Deep Tensor Spectral Clustering Network via Ensemble of Multiple Affinity Tensors

Hongmin Cai, *Senior Member, IEEE*, Yu Hu, Fei Qi, Bin Hu, *Fellow, IEEE*, and Yiu-ming Cheung, *Fellow, IEEE*

***Abstract*—Tensor spectral clustering (TSC) is an emerging approach that explores multi-wise similarities to boost learning. However, two key challenges have yet to be well addressed in the existing TSC methods: (1) The construction and storage of high-order affinity tensors to encode the multi-wise similarities are memory-intensive and hampers their applicability, and (2) they mostly employ a two-stage approach that integrates multiple affinity tensors of different orders to learn a consensus tensor spectral embedding, thus often leading to a suboptimal clustering result. To this end, this paper proposes a tensor spectral clustering network (TSC-Net) to achieve one-stage learning of a consensus tensor spectral embedding, while reducing the memory cost. TSC-Net employs a deep neural network that learns to map the input samples to the consensus tensor spectral embedding, guided by a TSC objective with multiple affinity tensors. It uses stochastic optimization to calculate a small part of the affinity tensors, thereby avoiding loading the whole affinity tensors for computation, thus significantly reducing the memory cost. Through using an ensemble of multiple affinity tensors, the TSC can dramatically improve clustering performance. Empirical studies on benchmark datasets demonstrate that TSC-Net outperforms the recent baseline methods.**

***Index Terms*—Clustering, tensor spectral clustering, deep neural networks, clustering ensemble.**

I. Introduction

CLUSTERING aims at partitioning samples into underlying clusters in an unsupervised manner. Over the past decades, clustering has been widely used in plenty of fields, including computer vision [1] and bioinformatics [2], to name a few. Among various clustering techniques, spectral clustering (SC) [3], [4] is a common one due to its simplicity and graph-theoretic interpretation. Nevertheless, data in computer vision or bioinformatics are complex [5], [6], where they are often high-dimensional and contaminated by noise [7]. SC relies on an affinity matrix to characterize pairwise similarities, which fall short in providing satisfactory clustering performance for such complex data [8], [9]. Converging evidence [10], [11] suggests that dealing with high-dimensional and noisy data requires characterizing more complex similarities, and thereafter tensor spectral clustering (TSC) [12] has been developed recently as a promising solution. TSC employs high-order affinity tensors to characterize multi-wise similarities among samples instead of merely pairwise similarities as in previous methods. It has been shown that such high-order affinity tensors are robust against noise and capable of characterizing comprehensive spatial structure for high-dimensional data to achieve a better performance [13].

The seminal TSC method was proposed in [11], [14], which constructs a third order affinity tensor to encode ternary similarities and utilizes the multilinear singular value decomposition (SVD) of the constructed affinity tensor to yield a tensor spectral embedding, which is an approximation of the cluster indicator matrix. Recently, another TSC method, IPS2 [13], has shown that multiple affinity tensors of different orders have complementary information to each other for clustering, and integration of them can improve the accuracy and robustness of clustering. IPS2 specifically proposes an integrative method that combines a fourth order affinity tensor with a second order affinity tensor, i.e., an affinity matrix, to learn a consensus tensor spectral embedding, which has been shown to boost the clustering performance in several benchmark datasets. Although TSC methods have advanced so far, two key challenges that hamper their applicability remain less explored.

One crucial challenge is that TSC methods need to construct and store the whole high-order affinity tensor, which is memory-intensive. If one is to construct a Kth order affinity tensor for m samples, the general memory cost is $\mathcal{O}(m^K)$. Taking $K = 3$ and $m = 1000$ as an example, one can check that the memory cost is nearly 7.451 GB if the resulting affinity tensor is stored in double-precision floating points. Such a prohibitively high memory cost becomes an insurmountable roadblock and severely inhibits the applicability of TSC. To alleviate this issue, most TSC methods have adopted sampling techniques such as column sampling [10], Nyström approximation [11], or iterative

Manuscript received 8 February 2023; revised 15 October 2023; accepted 29 January 2024. Date of publication 5 February 2024; date of current version 5 June 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFE0112200, in part by the National Natural Science Foundation of China under Grants U21A20520 and 62325204, in part by the Science and Technology Project of Guangdong Province under Grant 2022A0505050014, in part by the Key-Area Research and Development Program of Guangzhou City under Grant 202206030009, in part by the NSFC / Research Grants Council (RGC) Joint Research Scheme under Grant N_HKBU214/21, and in part by the General Research Fund of RGC under Grants 12201321, 12202622, and 12201323, and in part by RGC Senior Research Fellow Scheme under Grant SRFS2324-2S02. Recommended for acceptance by M. Salzmann. *(Corresponding author: Yiu-ming Cheung.)*

Hongmin Cai, Yu Hu, and Fei Qi are with the Department of Computer Science and Engineering, South China University of Technology, Guangdong 510641, China (e-mail: hmcai@scut.edu.cn).

Bin Hu is with the School of Medical Technology, Beijing Institute of Technology, Beijing 100811, China (e-mail: bh@bit.edu.cn).

Yiu-ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, 999077, Hong Kong (e-mail: ymc@comp.hkbu.edu.hk).

Digital Object Identifier 10.1109/TPAMI.2024.3361912

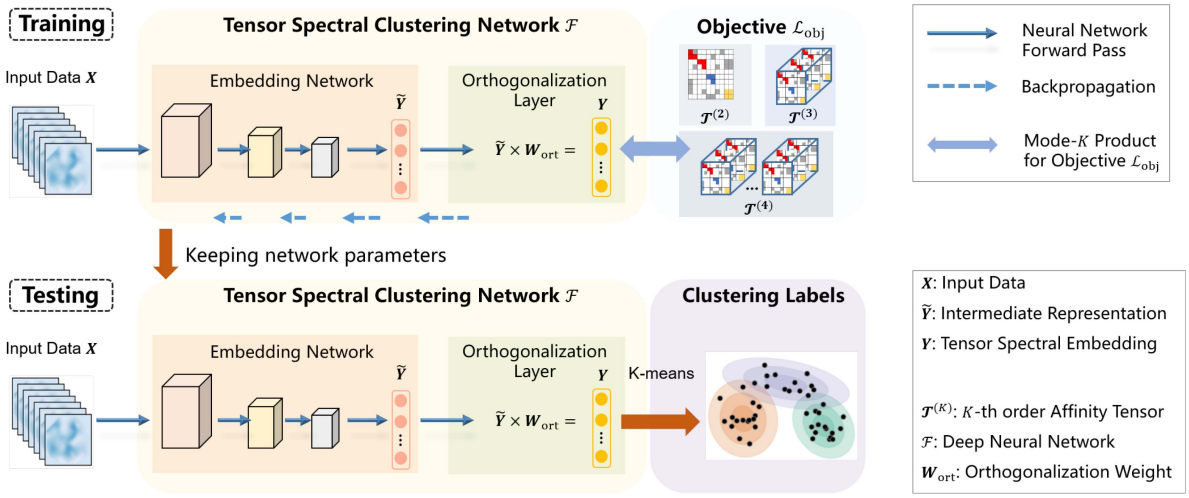


Fig. 1. Workflow of the proposed tensor spectral clustering network (TSC-Net). The notations used in this figure are defined in Table I. TSC-Net maps the input data to the tensor spectral embedding under the corresponding objective, where the orthogonalization layer is at the top to ensure that the output meets the orthogonal constraint.

sampling [12] to construct a sparse affinity tensor, in which most of the elements are zero. However, sampling techniques inevitably incur the information loss of the affinity tensor and may result in performance degradation.

The other challenge is that the existing TSC methods mostly employ a two-stage approach that integrates multiple affinity tensors to learn a consensus tensor spectral embedding, which often leads to a suboptimal clustering result. For example, the representative IPS2 [13] has proposed a two-stage approach, which first solves the tensor spectral embedding from a fourth order affinity tensor and the second order affinity tensor independently and then yields a consensus one by simply averaging them. Though demonstrating promising clustering performance in several benchmark datasets, the above two-stage approach disconnects the multiple affinity tensors in the process of reaching the consensus tensor spectral embedding and may lead to unsatisfactory performance.

Recently, deep neural network (DNN) has become a popular technique to learn underlying nonlinear mappings [15] in machine learning and computer vision. Since TSC methods can be regarded as a nonlinear mapping from an original sample space to an embedding space, a modern DNN with nonlinear activation functions is able to approximate such a nonlinear mapping [15], [16], [17]. As such, the DNN can be applied to learn the mapping from the input samples to the tensor spectral embedding. Subsequently, this paper aims to develop a DNN-based TSC method to reduce memory cost, while providing a joint framework to integrate multiple affinity tensors of different orders.

Accordingly, a tensor spectral clustering network, abbreviated as TSC-Net, is proposed. The TSC-Net is designed to learn a consensus tensor spectral embedding to integrate multiple affinity tensors in a stacked neural network. We instantiate the integrated method with the second, third, and fourth order affinity tensors. The stochastic gradient descent is thus applied to optimize the TSC-Net. The stochastic optimization allows us to calculate a small part of the affinity tensors at each time while circumventing storing the whole affinity tensors, thus decreasing the memory cost by a significant amount. For example, if the mini-batch size of the stochastic gradient descent is m_b, the memory cost to calculate a subset of the Kth order affinity tensor for a mini-batch is $\mathcal{O}(m_b^K)$. Considering the same case above with $K = 3$ and choosing mini-batch size $m_b = 128$, the memory cost is about 0.016 GB if the affinity tensor of the mini-batch is stored in double-precision floating points, which is only 0.214% of the memory cost when storing the whole third order affinity tensor.

In training, TSC-Net, consisting of an embedding network and an orthogonalization layer, learns the mapping from the input samples to the tensor spectral embedding via the standard TSC objective proposed in [10], [11], [12]. In testing, the input samples are propagated through TSC-Net to obtain the tensor spectral embedding, followed by the k-means algorithm to yield the cluster labels. The network structure is illustrated in Fig. 1. Empirical evaluations on benchmark datasets demonstrate the effectiveness of the proposed method by comparison with recent baselines.

The main contributions of this paper are as follows.

1) A tensor spectral clustering network, TSC-Net, is proposed to learn to map the input samples to the spectral embedding via stochastic optimization, thereby reducing the memory cost without sacrificing the clustering performance.
2) The proposed TSC-Net can seamlessly integrate multiple affinity tensors in a joint framework. We instantiate TSC-Net to integrate the second, third, and fourth order affinity tensors to demonstrate the improved clustering performance over nine recent baselines.
3) Extensive experiments on benchmark datasets have been conducted to verify the effectiveness of the proposed method in comparison to current competitors in terms of clustering performance and memory cost.

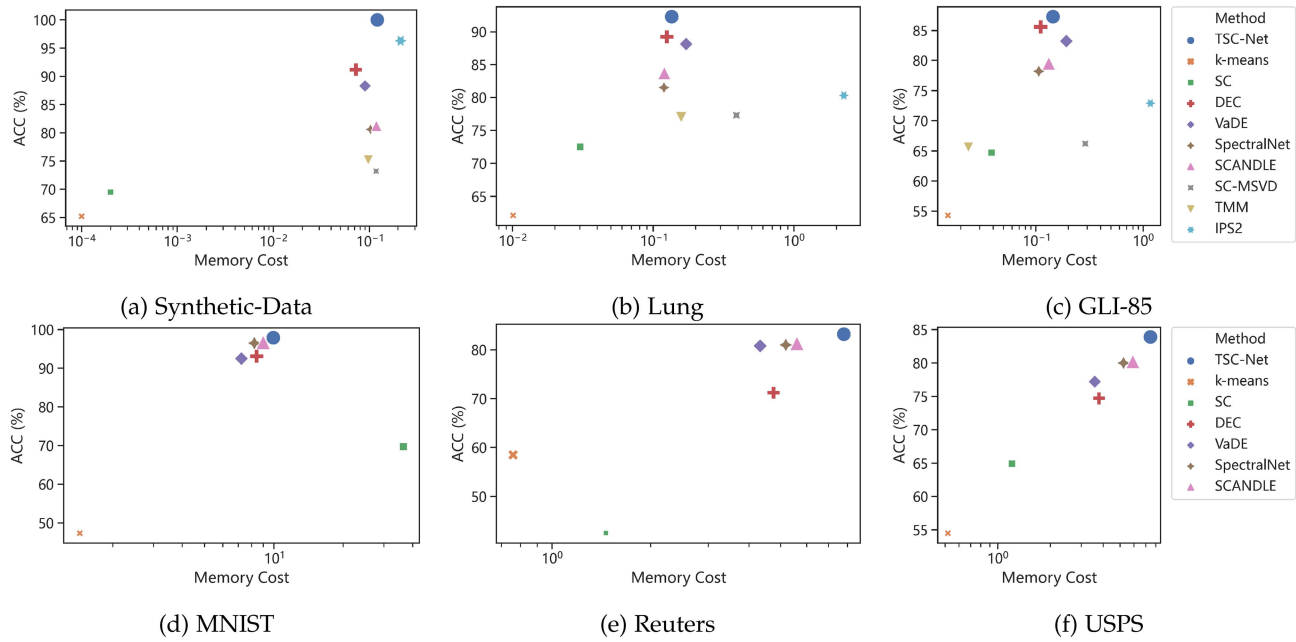


Fig. 2. ACC performance along with memory cost comparison of TSC-Net and its competitors on (a) Synthetic-Data, (b) Lung, (c) GLI-85, (d) MNIST, (e) Reuters, and (f) USPS. Note that SC-MSVD, TMM, and IPS2 are not able to run on MNIST, Reuters, and USPS due to out-of-memory issues.

The remainder of this article is structured as follows. We first make an overview of the related works in Section II. Next, Section III details our proposed TSC-Net. In Section IV, we conduct experiments to demonstrate the effectiveness of our proposed method. Finally, we draw a conclusion in Section V.

II. Related Works

A. Tensor Spectral Clustering

Suppose we have m samples with n features, denoted by a matrix $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_m] \in \mathbb{R}^{m\times n}$. The aim of clustering is to assign these samples into c disjoint clusters.

Spectral Clustering: The classic spectral clustering [3] starts by computing an affinity matrix or a second order affinity tensor $\boldsymbol{\mathcal{T}}^{(2)} \in \mathbb{R}^{m\times m}$, where every element $\boldsymbol{\mathcal{T}}^{(2)}_{ij}$ represents the pairwise similarity between the corresponding two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Subsequently, one can obtain the embedding $\boldsymbol{Y} \in \mathbb{R}^{m\times c}$ by a trace minimization problem

$$\min_{\boldsymbol{Y}\in\mathbb{R}^{m\times c}} -\mathrm{Tr}\left(\boldsymbol{Y}^\top \tilde{\boldsymbol{\mathcal{T}}}^{(2)} \boldsymbol{Y}\right)$$
$$s.t. \quad \boldsymbol{Y}^\top \boldsymbol{Y} = \boldsymbol{I}_c, \tag{1}$$

where $\tilde{\boldsymbol{\mathcal{T}}}^{(2)} = \boldsymbol{D}^{-1/2}\boldsymbol{\mathcal{T}}^{(2)}\boldsymbol{D}^{-1/2}$ is the normalized affinity matrix and $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{\mathcal{T}}^{(2)}_{ij}$ is the corresponding diagonal degree matrix.

The performance of spectral clustering pins on the pairwise similarity in the affinity matrix. However, the pairwise similarities are easily broken by noise contamination [18] or concentration effect [19] in data with high dimensions. To address this issue, recent works of tensor spectral clustering [13] attempted to use high-order tensor affinity among more than two samples to compensate for the inefficacy within the pairwise similarities.

Tensor Spectral Clustering: The basic goal of tensor spectral clustering is to determine to which clusters these samples belong based on the multi-wise similarities encoded in an affinity tensor. It has recently been proposed to characterize complex multi-wise similarities in an affinity tensor to achieve a better performance than pairwise similarity based approaches [20], [21], [22], [23], [23], [24], [24], [25], [26], [27]. Specifically, several works applied a combination of Euclidean distances among samples to construct the multi-wise similarities encoded in the affinity tensor and then employed high-order SVD or tensor trace norm maximization to derive the tensor spectral embedding matrix for clustering. For instance, Ghoshdastidar et al. [11] proposed a multilinear SVD method to decompose the affinity tensor and showed that this decomposition amounted to clustering samples by maximizing the squared associativity of the partition. Ghoshdastidar et al. [10] applied a trace optimization on the affinity tensor and developed a tensor sampling strategy [12] to save the computational cost.

Specifically, the K-wise similarities among m samples can be characterized by a Kth order affinity tensor $\boldsymbol{\mathcal{T}}^{(K)} \in \mathbb{R}^{\overbrace{m\times m\cdots\times m}^{K}}$, in which an element of $\boldsymbol{\mathcal{T}}^{(K)}$ represents the similarity of K corresponding samples. For instance, one can use scaled Gaussian distance to compute the second order affinity tensor, i.e., the affinity matrix, defined by

$$\boldsymbol{\mathcal{T}}^{(2)}_{i,j} = \exp\left(d_{ij}/\sigma^2\right), \forall i,j \in [m].$$

For the third order tensor affinity $\boldsymbol{\mathcal{T}}^{(3)}$, one can use anchor-based distance, defined by

$$\boldsymbol{\mathcal{T}}^{(3)}_{i,j,k} = \exp\left(-\max\left\{d_{ij}, d_{ik}, d_{jk}\right\}\right), \forall i,j,k \in [m].$$

Here, the metric d_{ij} is computed by $||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2$. Here, the scale σ is set as the median distance between a point to its third neighbor [15]. The fourth order affinity tensor $\boldsymbol{\mathcal{T}}^{(4)}$ is defined as in [13], which uses the ratio-based pair-to-pair similarities. The Fisher-ratio-like fourth-order tensor affinity among four samples is defined by

$$\boldsymbol{\mathcal{T}}^{(4)}_{i,j,k,l} = \exp\left(-\frac{d_{ij} + d_{kl} + d_{il} + d_{jk}}{d_{ik} + d_{jl}}\right), \forall i, j, k, l \in [m].$$

where d_{ij} denotes the distance between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

The tensor spectral clustering methods seek to find a consensus low dimensional embedding $\boldsymbol{Y}$ through minimizing a total associativity [10] between the embedding $\boldsymbol{Y}$ with the ensemble of second, third, and fourth orders as follows,

$$\begin{aligned}\min_{\boldsymbol{Y}} \; & -\sum_{s=1}^{c}\Big\{\boldsymbol{\mathcal{T}}^{(2)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \\ & + \alpha \boldsymbol{\mathcal{T}}^{(3)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \times_3 \boldsymbol{Y}_{:,s}^\top \\ & + \beta \boldsymbol{\mathcal{T}}^{(4)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \times_3 \boldsymbol{Y}_{:,s}^\top \times_4 \boldsymbol{Y}_{:,s}^\top\Big\} \\ s.t. \; & \boldsymbol{Y}^\top \boldsymbol{Y} = \boldsymbol{I}_c. \end{aligned} \tag{2}$$

where α and β are two hyperparameters for balancing the affinity tensors of various orders and c stands for the number of clusters. The operator $\times_k$ between a tensor $\boldsymbol{\mathcal{A}}$ and a vector $\boldsymbol{U}$ is called mode-k product and is defined as,

Definition 2.1. (*Mode-k Product*): Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m_1 \times m_2 \cdots \times m_K}$ be a Kth order tensor and $\boldsymbol{U} \in \mathbb{R}^{q \times m_k}$ be a matrix. The mode-k product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{U}$ is a Kth order tensor denoted by $\boldsymbol{\mathcal{A}} \times_k \boldsymbol{U} \in \mathbb{R}^{m_1 \times \cdots \times m_{k-1} \times q \times m_{k+1} \cdots \times m_K}$ such that

$$\begin{aligned}&(\boldsymbol{\mathcal{A}} \times_k \boldsymbol{U})_{r_1,\ldots,r_{k-1},t,r_{k+1},\ldots,r_K} \\ &= \sum_{r_k=1}^{m_k} \boldsymbol{\mathcal{A}}_{r_1,\ldots,r_{k-1},r_k,r_{k+1},\ldots,r_K} \boldsymbol{U}_{t,r_k}. \end{aligned} \tag{3}$$

Similar to the conventional spectral embedding, each row of the consensus embedding $\boldsymbol{Y}$ represents a sample in the embedding space, and the final cluster labels can be derived by performing a subsequent k-means algorithm on $\boldsymbol{Y}$.

The tensor spectral clustering methods have achieved superior performance when dealing with high-dimension low-sample-size (HDLSS) data [28], [29], [30]. However, the tensor spectral clustering methods share a common limitation, i.e., they need to construct and store the whole affinity tensor before deriving the tensor spectral embedding. The expensive memory costs to store the whole affinity tensor make the existing TSC methods less applicable for real-world tasks.

B. Deep Clustering

Deep clustering [31], [32] aims to integrate clustering and deep neural networks into a unified framework. Most deep clustering methods incorporate deep auto-encoder to extract features from complicated high-dimensional data to facilitate clustering, where there is a mutual boosting between clustering and deep feature representation. For example, DEC [33] attempt to achieve learning cluster centers and deep discriminating feature simultaneously, supervised by a clustering objective. In [34], deep self-evolution clustering (DSEC) has been proposed to train the network alternately with chosen pairs of samples. In [35], DCCM has been proposed to use pseudo-labels by a self-supervision scheme to guide clustering and to minimize mutual information to prune discriminative representations. A partition confidence maximization (PICA) [36] has been proposed to minimize a cluster partition uncertainty index, thereby learning the most confident clustering allocation. Recently, a group of deep clustering introduced self-expression to learn an affinity matrix with deep auto-encoders. One representative work, a deep subspace clustering network (DSC-Net), has been proposed in [37], which incorporated a self-expression module with a deep auto-encoder. The above deep clustering methods seek to mine discriminating features via exploiting deep neural networks. Alternatively, several recent works leverage deep neural networks to reduce the intensive computational cost of traditional clustering methods like spectral clustering. For instance, SpectralNet [15] learns a nonlinear mapping that can embed input data points into the spectral embedding, and demonstrate an effective memory cost reduction. SCANDLE [31] utilizes an adaptive neighbors technique to achieve spectral embedding with adaptively estimated affinities.

TABLE I
DEFINITION OF NOTATIONS

Notation	Remark
$\boldsymbol{X}$	Input data matrix
$\mathcal{F}$	Tensor Spectral Clustering Network
$\boldsymbol{\mathcal{T}}^{(K)}$	K-th order affinity tensor
$\widetilde{\boldsymbol{Y}}$	Intermediate representation
$\boldsymbol{Y}$	Tensor spectral embedding matrix
$\boldsymbol{Y}_{:,s}$	s-th column of $\boldsymbol{Y}$
$\boldsymbol{I}_c$	Identity matrix of size $\mathbb{R}^{c \times c}$
$\boldsymbol{\mathcal{A}} \times_k \boldsymbol{U}$	Mode-k product of tensor $\boldsymbol{\mathcal{A}}$ and matrix $\boldsymbol{U}$
$\mathcal{X}/\boldsymbol{x}_i$	Dataset / i-th sample
$\{\boldsymbol{W}_q\}_{q=1}^{Q}$	Weight matrices of Q fully-connected layers
$\boldsymbol{W}_{\text{ort}}$	Weight matrix in orthogonalization layer
$m/c/n$	Number of samples / clusters / features
m_b	Mini-batch size
$[m]$	Set of $\{1, 2, \cdots, m\}$

III. METHOD

This section details our proposed tensor spectral clustering network (TSC-Net). For ease of presentation, the main notations used in this paper are summarized in Table I. The network structure and objective function of TSC-Net are detailed in Section III-A, and an alternating training algorithm to optimize TSC-Net is shown in Section III-B. Next, the advantage of TSC-Net compared with the existing TSC methods in terms of memory cost is discussed in Section III-C.

A. Tensor Spectral Clustering Network

The major problem of existing TSC methods is that they need to construct and store the whole Kth order affinity tensor

before conducting tensor spectral embedding, which is memory-intensive and thus less applicable. To address the memory-intensive issue, we propose a tensor spectral clustering network (TSC-Net) that maps the input samples $\boldsymbol{X}$ to the tensor spectral embedding $\boldsymbol{Y}$ and trains it with stochastic optimization. The intuition behind TSC-Net is that the tensor spectral embedding $\boldsymbol{Y}$ can be regarded as a nonlinear mapping from an original sample space, $\boldsymbol{X}$, to the embedding space, and a modern neural network with nonlinear activation functions is able to approximate such a nonlinear mapping [15], [16], [17]. Accordingly, the proposed TSC-Net is allowed to approximate the tensor spectral embedding and can be trained in a stochastic manner to avoid storing the whole Kth order affinity tensor. As a result, TSC-Net enables a significant reduction of memory cost.

Formally, TSC-Net is defined as a neural network $\mathcal{F}: \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times c}$ that maps the m input samples to their corresponding tensor spectral embedding with possible parameters of neural networks introduced in the next. More specifically, the neural network maps the data matrix $\boldsymbol{X}$ to the tensor spectral embedding as $\boldsymbol{Y} = \mathcal{F}(\boldsymbol{X})$, where the ith row of $\boldsymbol{Y}$ denotes the tensor spectral embedding for the sample $\boldsymbol{x}_i$ in the data matrix.

Network Architecture: To achieve such a mapping while considering the orthogonal constraint for $\boldsymbol{Y}$, the neural network $\mathcal{F}$ is designed as follows. As shown in Fig. 1, the neural network $\mathcal{F}$ have the collection of learnable parameters $\{\{\boldsymbol{W}_q\}_{q=1}^{Q}, \boldsymbol{W}_{\text{ort}}\}$ and can be separated into two major parts:

1) an embedding network that maps the data matrix $\boldsymbol{X}$ to the intermediate representation $\widetilde{\boldsymbol{Y}}$. To be more specific, the embedding network with Q layers performs the layer-wise mapping as $\boldsymbol{H}^q = g(\boldsymbol{H}^{q-1}\boldsymbol{W}_q)$, for $q \in [Q]$, where $\boldsymbol{H}^0$ corresponds the input data matrix $\boldsymbol{X}$, $\boldsymbol{H}^Q$ corresponds to the intermediate representation $\widetilde{\boldsymbol{Y}}$, $\boldsymbol{W}_q$ denotes the weight matrix for the qth fully-connected layer, and g denotes the ReLU activation function
2) an orthogonalization layer with weight matrix $\boldsymbol{W}_{\text{ort}} \in \mathbb{R}^{c\times c}$ that maps the intermediate representation $\widetilde{\boldsymbol{Y}}$ to $\boldsymbol{Y} = \widetilde{\boldsymbol{Y}}\boldsymbol{W}_{\text{ort}}$ to ensure the orthogonal constraint, i.e., $\boldsymbol{Y}^\top\boldsymbol{Y} = \boldsymbol{I}_c$. The elaboration of the orthogonalization layer will be presented in Section III-B.

TSC-Net achieves the tensor spectral embedding by minimizing the following objective:

$$\begin{aligned}\min_{\{\boldsymbol{W}_q\}_{q=1}^{Q},\boldsymbol{W}_{\text{ort}}} \mathcal{L}_{\text{obj}} = -\sum_{s=1}^{c}\Big\{&\boldsymbol{\mathcal{T}}^{(2)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \\ &+ \alpha\boldsymbol{\mathcal{T}}^{(3)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \times_3 \boldsymbol{Y}_{:,s}^\top \\ &+\beta\boldsymbol{\mathcal{T}}^{(4)} \times_1 \boldsymbol{Y}_{:,s}^\top \times_2 \boldsymbol{Y}_{:,s}^\top \times_3 \boldsymbol{Y}_{:,s}^\top \times_4 \boldsymbol{Y}_{:,s}^\top\Big\}.\end{aligned} \tag{4}$$

The orthogonalization layer with $\boldsymbol{W}_{\text{ort}}$ is implemented to ensure $\boldsymbol{Y}^\top\boldsymbol{Y} = \boldsymbol{I}_c$. Different from the original TSC model, which is solved by a memory-intensive tensor decomposition [10], [11], [12], the proposed TSC-Net can be trained with a stochastic optimization under (4) with less memory requirement. This will be concretely illustrated in Section III-B.

B. Alternating Stochastic Optimization Algorithm

One major advantage of the proposed TSC-Net is that it allows us to optimize the tensor spectral embedding in a stochastic manner and merely needs to construct a small part of the affinity tensor, thus reducing the memory cost. To optimize TSC-Net with (4), we propose an alternating stochastic optimization algorithm, which is an iterative scheme and alternates between the embedding stage and the orthogonalization stage. More concretely, let us assume the algorithm runs Ω epochs. In each epoch, we randomly sample $T = m/m_b$ iterations to ensure walking through the entire data matrix, where m and m_b denote the whole data size and mini-batch size, respectively, and T is assumed to be an integer for simplicity. In each iteration, m_b samples are randomly sampled. In what follows, suppose we are in an arbitrary possible epoch $\omega \in [\Omega]$, and we omit the notation about ω for simplicity. For a possible iteration $t \in [T]$, the index $I_t \subseteq \{1, 2, \ldots, m\}$ of the randomly sampled samples of size $|I_t| = m_b$, and the mini-batch $\boldsymbol{X}_{I_t}$, the algorithm iteratively alternates between:

Orthogonalization stage:

1) Forward passing to obtain the intermediate representation $\widetilde{\boldsymbol{Y}}_{I_t}^{(t)}$
2) Updating orthogonalization layer

$$\boldsymbol{W}_{\text{ort}}^{(t)} = \left(\boldsymbol{L}^{(t)-1}\right)^\top, \tag{5}$$

where $\boldsymbol{L}^{(t)} \in \mathbb{R}^{c\times c}$ is a lower triangular matrix derived by the Cholesky decomposition as $\widetilde{\boldsymbol{Y}}_{I_t}^{(t)\top}\widetilde{\boldsymbol{Y}}_{I_t}^{(t)} = \boldsymbol{L}^{(t)}\boldsymbol{L}^{(t)\top}$. As such, the mini-batch of the output of TSC-Net, $\boldsymbol{Y}_{I_t}^{(t)} = \widetilde{\boldsymbol{Y}}_{I_t}^{(t)}\boldsymbol{W}_{\text{ort}}^{(t)}$, can meet the orthogonal constraint and can be verified by the Theorem 3.1.

In practice, the full rankness of $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}}$ can be ensured by adding sufficiently small numbers, e.g., 10^{-5}, to the diagonal elements. It is noteworthy that the orthogonal constraint works for a mini-batch of samples to prevent trivial solutions, and the orthogonal constraint across mini-batches during stochastic optimization is not required.

3) Continuing forward passing: $\boldsymbol{Y}_{I_t}^{(t)} = \widetilde{\boldsymbol{Y}}_{I_t}^{(t)}\boldsymbol{W}_{\text{ort}}^{(t)}$.

Theorem 3.1. Given a matrix $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{m\times c}$ and suppose $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}}$ is full rank, a lower triangular matrix $\boldsymbol{L} \in \mathbb{R}^{c\times c}$ is derived by the Cholesky decomposition of $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}}$ as $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}} = \boldsymbol{L}\boldsymbol{L}^\top$, then $\boldsymbol{Y} = \widetilde{\boldsymbol{Y}}(\boldsymbol{L}^{-1})^\top$ satisfies the orthogonal constraint.

Proof. If $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}}$ is full rank with the Cholesky decomposition $\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}} = \boldsymbol{L}\boldsymbol{L}^\top$ and $\boldsymbol{Y} = \widetilde{\boldsymbol{Y}}(\boldsymbol{L}^{-1})^\top$, one can check that $\boldsymbol{Y}^\top\boldsymbol{Y} = \boldsymbol{L}^{-1}\widetilde{\boldsymbol{Y}}^\top\widetilde{\boldsymbol{Y}}(\boldsymbol{L}^{-1})^\top = \boldsymbol{L}^{-1}\boldsymbol{L}\boldsymbol{L}^\top(\boldsymbol{L}^{-1})^\top = \boldsymbol{I}_c$. □

Embedding Stage: By fixing $\boldsymbol{W}_{\text{ort}}^{(t)}$, one then uses the stochastic gradient descent to update the embedding network parameters including $\{\boldsymbol{W}_q\}_{q=1}^{Q}$

$$\boldsymbol{W}_q^{(t+1)} = \boldsymbol{W}_q^{(t)} - \eta\frac{\partial\mathcal{L}_{\text{obj}}}{\partial\boldsymbol{W}_q^{(t)}}. \tag{6}$$

Algorithm 1: Training and Testing of **TSC-Net**.

Input:
Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_m]$;
Mini-batch size m_b;
The hyperparameters α.
Output:
Trained TSC-Net $\mathcal{F}$;
Clustering labels;
Training
1: Randomly initializing the collection of network parameters $\{\{\boldsymbol{W}_q\}_{q=1}^{Q}, \boldsymbol{W}_{\text{ort}}\}$;
2: **while** *not reaching convergence criterion* **do**
3: Randomly sampling a mini-batch of m_b samples;
4: Constructing the affinity tensors $\boldsymbol{\mathcal{T}}^{(2)}$, $\boldsymbol{\mathcal{T}}^{(3)}$, and $\boldsymbol{\mathcal{T}}^{(4)}$ for the m_b samples;
Orthogonalization Stage:
5: Updating $\boldsymbol{W}_{\text{ort}}$ by (5);
Embedding Stage:
6: Updating $\{\boldsymbol{W}_q\}_{q=1}^{Q}$ by (6);
7: **end while**
Testing
8: Forward passing all the samples through $\mathcal{F}$ to obtain the tensor spectral embedding $\boldsymbol{Y}$;
9: Performing k-means on $\boldsymbol{Y}$ to obtain the clustering labels.

where η denotes the learning rate, and the detailed derivation of $\frac{\partial \mathcal{L}_{\text{obj}}}{\partial \boldsymbol{W}_q^{(t)}}$ is provided in Appendix A, available online.[1] One can check that only m_b samples are required in such an updating with $\mathcal{O}(m_b^2 + m_b^3 + m_b^4)$ memory cost for the affinity tensors.

When finishing an epoch, we reshuffle the data matrix before continuing to the next epoch. The convergence criterion is set as either reaching the maximum epoch $\Omega = 300$ or the relative changes of the objective (4) during successive epochs in training is less than a predefined threshold $\epsilon = 10^{-3}$. Once TSC-Net is trained, all the parameters, $\{\{\boldsymbol{W}_q\}_{q=1}^{Q}, \boldsymbol{W}_{\text{ort}}\}$, are freezed. In the testing, the whole samples are propagated through TSC-Net to obtain the tensor spectral embedding $\boldsymbol{Y}$, and the k-means algorithm is employed on it to obtain the cluster labels. These algorithmic procedures are summarized below in Algorithm 1.

C. Memory Cost Comparison of TSC-Net and Existing Methods

This subsection centers on how much the proposed TSC-Net or TSC-Net can reduce the memory cost in comparison to the existing TSC methods. The stochastic optimization introduced in the previous section allows TSC-Net or TSC-Net to calculate a small part of the affinity tensors at each time and avoid constructing the whole affinity tensor, while most of the existing TSC methods use tensor decomposition, e.g., [10], [11], [12], and require loading the whole affinity tensor. Hence, in contrast to the conventional TSC, our proposed TSC-Net and TSC-Net enables a reduction of memory cost.

It is difficult to conduct an exact memory cost comparison between TSC-Net and the existing TSC methods since TSC-Net involves not only constructing the affinity tensors but also training the neural network parameters. It is informative to compare IPS2 [13] and TSC-Net in terms of memory cost on m samples. IPS2 needs to store a second order affinity tensor and a fourth order affinity tensor with $\mathcal{O}(m^2 + m^4)$ memory cost. By assuming the batch size is m_b, TSC-Net only needs $\mathcal{O}(m_b^2 + m_b^3 + m_b^4)$ memory cost for the affinity tensors at each time due to the stochastic gradient descent. Taking $m = 1000$ and $m_b = 128$ as an example, the actual affinity tensor cost of IPS2 and TSC-Net are 931.320 GB and 0.252 GB, respectively, if the corresponding affinity tensors are uniformly stored in double-precision floating points. Additionally, according to the recent literature [38], the stochastic optimization of training the neural network parameters usually costs less than 1 GB of memory, so the total memory cost of TSC-Net is less than 0.252 GB (affinity tensors) + 1 GB (training neural network parameters) = 1.252 GB. In a nutshell, TSC-Net is much more memory-efficient compared with IPS2 as the sample size increases. In Section IV, we will experimentally demonstrate the memory cost of TSC-Net on several benchmark datasets with different sample sizes ranging from 50 to 70,000.

IV. Experiments and Results

In this section, the proposed TSC-Net is evaluated on benchmark datasets in comparison with several baseline methods. Specifically, the proposed TSC-Net is assessed in terms of memory cost and performance improvement, especially compared with state-of-the-art TSC methods. Afterward, ablation studies are conducted to verify the effectiveness of the ensemble of multiple affinity tensors. Finally, the hyperparameter analysis and convergence behavior analysis are presented. Due to space limitations, the noise-robustness testament of TSC-Net-234 and its competitors are provided in Appendix B, available online.[2]

A. Experiment Setup

Software Environment: All our experiments were performed on a desktop computer with a 3.70 GHz Intel(R) Core(TM) i7-8700 K CPU, 64.0 GB of RAM, and GTX 1080 Ti 10 G. The code is based on Python 3.6, TensorFlow 1.15.

Evaluation Metrics: Following the convention in the clustering literature [3], [11], we employed three metrics for performance comparison throughout all experiments: Accuracy (**ACC**), Normalized Mutual Information (**NMI**), and Purity (**Purity**). For all three metrics, higher numerical values consistently indicate better clustering performance.

Benchmark Datasets: In the experiments, six benchmark datasets were adopted to evaluate the performance of the proposed method and the compared methods, and the corresponding statistics are shown in Table II. Particularly, Synthetic-Data

[1]https://github.com/Huyu2Jason/Publication_Appendix/blob/main/TPAMI-2023-01-0162-appendix.pdf

[2]https://github.com/Huyu2Jason/Publication_Appendix/blob/main/TPAMI-2023-01-0162-appendix.pdf

TABLE II
STATISTICS ON SIX BENCHMARK DATASETS

Dataset	# Sample	# Feature	# Cluster
GLI-85	85	22283	2
Reuters	10000	2000	4
Lung	100	12600	5
MNIST	70000	784	10
Synthetic-Data	50	500	3
USPS	9298	256	10

adopted from [13] is composed of 50 samples from three clusters, drawn from i.i.d Normal distributions with an equal standard deviation of 0.5 and a different mean of 1, 2.5, and 5, with each sample having 500 features. Lung [39] and GLI-85 [40] are two typical bioinformatics datasets of high dimensionality. MNIST [41] and USPS[3] are two typical image datasets, while Reuters [42] is a popular documentation dataset.

Compared Methods: As baselines to our method, the competitors fall into three categories. (1) traditional clustering including: [3] (SC, NIPS-2002) and k-means. (2) Tensor spectral clustering including Spectral Clustering Using Multilinear SVD [11] (SC-MSVD, AAAI-2015), Uniform hypergraph partitioning: Provable tensor methods and sampling techniques [12] (TMM, JMLR-2017), Integrating tensor similarity and pairwise similarity [13] (IPS2, TPAMI-2022). (3) Deep clustering including Variational deep embedding: an unsupervised and generative approach to clustering (VaDE, IJCAI-2017) [43], Unsupervised deep embedding for clustering analysis (DEC, ICML-2016) [33], Spectral clustering using deep neural networks (SpectralNet, ICLR-2018) [15], Spectral Clustering With Adaptive Neighbors for Deep Learning (SCANDLE, TNNLS-2021) [31].

Implementation Details: With regard to the architecture of TSC-Net, the fully connected layers similar to [15] were employed consistently. Specifically, the embedding network involves the first two fully connected layers with both 1024 neurons followed by the ReLU activation function, the third fully connected layer with 512 neurons followed by the ReLU activation function, and the last classification layer with c neurons followed by the Tanh activation function. By contrast, the orthogonalization layer is automatically inferred by Theorem 3.1. TSC-Net is trained by Adam optimizer with the learning rate 0.001 and the maximum epoch $\Omega = 300$, where the mini-batch size is set as 128 if the sample size m is larger than 128 or as $m/5$ otherwise. The grid search method was used to find the task-related hyperparameters for α and β, both from the set [0.001, 0.01, 0.1, 1, 10, 100]. The sensitivity analysis is shown in Section IV-E. Regarding the compared methods, we adopted their default settings, as stated in the original articles.

In addition, the above datasets were all pre-processed by an auto-encoder, in line with the experimental setting as in [15], [31]. Specifically, the auto-encoder from [43] was employed to extract the deep feature representation of each dataset. Subsequently, all the experiments were conducted based on the deep feature representation instead of the original feature. For a fair comparison, we ran each method 50 times on each dataset and calculated the mean values of the corresponding metrics.

[3][Online]. Available: https://www.kaggle.com/bistaumanga/usps-dataset

B. Clustering Performance Comparison on Benchmark Datasets

1) Comparison With Existing TSC Methods: Table III presents the comparison of the proposed TSC-Net with state-of-the-art TSC methods, including IPS2. TSC-Net consistently outperforms all the TSC methods on all the metrics on the benchmark datasets, if other TSC methods are available to perform experiments. Specifically, TSC-Net is superior to IPS2 by 3.7%, 13.4%, and 16.0% in terms of ACC on Synthetic-Data, Lung, and GLI-85, respectively. The IPS2 is quite relevant to TSC-Net since it leverages the second and fourth order affinity tensors to obtain the consensus tensor spectral embedding. However, it performs the affinity ensemble to obtain the consensus embedding in a two-stage manner. By contrast, TSC-Net allows multiple affinity tensor to integrate in a one-stage manner, enhances and quality of the learned embedding, and thus increase the clustering performance. Fig. 3(a)–(c) provides a visual explanation of the T-SNE results on learned spectral embedding for each method. In those figures, TSC-Net obtains a more discriminating cluster boundary than IPS2, which is consistent with better clustering performance.

2) Comparison With Deep Clustering Methods: Table III also presents the comparison of the proposed TSC-Net with recent deep clustering methods, including SpectralNet, SCANDAL, and DEC. TSC-Net uniformly outstrips all these deep clustering methods in all the metrics on the benchmark datasets. Particularly, TSC-Net achieves improvement in comparison to SpectralNet by 19.4%, 12.2%, 10.7%, 1.4%, 2.2%, and 3.9% in terms of ACC on Synthetic-Data, Lung, GLI-85, MNIST, Reuters, and UPSP, respectively. SpectralNet is most relevant to TSC-Net since SpectralNet leverages the second order affinity tensors to obtain the tensor spectral embedding based on a deep neural network. By contrast, TSC-Net leverages the ensemble of the second, third, and fourth order affinity tensors to achieve a better clustering performance. The performance gap between TSC-Net and SpectralNet illustrates that different affinity tensors can complement each other in terms of improving clustering performance.

C. Memory Cost Comparison on Benchmark Datasets

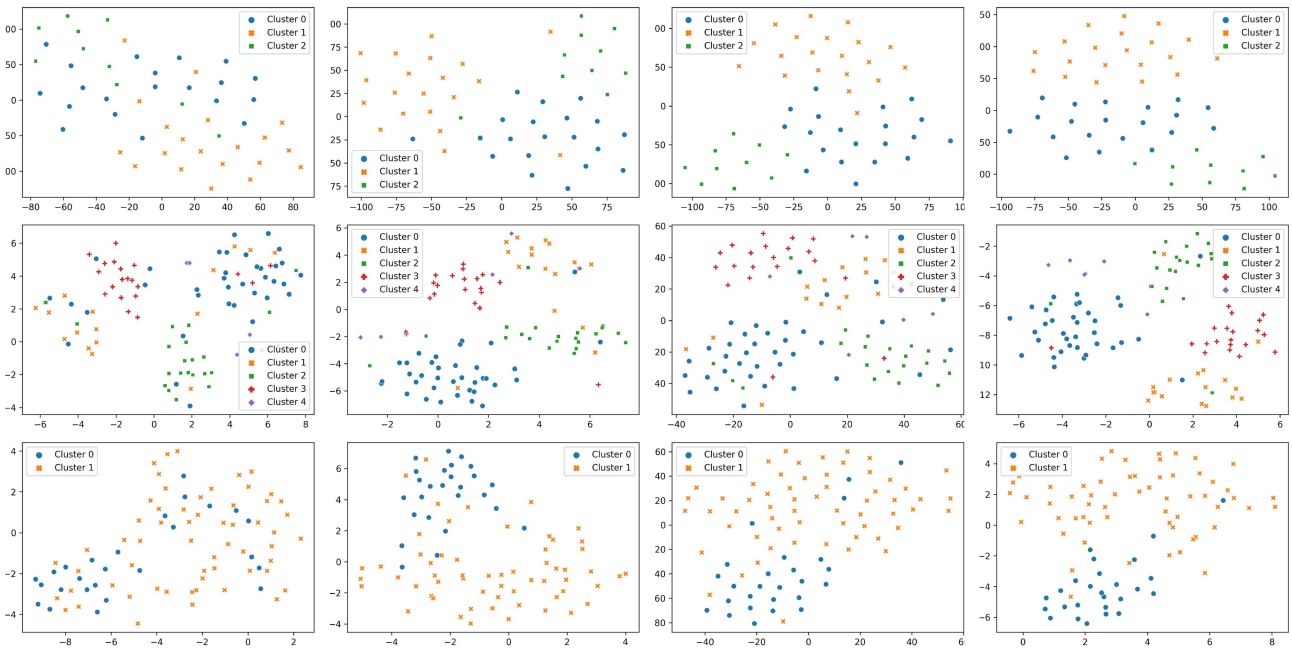
In this subsection, the memory cost comparison of TSC-Net and the existing TSC methods is demonstrated on the benchmark datasets. The memory costs of the existing TSC methods, SC, and k-means are evaluated by Matlab 2019a software since their public codes are all based on Matlab. Differently, the memory costs of TSC-Net and deep clustering methods are evaluated by the Nvidia-SMI software, as they are based on Nvidia GPU (GTX 1080 Ti).

Apart from the clustering performance shown in Table III, Fig. 2(a)–(f) demonstrates the clustering performance and

TABLE III
PERFORMANCE COMPARISON ON SYNTHETIC-DATA, LUNG, AND GLI-85 (MEAN AND STANDARD DEVIATION %)

Dataset	Synthetic-Data			Lung			GLI-85		
	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity
k-means	65.2 (0.5)	57.9 (0.4)	69.2 (0.6)	62.1 (0.3)	50.9 (0.5)	63.8 (0.3)	54.3 (0.3)	48.5 (0.7)	59.4 (0.5)
SC (NIPS-2002)	69.5 (0.7)	60.2 (0.5)	70.9 (0.4)	72.5 (0.3)	65.9 (0.7)	74.2 (0.5)	64.7 (0.5)	57.4 (0.4)	70.0 (0.3)
SC-MSVD (AAAI-2015)	73.2 (0.7)	66.8 (0.5)	78.2 (0.8)	77.3 (0.4)	68.2 (0.5)	79.5 (0.4)	66.2 (0.2)	59.8 (0.2)	72.7 (0.3)
TMM (JMLR-2017)	75.2 (1.0)	68.9 (0.5)	80.4 (1.0)	77.0 (0.3)	67.9 (0.4)	79.1 (0.5)	65.6 (0.3)	58.4 (0.4)	70.9 (0.2)
IPS2 (TPAMI-2022)	96.3 (1.2)	92.1 (1.0)	96.8 (0.9)	80.3 (0.4)	69.5 (0.3)	82.3 (0.4)	72.9 (0.3)	64.7 (0.5)	77.2 (0.4)
DEC (ICML-2016)	91.2 (0.9)	82.3 (1.0)	92.3 (0.8)	89.2 (0.6)	78.7 (0.7)	91.2 (0.4)	85.6 (0.6)	74.8 (0.4)	88.9 (0.6)
VaDE (IJCAI-2017)	88.3 (1.2)	78.1 (1.0)	89.9 (1.1)	88.1 (0.2)	76.9 (0.5)	90.0 (0.5)	83.2 (0.4)	71.9 (0.8)	87.2 (0.7)
SpectralNet (ICLR-2018)	80.6 (1.2)	69.0 (0.8)	82.7 (0.9)	81.5 (0.5)	75.4 (0.5)	84.1 (0.3)	78.2 (0.3)	67.2 (0.5)	81.2 (0.5)
SCANDLE (TNNLS-2021)	81.2 (1.0)	69.9 (0.7)	83.4 (1.0)	83.7 (0.3)	77.9 (0.4)	86.4 (0.2)	79.5 (0.3)	68.0 (0.5)	82.9 (0.3)
TSC-Net (Ours)	**100.0 (0.0)**	**100.0 (0.0)**	**100.0 (0.0)**	**93.7 (0.7)**	**82.2 (0.4)**	**96.2 (0.2)**	**88.9 (0.6)**	**78.5 (0.5)**	**91.8 (0.2)**
Dataset	MNIST			Reuters			USPS		
	ACC	NMI	Purity	ACC	NMI	Purity	ACC	NMI	Purity
k-means	47.3 (0.7)	42.1 (0.5)	51.5 (0.4)	58.5 (0.7)	47.2 (0.9)	61.3 (0.7)	54.5 (0.7)	57.0 (0.8)	60.1 (0.3)
SC (NIPS-2002)	69.7 (0.4)	65.6 (0.2)	71.6 (0.3)	42.5 (0.5)	19.7 (0.3)	45.1 (0.3)	64.9 (0.6)	79.9 (0.8)	71.2 (0.7)
SC-MSVD (AAAI-2015)	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
TMM (JMLR-2017)	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
IPS2 (TPAMI-2022)	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**
DEC (ICML-2016)	93.1 (1.2)	85.1 (1.0)	94.3 (0.8)	71.2 (1.0)	51.9 (1.1)	73.8 (0.6)	74.7 (1.0)	76.2.2 (0.8)	77.0 (0.7)
VaDE (IJCAI-2017)	92.5 (1.4)	84.1 (1.3)	93.5 (1.2)	80.8 (1.7)	53.4 (1.4)	82.1 (1.0)	77.2 (0.7)	80.1 (1.0)	79.7 (0.8)
SpectralNet (ICLR-2018)	96.5 (0.9)	92.2 (0.8)	97.8 (1.0)	81.0 (0.8)	57.3 (0.6)	83.4 (1.0)	80.0 (0.5)	84.1 (0.6)	82.8 (0.7)
SCANDLE (TNNLS-2021)	96.7 (1.1)	92.3 (0.7)	97.9 (0.7)	81.3 (1.0)	57.6 (0.7)	83.9 (1.0)	80.2 (0.4)	84.7 (0.5)	83.9 (0.5)
TSC-Net (Ours)	**97.9 (0.5)**	**93.8 (0.3)**	**98.6 (0.8)**	**83.2 (0.6)**	**61.3 (0.5)**	**85.7 (0.8)**	**83.9 (0.5)**	**87.2 (0.7)**	**85.5 (1.0)**

The best results are boldfaced and the second-best ones are underlined.



(a) SC (b) DEC (c) IPS2 (d) TSC-Net

Fig. 3. From the top to the bottom, the T-SNE visualization of (a) SC; (b) IPS2, (c) DEC, and (d) TSC-Net on Synthetic-Data, Lung, and GLI-85, to demonstrate the quality of the embedding learned by each clustering method. As can be seen, TSC-Net can learn a more separable embedding with more clear class boundaries.

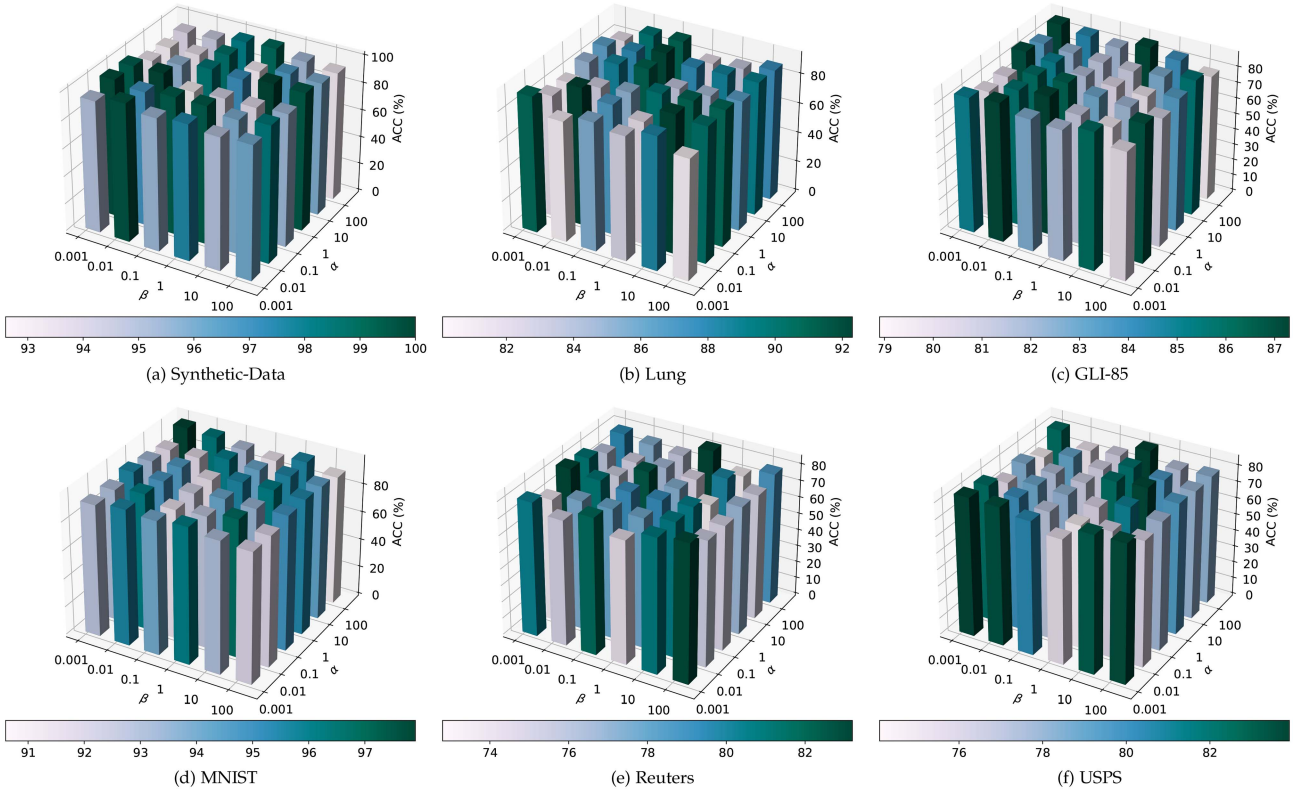


Fig. 4. The ACC performance of our proposed TSC-Net with respect to the varying hyperparameter α and β on (a) Synthetic-Data (b) Lung (c) GLI-85 (d) MNIST (e) Reuters (f) USPS.

TABLE IV
ABLATION STUDIES ON THE ENSEMBLE OF MULTIPLE AFFINITY TENSOR IN TSC-NET

$\mathcal{T}^{(2)}$	$\mathcal{T}^{(3)}$	$\mathcal{T}^{(4)}$	Synthetic-Data	Lung	GLI-85	MNIST	Reuters	USPS
✓	×	×	80.6 (1.2)	81.5 (0.5)	78.2 (0.3)	96.5 (0.9)	81.0 (0.8)	80.0 (0.5)
×	✓	×	81.1 (0.6)	86.7 (0.3)	79.7 (0.2)	96.9 (0.6)	81.2 (0.6)	81.5 (0.3)
×	×	✓	89.2 (0.4)	82.9 (1.0)	79.1 (0.4)	92.6 (0.8)	80.1 (0.6)	80.3 (0.4)
✓	✓	×	81.3 (0.3)	88.4 (0.4)	81.8 (0.3)	97.0 (0.4)	82.5 (0.5)	81.7 (0.3)
✓	×	✓	97.4 (0.5)	84.5 (0.5)	80.9 (0.2)	96.7 (0.5)	81.4 (0.5)	80.4 (0.2)
×	✓	✓	98.6 (0.3)	89.8 (0.3)	84.6 (0.4)	96.8 (0.4)	82.3 (0.3)	82.1 (0.4)
✓	✓	✓	**100.0 (0.0)**	**92.3 (0.7)**	**87.3 (0.6)**	**97.9 (0.5)**	**83.2 (0.6)**	**83.9 (0.5)**

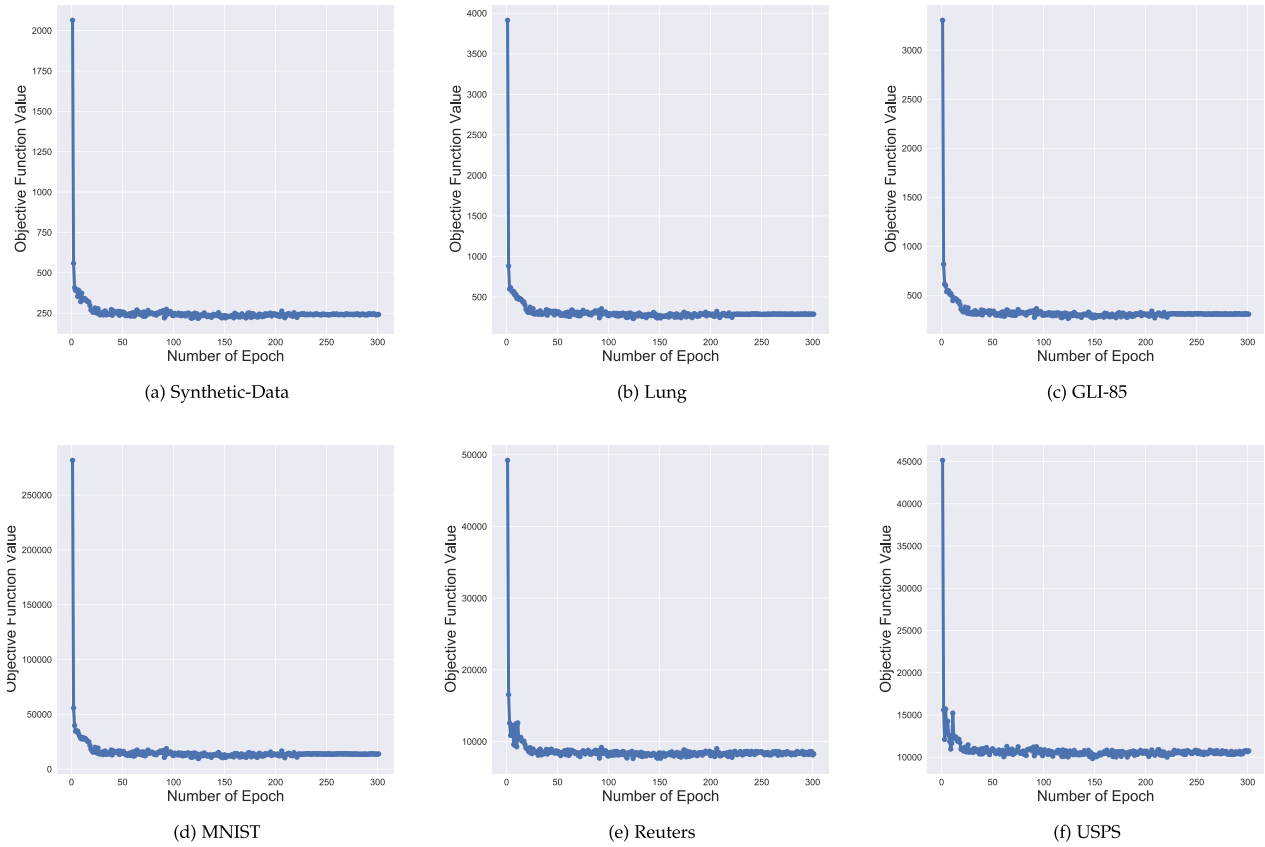
We report the ACC (mean and standard deviation %) of TSC-Net with different combinations of $\mathcal{T}^{(2)}$, $\mathcal{T}^{(3)}$, and $\mathcal{T}^{(4)}$ on benchmark datasets. The best results are boldfaced and the second-best ones are underlined.

memory cost comparison among TSC-Net and its competitors. Combining results from those tables, one can notice that TSC-Net achieves better clustering performance while its memory cost is relatively lower in comparison with other TSC methods like IPS2. Specifically, in Lung, whose sample size is 100, the memory cost of IPS2 is 2.2578 GB, whereas the one of TSC-Net is 0.1352 GB. Such a cost gap stems from the fact that TSC-Net adopts the batch-wise stochastic gradient for optimization without having to load the whole affinity tensors and saves the cost significantly. The results are in accordance with the memory cost analysis in Section III-C, indicating that our method achieves a better clustering performance while reducing the memory cost.

Notably, IPS2, TMM, and SC-MSVD fail to perform on large-scale datasets like MNIST (70,000 samples), Reuters (10,000 samples), and USPS (9298 samples) because the memory cost of constructing the affinity tensors is prohibitively high. In comparison, TSC-Net adopts the batch-wise stochastic gradient and only needs to construct and store a small part of affinity tensors on the basis of the batch size, which translates into a relatively low memory cost for large-scale datasets.

D. Ablation Study on Ensemble of Multiple Affinity Tensors

In Table IV, the ablation study of the ensemble of multiple affinity tensor results is illustrated to verify the contribution of each affinity tensor for clustering. Specifically, different combinations of affinity tensors are taken into consideration, where $\surd$ means being incorporated and $\times$ means being removed. From Table IV, one can notice that all the incorporated affinity tensors, i.e., the second, third, and fourth order ones, contribute



(a) Synthetic-Data (b) Lung (c) GLI-85

(d) MNIST (e) Reuters (f) USPS

Fig. 5. Objective function values of TSC-Net with the number of the epochs on six benchmark datasets.

to the clustering improvement. In particular, the ensemble of all three affinity tensors achieves the best clustering performance compared with other combinations that incorporate only part of the three. The performance improvements between TSC-Net and the second-best competitor are by 1.4%, 2.5%, 2.7%, 0.9%, 0.7%, and 2.2% in terms of ACC on Synthetic-Data, Lung, GLI-85, MNIST, Reuters, and UPSP, respectively. Also, one can find that the clustering performance by different affinity tensors is task-dependent. For instance, using solely fourth order affinity achieves better clustering performance than solely using third or second order affinity tensor on Synthetic-Data, Lung, and GLI-85, while solely using fourth order affinity achieve worse performance than solely using third order affinity tensor on MNIST and Reuters. This phenomenon indicates that different affinity tensors contribute unequally to clustering performance, depending on specific tasks.

E. Hyperparameter Analysis

TSC-Net has two hyperparameters, i.e., α and β, and their different numerical values of them imply different weight contributions of the corresponding affinity tensors. The previous section illustrates how combining various affinity tensors is task-dependent, and thus we consider using the grid search technique to find the task-optimal hyperparameters α and β both from the set [0.001, 0.01, 0.1, 1, 10, 100]. We show the hyperparameter sensitivity in terms of ACC on Synthetic-Data, Lung, GLI-85, MNIST, Reuters, and UPSP in Fig. 4. From the figures, we find that TSC-Net comes with a relatively stable ACC performance under a wide range of hyperparameters across all six datasets.

TABLE V
RUNNING TIME (IN SECONDS) OF TSC-NET ON THE SIX BENCHMARK DATASETS

Dataset	Running Time
Synthetic-Data	8.76
Lung	22.73
GLI-85	17.65
MNIST	25539.65
Reuters	20369.24
USPS	18695.34

F. Convergence and Running Time of TSC-Net

In this subsection, the convergence and running time of TSC-Net are investigated. We employed TSC-Net on the benchmark datasets, including Synthetic-Data, Lung, GLI-85, MNIST, Reuters, and UPSP, respectively. We then reported the objective function value of (2) with respect to the increasing epochs in Fig. 5 and the running time in seconds in Table V. Overall, the objective function values under the proposed alternative stochastic optimization monotonically decrease with epochs on

all the benchmark datasets, where at the first several epochs, the values remarkably decrease and then continuously decrease before the two hundred and fifty epoch. In summary, TSC-Net achieves a convergence quickly.

V. Conclusion

In this paper, we have proposed a tensor spectral clustering network for reducing the memory cost considerably and integrating multiple affinity tensors in a memory-efficient manner. Unlike existing methods, which uniformly need to load the whole affinity tensors, our method maps the input samples into the tensor spectral embedding with a neural network and allows for a batch-wise affinity tensor construction, which enables us to reduce the memory cost. More critically, compared with the previous method using a two-stage integration, our proposed method seamlessly ensemble multiple affinity tensors in a one-stage manner to improve the clustering performance while keeping a low memory cost. Experimental results have demonstrated that our method achieves considerable performance improvement while enjoying less memory cost on benchmark datasets.

Overall, the proposed method has demonstrated the effectiveness of high-dimensional data clustering. There are two potential directions for improvement:

1. Redundancy in Multiple Affinity Tensors: While the proposed method is capable of jointly ensembling multiple affinity tensors, one potential limitation lies in the redundancy that might arise in this ensemble process. The inclusion of multiple affinity tensors for similarity estimation may result in computational redundancy and increased computational costs. Specifically, the question of whether the additional complexity of incorporating multiple affinity tensors consistently improves the clustering performance needs to be addressed. Future research could explore strategies to efficiently select or weight the most informative affinity tensors to mitigate redundancy while preserving the benefits of ensemble learning.

2. Absence of Mutual Improvement between Clustering and Feature Learning: Another important aspect to consider is the reliance solely on the given affinity tensors for tensor spectral embedding. This implies that the tensor spectral clustering network does not actively update or adapt the affinity tensors during the learning process. The absence of mutual improvement between clustering and deep feature learning is a noteworthy limitation. Future research directions could explore mechanisms for dynamically updating the affinity tensors as part of the learning process. This would allow a network to adapt and refine the affinity information, potentially leading to improved clustering results.

References

[1] X. Peng, J. Feng, J. T. Zhou, Y. Lei, and S. Yan, "Deep subspace clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 12, pp. 5509–5521, Dec. 2020.

[2] Y. Zhang, X. Hu, and X. Jiang, "Multi-view clustering of microbiome samples by robust similarity network fusion and spectral clustering," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 2, pp. 264–271, Mar./Apr. 2017.

[3] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, MIT Press, 2001, pp. 849–856.

[4] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, "Ultra-scalable spectral clustering and ensemble clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1212–1226, Jun. 2020.

[5] A. Friedman, M. D. Keselman, L. G. Gibb, and A. M. Graybiel, "A multi-stage mathematical approach to automated clustering of high-dimensional noisy data," in *Proc. Nat. Acad. Sci. USA*, vol. 112, no. 14, pp. 4477–4482, 2015.

[6] J.-J. Liu, Q. Hou, and M.-M. Cheng, "Dynamic feature integration for simultaneous detection of salient object, edge, and skeleton," *IEEE Trans. Image Process.*, vol. 29, pp. 8652–8667, 2020.

[7] B. Byeon and K. Rasheed, "Simultaneously removing noise and selecting relevant features for high dimensional noisy data," in *Proc. 7th Int. Conf. Mach. Learn. Appl.*, 2008, pp. 147–152.

[8] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 838–845.

[9] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "Marginalized multiview ensemble clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 600–611, Feb. 2020.

[10] D. Ghoshdastidar and A. Dukkipati, "A provable generalized tensor spectral method for uniform hypergraph partitioning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 400–409.

[11] D. Ghoshdastidar and A. Dukkipati, "Spectral clustering using multilinear SVD: Analysis, approximations and applications," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 2610–2616.

[12] D. Ghoshdastidar and A. Dukkipati, "Uniform hypergraph partitioning: Provable tensor methods and sampling techniques," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 1638–1678, 2017.

[13] H. Peng, Y. Hu, J. Chen, H. Wang, Y. Li, and H. Cai, "Integrating tensor similarity to enhance clustering performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2582–2593, May 2022.

[14] G. Chao, S. Wang, S. Yang, C. Li, and D. Chu, "Incomplete multi-view clustering with multiple imputation and ensemble clustering," *Appl. Intell.*, vol. 52, pp. 14811–14821, 2022.

[15] U. Shaham, K. Stanton, H. Li, R. Basri, B. Nadler, and Y. Kluger, "SpectralNet: Spectral clustering using deep neural networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=HJ_aoCyRZ

[16] Y. Han and M. Filippone, "Mini-batch spectral clustering," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 3888–3895.

[17] O. Shamir, "A stochastic PCA and SVD algorithm with an exponential convergence rate," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2015, pp. 144–152.

[18] C. H. Nguyen and H. Mamitsuka, "Learning on hypergraphs with sparsity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2710–2722, Aug. 2021.

[19] D. François, V. Wertz, and M. Verleysen, "The concentration of fractional distances," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 7, pp. 873–886, Jul. 2007.

[20] X. Liu et al., "Late fusion incomplete multi-view clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2410–2423, Oct. 2019.

[21] V. Govindu, "A tensor decomposition for geometric grouping and segmentation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 1150–1157.

[22] A. Shashua, R. Zass, and T. Hazan, "Multi-way clustering using super-symmetric non-negative tensor factorization," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2006, pp. 595–608.

[23] S. Hu and L. Qi, "Algebraic connectivity of an even uniform hypergraph," *J. Combinatorial Optim.*, vol. 24, no. 4, pp. 564–579, 2012.

[24] G. Li, L. Qi, and G. Yu, "The z-eigenvalues of a symmetric tensor and its application to spectral hypergraph theory," *Numer. Linear Algebra Appl.*, vol. 20, no. 6, pp. 1001–1029, 2013.

[25] L. Qi, "H -eigenvalues of Laplacian and signless Laplacian tensors," *Commun. Math. Sci.*, vol. 12, no. 6, pp. 1045–1064, 2014.

[26] Y. Chen, L. Qi, and X. Zhang, "The fiedler vector of a Laplacian tensor for hypergraph partitioning," *SIAM J. Sci. Comput.*, vol. 39, no. 6, pp. A2508–A2537, 2017.

[27] J. Chang, Y. Chen, L. Qi, and H. Yan, "Hypergraph clustering using a new Laplacian tensor with applications in image processing," *SIAM J. Imag. Sci.*, vol. 13, no. 3, pp. 1157–1178, 2020.

[28] W. Yang, C. Hui, D. Sun, X. Sun, and Q. Liao, "Clustering through probability distribution analysis along eigenpaths," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 2, pp. 875–884, Feb. 2021.
[29] Y. Chen et al., "KNN-BLOCK DBSCAN: Fast clustering for large-scale data," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 51, no. 6, pp. 3939–3953, Jun. 2021.
[30] X. Xu, J. Li, M. Zhou, J. Xu, and J. Cao, "Accelerated two-stage particle swarm optimization for clustering not-well-separated data," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 11, pp. 4212–4223, Nov. 2020.
[31] Y. Zhao and X. Li, "Spectral clustering with adaptive neighbors for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2068–2078, Apr. 2023.
[32] J. Cai, J. Fan, W. Guo, S. Wang, Y. Zhang, and Z. Zhang, "Efficient deep embedded subspace clustering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1–10.
[33] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2016, pp. 478–487.
[34] J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan, "Deep self-evolution clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 809–823, Apr. 2020.
[35] J. Wu et al., "Deep comprehensive correlation mining for image clustering," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8150–8159.
[36] J. Huang, S. Gong, and X. Zhu, "Deep semantic clustering by partition confidence maximisation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8849–8858.
[37] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Curran Associates, Inc., 2017, pp. 23–32.
[38] M. Tan and Q. Le, "EfficientNetv2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 10096–10106.
[39] A. Bhattacharjee et al., "Classification of human lung carcinomas by MRNA expression profiling reveals distinct adenocarcinoma subclasses," in *Proc. Nat. Acad. Sci. USA*, vol. 98, no. 24, pp. 13790–13795, 2001.
[40] W. A. Freije et al., "Gene expression profiling of gliomas strongly predicts survival," *Cancer Res.*, vol. 64, no. 18, pp. 6503–6510, 2004.
[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[42] D. D. Lewis, Y. Yang, T. Russell-Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.
[43] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1965–1972.

Hongmin Cai (Senior Member, IEEE) received the BS and MS degrees in mathematics from the Harbin Institute of Technology, Harbin, China, in 2001 and 2003, respectively, and the PhD degree in applied mathematics from Hong Kong University, in 2007. He is a professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. He had been a guest professor with Kyoto University, Japan, in 2019. His current research interests include bioinformatics, machine learning, and omics data analysis.

Yu Hu received the bachelor of science degree in electrical engineering and automation from the School of Information and Electric Engineering, China University of Mining and Technology, Xuzhou, Jiangsu, in 2017, and the PhD degree in computer science from the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, in 2023. He is a research associate currently affiliated with the Guangdong Artificial Intelligence and Digital Economy Laboratory (Pazhou Lab) situated in Guangzhou. His present research endeavors revolve around the domains of reinforcement learning, test-time domain adaptation, and life-long learning.

Fei Qi received the BS and MS degrees from Xiamen University, Xiamen, Fujian, China, in 2013 and 2016, respectively. He is currently working toward the PhD degree in computer science and engineering with the South China University of Technology, Guangzhou, Guangdong, China. His research interests include machine learning and image processing.

Bin Hu (Fellow, IEEE) received the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, China, in 1998. Since 2008, he has been a professor and the dean of the School of Information Science and Engineering, Lanzhou University, China. He had been also guest professorship in ETH Zurich, Switzerland till 2011. His research interests include pervasive computing, computational psychophysiology, and data modeling.

Yiu-ming Cheung (Fellow, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. He is currently a chair professor with the Department of Computer Science in Hong Kong Baptist University, Hong Kong. His current research interests include machine learning and visual computing, as well as their applications in data science, pattern recognition, multi-objective optimization, and information security. He is the editor-in-chief of *IEEE Transactions on Emerging Topics in Computational Intelligence*. Also, he serves as an associate editor for *IEEE Transactions on Cybernetics*, *IEEE Transactions on Cognitive and Developmental Systems*, *Pattern Recognition*, *Knowledge and Information Systems*, and *Neurocomputing*, just to name a few. He is a fellow of the AAAS, IET, BCS, and AAIA.