, May 2011.

[48] X. Zhang, X. Zhang, H. Liu, and X. Liu, "Multi-task multi-view clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3324–3338, Dec. 2016.

[49] X. Yan, K. Shi, Y. Ye, and H. Yu, "Deep correlation mining for multi-task image clustering," *Expert Syst. Appl.*, vol. 187, Jan. 2022, Art. no. 115973.
[50] Y. Zhu and S. Newsam, "DenseNet for dense flow," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 790–794.
[51] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The dynamic Hungarian algorithm for the assignment problem with changing costs," Robot. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-07-27, 2007.
[52] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 1–12, 2008.

Xiaobo Zhang (Member, IEEE) received the M.S. and Ph.D. degrees from Southwest Jiaotong University (SWJTU), Chengdu, China, in 2011 and 2021, respectively.

He is currently working as an Assistant Research Fellow with the School of Computing and Artificial Intelligence, SWJTU. He is also a Visiting Scholar at the National University of Singapore, Singapore. His current research interests include multitasking learning, medical data mining, multiview learning, clustering, semisupervised learning, deep learning, and visualization.

Tao Wang received the B.S. degree from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, in 2020, where he is currently pursuing the master's degree with the School of Computer and Artificial Intelligence.

His research interests include computer vision and image processing, machine learning and deep learning, and medical image analysis.

Xiaole Zhao (Member, IEEE) received the B.S. and M.S. degrees from the School of Computer Science and Technology, Southwest University of Science and Technology (SWUST), Mianyang, China, in 2013 and 2016, respectively, and the Ph.D. degree from the School of Life Science and Technology, University of Electronic Science and Technology of China (UESTC), Chengdu, China.

He is currently working as an Assistant Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University (SWJTU), Chengdu. His research interests mainly include computer vision and image processing, machine learning and deep learning, and medical image analysis.

Dengmin Wen received the B.S. and M.S. degrees from Southwest Jiaotong University(SWJTU), Chengdu, China, in 1983 and 1988, respectively.

He is currently working as an Assistant Professor with the School of Computing and Artificial Intelligence, SWJTU. His research interests mainly include computer applications, software architecture research, and object-oriented technology.

Donghai Zhai received the B.E., M.S., and Ph.D. degrees from Southwest Jiaotong University, Chengdu, China, in 1997, 2000, and 2003, respectively.

From 2003 to 2005, he was employed at IBM China Research Laboratory, Beijing, China, where he performed system analyses and architecture design. Since 2006, he has been associated with the Department of Software Engineering, Southwest Jiaotong University. He is also a Visiting Scholar at Louisiana State University (LSU), Baton Rouge, LA, USA. His research interests include autonomous driving, digital image processing, computer vision, and pattern recognition.