Consensus Affinity Graph Learning for Multiple Kernel Clustering

Zhenwen Ren, *Member, IEEE*, Simon X. Yang, *Senior Member, IEEE*, Quansen Sun, *Member, IEEE*, and Tao Wang, *Member, IEEE*

***Abstract*—Significant attention to multiple kernel graph-based clustering (MKGC) has emerged in recent years, primarily due to the superiority of multiple kernel learning (MKL) and the outstanding performance of graph-based clustering. However, many existing MKGC methods design a fat model that poses challenges for computational cost and clustering performance, as they learn both an affinity graph and an extra consensus kernel cumbersomely. To tackle this challenging problem, this article proposes a new MKGC method to learn a consensus affinity graph directly. By using the self-expressiveness graph learning and an adaptive local structure learning term, the local manifold structure of the data in kernel space is preserved for learning multiple candidate affinity graphs from a kernel pool first. After that, these candidate affinity graphs are synthesized to learn a consensus affinity graph via a thin autoweighted fusion model, in which a self-tuned Laplacian rank constraint and a top-*k* neighbors sparse strategy are introduced to improve the quality of the consensus affinity graph for accurate clustering purposes. The experimental results on ten benchmark datasets and two synthetic datasets show that the proposed method consistently and significantly outperforms the state-of-the-art methods.**

***Index Terms*—Affinity graph learning, multiple kernel clustering, multiple kernel learning, subspace clustering.**

I. Introduction

CLUSTERING has long been a fundamental topic in the machine learning and artificial intelligence community. The goal of clustering is to partition unlabeled data into different clusters. Recently, there has been a surge of graph-based clustering methods [1]–[7], which consist of first constructing an affinity matrix/graph using graphical representations of the relationships among data points, and then applying spectral or graph-theoretic algorithms to accomplish clustering [8].

It is well known that the performance of such graph-based methods is heavily dependent on the quality of the constructed affinity graph [4], [9]–[12]. However, in practice, the data points may not fit exactly to a linear pattern, so the affinity graph constructed on the nonlinear data is unable to well reveal the similarity structure [13]. To efficiently handle nonlinear data, some methods have consequently extended linear hypothesis to nonlinear counterparts by a kernel trick, namely, single kernel learning (SKL) [13]. Although SKL is a powerful technology to efficiently handle nonlinear data, its performance is crucially determined by the choice of kernel function and parameters, that are not user friendly since the most suitable kernel and the associated parameters for a specific dataset are usually difficult to decide [14]. But multiple kernel learning (MKL) [15], [16] is an efficient technology for performing automatic kernel selection and integrating complementary information between multiple base kernels. When integrating MKL and graph-based clustering, this is known as multiple kernel graph-based clustering (MKGC).

Recently, many MKGC methods have produced and achieved promising results [14]–[24], which typically work as follows: 1) constructing multiple kernel Gram matrices by relying on multiple base kernels; 2) learning an affinity graph and a consensus kernel; and 3) producing the clustering result using this learned affinity graph. Fig. 1(a) gives the illustration of these methods. However, these existing methods still suffer from the following drawbacks: 1) they learn affinity graph and consensus kernel simultaneously, resulting in a fat model and high computational complexity; 2) they pay more attention to the consensus kernel rather than an affinity graph, and this violates the fact that the affinity graph is the key of the graph-based clustering task; and 3) they mainly use self-expressiveness framework to learn an affinity graph, but ignore the local manifold structure of the data in a kernel space, thus impairing the final clustering performance greatly.

To tackle these problems, in this article, we propose a new MKGC method, called CAGL, which contains two parts: 1) candidate affinity graphs learning (CAGL-I) and 2) consensus affinity graph learning (CAGL-II). Fig. 1(b) gives the illustration of CAGL. Different from the existing MKGC

Manuscript received December 12, 2019; revised April 6, 2020; accepted June 4, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61673220 and Grant 61802188, in part by the Intelligent Manufacturing and Robot Special Project of Major Science and Technology Project in Sichuan under Grant 2020ZDZX0014, in part by the Science and Technology Special Project of Major Scientific Instruments and Equipment Project in Sichuan under Grant 19ZDZX0119, and in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180458. This article was recommended by Associate Editor R. Tagliaferri. *(Corresponding author: Quansen Sun.)*

Zhenwen Ren is with the School of National Defence Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China, and also with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: rzw@njust.edu.cn).

Simon X. Yang is with the Advanced Robotics and Intelligent Systems Laboratory, School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada (e-mail: syang@uoguelph.ca).

Quansen Sun and Tao Wang are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: sunquansen@njust.edu.cn; wangtaoatnjust@163.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.3000947

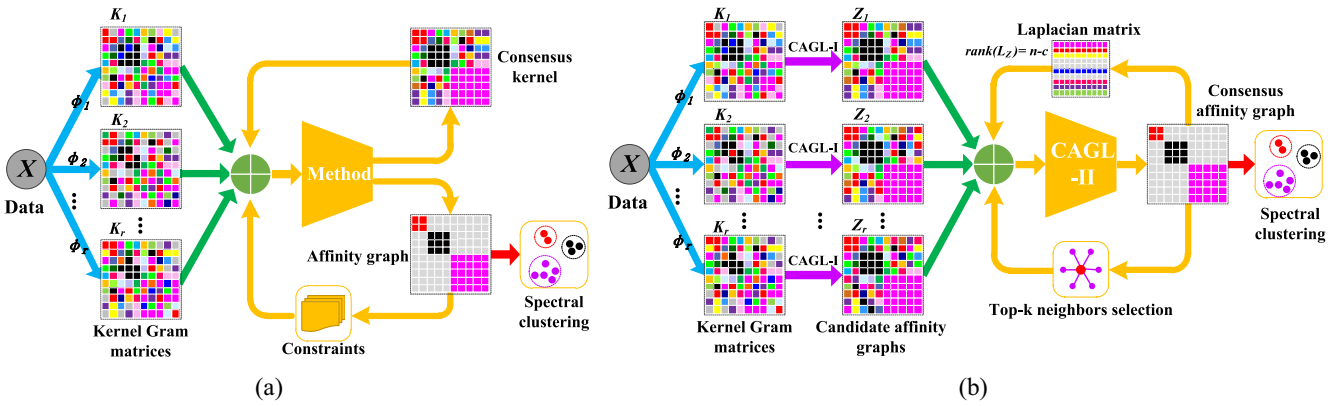


Fig. 1. Existing MKGC methods usually learn an affinity graph and an extra consensus kernel cumbersomely, hence they face challenges for computational cost and clustering performance. In contrast, the proposed method learns a consensus affinity graph directly. (a) Illustration of the existing MKGC methods. (b) Illustration of the proposed MKGC method (i.e., CAGL).

methods, which learn a consensus kernel and an affinity graph, CAGL learns a consensus affinity graph from multiple candidate affinity graphs directly. Especially, CAGL-I encourages preserving the local manifold structure of the input data in the kernel space in order to provide multiple candidate complete affinity graphs to CAGL-II. Subsequently, CAGL-II concentrates on learning an anticipant consensus affinity graph by using a thin autoweighted graph fusion model that automatically assigns an appropriate weight to each candidate affinity graph. Notably, to obtain an anticipant consensus affinity graph with better intercluster separability and exactly c connected components (where c is the number of clusters), a Laplacian rank constraint is imposed; meanwhile, a top-k neighbors sparse strategy is introduced to remove the redundant graph edges. Afterward, an efficient algorithm is derived to solve the resulting thin model. The main contributions are summarized as follows.

1) An adaptive local structure learning (ALSL) term is proposed to preserve the local manifold structure in kernel space, which reveals its latent correlation to the widely employed kernel self-expressiveness graph learning model.
2) A novel MKGC method, called CAGL is proposed, which is completely different from the existing MKGC methods. To the best of our knowledge, this is the first study to transfer the goal of MKL to multiple graph learning. Overall, CAGL has the following merits: 1) it has lower computational complexity, because the optimization target is simple and direct; 2) the imposed rank constraint can self-tune its effect to force the resulting affinity graph meeting exactly c connected components; 3) the top-k neighbors sparse strategy can cut the redundant edges of the affinity graph; and 4) the proposed thin autoweighted graph fusion model can integrate complementary information between different candidate affinity graphs. Consequentially, 2)–4) ensure that CAGL learns a higher quality affinity graph for the clustering purpose.
3) The experiments on ten benchmark datasets and two synthetic datasets validate the effectiveness and efficiency of CAGL. To the best of our knowledge, the highest multiple kernel clustering performance is obtained to date reported.

Paper Organization: The remainder of this article is organized as follows. In Section II, the related works related to self-expressiveness graph learning and MKGC are briefly overviewed. Then, the proposed CAGL method is presented in Section III. In Section IV, an efficient algorithm is given to tackle the final objective function and some analysis of the algorithm is presented. Sections V and VI offer the sufficient experimental results and conclusion, respectively.

Notation Summary: We denote a matrix by a boldface capital letter and a column vector by a boldface lowercase letter. For matrix $\boldsymbol{A} \in \mathbb{R}^{d\times n}$, the (i,j)th entry of $\boldsymbol{A}$ is denoted by a_{ij} and the transpose of the ith row of $\boldsymbol{A}$ is denoted by $\boldsymbol{a}_i \in \mathbb{R}^{n\times 1}$. For vector $\boldsymbol{a} \in \mathbb{R}^{d\times 1}$, the ith entry of $\boldsymbol{a}$ is denoted by a_i. For $\mathbf{1}$ and $\boldsymbol{I}$, they indicate the vectors of all ones and identity matrix with compatible sizes, respectively. Moreover, $\lfloor \boldsymbol{a} \rfloor_+ = \max(0, \boldsymbol{a})$ stands for the non-negative part of $\boldsymbol{a}$. $\mathrm{Tr}(\mathbf{A})$, $\mathrm{rank}(\mathbf{A})$, $\|\mathbf{A}\|_F^2$, and $\|\mathbf{A}\|_*$ denote the trace, rank, Frobenius-norm, and nuclear-norm of matrix $\mathbf{A}$, respectively.

II. Related Works

A. Self-Expressiveness Graph Learning

As we know, one sample from one subspace can be represented as a linear combination of other samples in the same subspace, known as the self-expressiveness property [10], [25], [26]. Accordingly, the graph-based clustering methods can rely on the self-expressiveness to build a reliable affinity graph. The mathematical definition is given by

$$\min_{\boldsymbol{Z}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{Z}\|_F^2 + \alpha \mathcal{R}(\boldsymbol{Z}) \tag{1}$$

where $\boldsymbol{Z}$ is the coefficient matrix and $\alpha > 0$ is a regularization parameter to balance the term $\mathcal{R}(\boldsymbol{Z})$. The main difference of different methods depends on the choice of term $\mathcal{R}(\boldsymbol{Z})$, such as sparse [27]; low rank [28]; structure preserving [23], [29]; block-diagonal constraint [10], [13]; adaptive weighted representation [30]; continuous label learning [19]; and more. After

obtaining $\boldsymbol{Z}$, the affinity matrix/graph is usually constructed by $\boldsymbol{Z} = (|\boldsymbol{Z}|^{\mathrm{T}} + |\boldsymbol{Z}|)/2$ to eliminate unbalance.

However, problem (1) cannot efficiently handle nonlinear data. To tackle this tricky problem, the nonlinear data can be mapped into a kernel space, and then linear pattern analysis can be accomplished [13]. Consequently, an affinity matrix in a kernel space can be learned by upgrading problem (1) to

$$\begin{aligned}&\min_{\boldsymbol{Z}} \|\phi(\boldsymbol{X}) - \phi(\boldsymbol{X})\boldsymbol{Z}\|_F^2 + \alpha\mathcal{R}(\boldsymbol{Z}) \\ &= \min_{\boldsymbol{Z}} \mathrm{Tr}\big(\boldsymbol{K} - 2\boldsymbol{K}\boldsymbol{Z} + \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Z}\big) + \alpha\mathcal{R}(\boldsymbol{Z}) \end{aligned} \tag{2}$$

where $\phi(\cdot)$ is a kernel mapping and $\boldsymbol{K}$ is a kernel Gram matrix.

B. Related MKGC Methods

In recent years, under the self-expressiveness graph learning model or other graph learning technologies [31]–[33], MKGC has gained significant attention from the research community and many MKGC methods have been proposed [14]–[23]. For instance, multiple kernel k-means (MKKM) [34] extends k-means into a multiple kernel setting for clustering. To enhance the robustness of MKKM, an extended version of MKKM, robust multiple kernel k-means (RMKKM) [15], has been proposed to simultaneously find the optimal combination of multiple kernels and the best clustering label. Based on MKKM, multiview clustering via late fusion alignment maximization (MVC-LFA) [35] proposes to maximally align the consensus partition with the weighted base partitions, which can significantly reduce the computational complexity. Regarding the similarity between the affinity matrix and the kernel matrix, affinity aggregation for spectral clustering (AASC) [36] replaces the single affinity matrix in spectral clustering with multiple kernel (SCMK) matrices, which can be deemed as an MKL version of spectral clustering. To learn a better consensus kernel, SCMKs [19] learn a consensus kernel based on the fact that the optimal consensus kernel is a linear combination of base kernels. In contrast, self-weighted MKL (SMKL) [14] assumes that the consensus kernel is a neighborhood of multiple base kernels for clustering and semisupervised classification. Similar to SMKL, neighbor-kernel-based MKL [22] considers the neighborhood structure among base kernels. By considering the low-rank property of the samples, low-rank kernel learning graph-based clustering (LKGr) [20] has been proposed to impose a low-rank constraint on the kernel matrix. To resist noise, joint robust multiple kernel subspace clustering (JMKSC) [21] uses the self-expressiveness and block-diagonal regularizer to learn a block-diagonal affinity matrix for accurate clustering. By aligning each base kernel with the corresponding local optimal similarity matrix, local kernel alignment maximization (LKAMKC) [17] improves the clustering performance for multiple kernel clustering. Based on LKAMKC, the global and local structure alignment framework for multiple kernel clustering (GLSAMKC) [18] well considers both the alignment between the global and local structures of data with the same optimal similarity matrix. Neighbor-kernel MKC [22] proposes an effective neighbor-kernel-based multiple kernels clustering method by considering the intrinsic neighborhood structure among base kernels. Recently, a local structural graph and low-rank consensus MKL (LLMKL) [23] integrates the global and local structure preserving into a unified learning model, and has produced promising results.

However, these state-of-the-art methods have some drawbacks that need to be defeated (see Section I).

III. Proposed Methodology

In this section, we first present the preprocessing step [i.e., multiple base kernels construction (MBKC)] of the kernel method to construct a kernel Gram matrix and then present the proposed CAGL method (including CAGL-I and CAGL-II) to learn a consensus affinity graph directly.

A. Multiple Base Kernels Construction

Given a columnwise data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ with d features and n unlabeled samples from c clusters. Let $\phi : \mathbb{R}^d \to \mathcal{H}$ be a mapping function that maps the data from the original space to the reproducing kernel Hilbert space $\mathcal{H}$, and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ be a positive semidefinite kernel Gram matrix, whose (i, j)th entry is computed as

$$\boldsymbol{K}_{ij} = \big(\phi(\boldsymbol{X})^{\mathrm{T}}\phi(\boldsymbol{X})\big)_{ij} = \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi\big(\boldsymbol{x}_j\big) = ker\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) \tag{3}$$

where *ker* is a kernel function. More often than not, we assume that $\phi(\boldsymbol{x}_i)_{i=1}^n$ resides in multiple linear spaces in $\mathcal{H}$.

For MKL, multiple base kernels (i.e., $\phi_1, \ldots, \phi_r$) are constituted of a kernel pool, thus, their corresponding kernel Gram matrices (i.e., $\boldsymbol{K}^1, \ldots, \boldsymbol{K}^r$) are constructed accordingly.

B. Candidate Affinity Graphs Learning

Although problem (2) can obtain an affinity graph in $\mathcal{H}$, the learned affinity graph never gives much thought to the local manifold structure of the data in $\mathcal{H}$, which contains important discriminative information for graph learning [37].

A local structure preserving model, ALSL [31], [33], [38], is widely employed in linear space for unsupervised feature extraction [39]; graph learning [40], [41]; and data clustering [2]. The model of ALSL is

$$\min_{\boldsymbol{Z}} \sum_{i,j=1}^{n} \Big(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 z_{ij} + \beta z_{ij}^2\Big) \quad \text{s.t. } \boldsymbol{z}_i^{\mathrm{T}}\mathbf{1} = 1, 0 \le z_{ij} \tag{4}$$

where $\beta > 0$ is a parameter, and the constraint can guarantee the probability property of each row (i.e., $\boldsymbol{z}_i$) of $\boldsymbol{Z}$.

However, ALSL is designed to the original space rather than a kernel space. In order to preserve the local manifold structure in a kernel space, we propose a new model, namely, kernel ALSL (KALSL), which is formulated as

$$\min_{\boldsymbol{Z}} \sum_{i,j=1}^{n} \Big(-ker\big(\boldsymbol{x}_i, \boldsymbol{x}_j\big) z_{ij} + \beta z_{ij}^2\Big) \quad \text{s.t. } \boldsymbol{z}_i^{\mathrm{T}}\mathbf{1} = 1, 0 \le z_{ij} \tag{5}$$

where $-ker(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can present the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in $\mathcal{H}$, and the second term is used to avoid trivial solution.

Mathematically, problem (5) can be transformed into

$$\min_{\boldsymbol{Z}} -\mathrm{Tr}(\boldsymbol{K}\boldsymbol{Z}) + \beta\|\boldsymbol{Z}\|_F^2 \quad \text{s.t. } \boldsymbol{Z}\mathbf{1} = \mathbf{1}, 0 \le \boldsymbol{Z}. \tag{6}$$

It can be seen that (2) already comprises a negative term, that is, $-\operatorname{Tr}(\boldsymbol{KZ})$. Therefore, KALSL can explain this term in problem (2) well, that is, it can preserve the local manifold structure in $\mathcal{H}$ such that the learned affinity graph contains more detailed local structure information. Evidently, the power of KALSL can be tuned rather than a fixed value. Thus, (2) can be upgraded to

$$\begin{aligned} \min_{\boldsymbol{Z}} \ & \operatorname{Tr}\big(\boldsymbol{K}+\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Z}\big)-\alpha\operatorname{Tr}(\boldsymbol{K}\boldsymbol{Z})+\beta\|\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \ & \boldsymbol{Z}\boldsymbol{1}=\boldsymbol{1}, 0\le\boldsymbol{Z} \end{aligned} \tag{7}$$

where both $\alpha \geq 2$ and $\beta > 0$ are tradeoff parameters. $\mathcal{R}(\boldsymbol{Z}) = \|\boldsymbol{Z}\|_F^2$ can enhance the grouping effect and avoid trivial solution.

For given r base kernels, their corresponding candidate affinity graphs $\{\boldsymbol{Z}^m\}_{m=1}^r$ can be learned separately using (7).

C. Consensus Affinity Graph Learning

In order to learn an optimal consensus affinity graph for the clustering purpose, we propose a thin autoweighted fusion model by assuming that the important graphs have big weight values while the unimportant graphs have low or near-zero weight values. Therefore, this intuitive thought can be formulated as

$$\min_{\boldsymbol{Z}} \sum_{m=1}^{r} w_m\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2 \quad \text{s.t. } 0\le\boldsymbol{Z}, \boldsymbol{Z}\boldsymbol{1}=\boldsymbol{1} \tag{8}$$

where $\boldsymbol{Z}$ is the anticipated consensus affinity graph and $\boldsymbol{w} = [w_1, \ldots, w_r]^{\mathrm{T}}$ is the weight vector indicated the contribution of r candidate affinity graphs.

However, the learned graph from (8) is a redundancy graph (RG), whose edges need to be carefully adjusted and cut, mainly for the following two flaws: 1) RG contains too many redundant edges that need to be cut and 2) RG does not contain exact c connected components. These flaws are crucial to clustering performance [10]. For the first flaw, we propose to construct a sparse graph by using the top-k neighbors selection strategy; for the second flaw, Theorem 1 can tackle it.

Theorem 1 [10]: For any affinity matrix $\boldsymbol{Z}$, the multiplicity c of the eigenvalue 0 of the corresponding Laplacian matrix $\boldsymbol{L}_Z$ equals the number of connected components (blocks) in the graph $\boldsymbol{Z}$, where the Laplacian matrix $\boldsymbol{L}_Z = \boldsymbol{D}-(\boldsymbol{Z}^{\mathrm{T}}+\boldsymbol{Z})/2$ and $\boldsymbol{D}$ is the diagonal degree matrix with $d_{ii} = \sum_j[(z_{ij}+z_{ji})/2]$.

Motivated by such a theorem, it can be proved that if $\operatorname{rank}(\boldsymbol{L}_Z) = n - c$, the affinity graph $\boldsymbol{Z}$ will contain exact c connected components. According to rank theory and Fan's theorem [3], we obtain

$$\operatorname{rank}(\boldsymbol{L}_Z)=n-c \Rightarrow \min_{\boldsymbol{P}\in\mathbb{R}^{n\times c},\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}=\boldsymbol{I}} \operatorname{Tr}\big(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P}\big) \tag{9}$$

where $\boldsymbol{P}^{\mathrm{T}} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ is the cluster indicator matrix.

By doing so, the learned $\boldsymbol{Z}$ is block diagonal with proper permutation. As a result, problem (8) can be transformed into

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{Z},\boldsymbol{P}} \ & \sum_{m=1}^{r} w_m\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2+\gamma\operatorname{Tr}\big(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P}\big) \\ \text{s.t.} \ & 0\le\boldsymbol{Z}, \boldsymbol{Z}\boldsymbol{1}=\boldsymbol{1}, \boldsymbol{P}\in\mathbb{R}^{n\times c}, \boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}=\boldsymbol{I} \end{aligned} \tag{10}$$

where $\gamma > 0$ is a self-tuned tradeoff parameter.

IV. Optimization

The final problem (7) (i.e., CAGL-I) has a closed-form solution, while problem (10) (i.e., CAGL-II) is hard to be tackled directly. In this section, we propose a coordinate descent-based iterative algorithm to solve problem (10), since each subproblem of (10) is convex and differentiable.

A. Solving CAGL-I

Analogously to [30], for computational efficiency, a latent solution $\hat{\boldsymbol{Z}}$ of (7) can be first computed by solving

$$\hat{\boldsymbol{Z}} = \arg\min_{\boldsymbol{Z}} \operatorname{Tr}\big(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Z}\big)-\alpha\operatorname{Tr}(\boldsymbol{K}\boldsymbol{Z})+\beta\|\boldsymbol{Z}\|_F^2 \tag{11}$$

which has a closed-form solution as $\hat{\boldsymbol{Z}} = (\boldsymbol{K}+\beta\boldsymbol{I})^{-1}(\alpha/2\boldsymbol{K})$. After that, the optimal solution $\boldsymbol{Z}^*$ of (7) can be easily obtained by optimizing the following alternative problem:

$$\min_{\boldsymbol{Z}\ge 0,\boldsymbol{Z}\boldsymbol{1}=\boldsymbol{1}} \|\boldsymbol{Z}-\hat{\boldsymbol{Z}}\|_F^2 \tag{12}$$

which can be efficiently solved by an iterative algorithm [30].

B. Solving CAGL-II

1) Update $\boldsymbol{w}$ as Given $\boldsymbol{Z}$ and $\boldsymbol{P}$: The weight of the mth candidate graph is w_m, which is computed by Proposition 1.

Proposition 1: The weight of the mth candidate affinity graph is determined by $w_m = [1/(2\sqrt{\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2})]$.

Proof: Motivated by the iteratively reweighted technique [42], we define an auxiliary problem without $\boldsymbol{w}$ as follows:

$$\begin{aligned} \min_{\boldsymbol{Z}} \ & \sum_{m=1}^{r}\sqrt{\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2}+\lambda\operatorname{Tr}\big(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P}\big) \\ \text{s.t.} \ & z_{ij}\ge 0, \boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{1}=1, \boldsymbol{P}\in\mathbb{R}^{n\times c}, \boldsymbol{P}^{\mathrm{T}}\boldsymbol{P}=\boldsymbol{I}. \end{aligned} \tag{13}$$

The Lagrange function of (13) is $\sum_{m=1}^{r}\sqrt{\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2}+\lambda\operatorname{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P})+\Phi(\boldsymbol{\Lambda},\boldsymbol{Z})$, where $\boldsymbol{\Lambda}$ is the Lagrange multiplier and $\Phi(\boldsymbol{\Lambda},\boldsymbol{Z})$ indicates the indicator function of $\boldsymbol{Z}$ from the constraints (i.e., $z_{ij}\ge 0, \boldsymbol{z}_i^{\mathrm{T}}\boldsymbol{1}=1$). By setting this Lagrange function with respect to w.r.t. $\boldsymbol{Z}$ as 0, we then have

$$\sum_{m=1}^{r}\widehat{w}_m\frac{\partial\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F^2}{\partial\boldsymbol{Z}}+\frac{\partial\Omega(\boldsymbol{Z})}{\partial\boldsymbol{Z}}=0 \tag{14}$$

where $\Omega(\boldsymbol{Z})=\lambda\operatorname{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P})+\Phi(\boldsymbol{\Lambda},\boldsymbol{Z})$ and $\widehat{w}_m = 1/(2\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F)$. Obviously, problem (14) is the same as the derivation of the Lagrange function of problem (10). Thus, $\widehat{w}_m$ can be considered as w_m in (10). To avoid dividing by 0 in theory, $\widehat{w}_m$ can be transformed into

$$w_m=\frac{1}{2\|\boldsymbol{Z}-\boldsymbol{Z}^m\|_F+\zeta} \tag{15}$$

where ζ is infinitely close to 0. The proof is completed. ■

2) Update $\boldsymbol{Z}$ as Given $\boldsymbol{w}$ and $\boldsymbol{P}$: The optimization problem for updating the affinity matrix/graph $\boldsymbol{Z}$ can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{Z}} \ & \sum_{m=1}^{r}\sum_{i,j=1}^{n} w_m\Big(z_{ij}-z_{ij}^m\Big)^2+\gamma\sum_{i,j=1}^{n}\|\boldsymbol{p}_i-\boldsymbol{p}_j\|_2^2 z_{ij} \\ \text{s.t.} \ & 0\le z_{ij}, \boldsymbol{1}^{\mathrm{T}}\boldsymbol{z}_i=1 \end{aligned} \tag{16}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are the ith and jth rows of $\boldsymbol{P}$, respectively. By using the mathematical derivation in the Appendix, we then can independently optimize each row of $\boldsymbol{Z}$ (i.e., $\boldsymbol{z}_i$) via

$$\min_{\boldsymbol{z}_i} \sum_{m=1}^{r} \left\| \boldsymbol{z}_i - \boldsymbol{e}^m \right\|_2^2 \quad \text{s.t. } 0 \le \boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{T}} \mathbf{1} = 1 \tag{17}$$

where $\boldsymbol{e}^m = \boldsymbol{z}_i^m - (\gamma / 2 r w_m) \boldsymbol{f}_i$ and $f_{ij} = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2^2$. Such a problem can be efficiently solved by Theorem 2.

Theorem 2: For any r vectors $\{\boldsymbol{e}^m\}_{m=1}^r$, the following problem has a closed-form solution $\boldsymbol{z}^*$, that is:

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}} \sum_{m=1}^{r} \left\| \boldsymbol{z} - \boldsymbol{e}^m \right\|_2^2 \quad \text{s.t. } 0 \le \boldsymbol{z}, \boldsymbol{z}_i^{\mathrm{T}} \mathbf{1} = 1. \tag{18}$$

Proof: Analogously to [9], the Lagrangian function $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\omega}, \pi)$ of problem (18) is defined as

$$\mathcal{L}(\boldsymbol{z}, \boldsymbol{\omega}, \pi) = \frac{1}{2} \sum_{m=1}^{r} \left\| \boldsymbol{z} - \boldsymbol{e}^m \right\|_2^2 - \pi \left(\boldsymbol{z}_i^{\mathrm{T}} \mathbf{1} - 1 \right) - \boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{z} \tag{19}$$

where $\boldsymbol{\omega} \ge 0$ and π are Lagrange multipliers.

Suppose the optimal solution of (19) is $\boldsymbol{z}^*$ when $\boldsymbol{\omega} = \boldsymbol{\omega}^*$ and $\pi = \pi^*$. According to the Karush–Kuhn–Tucker (KKT) conditions [43], for $\forall j$, we have $\sum_{m=1}^r z_j^* - \sum_{m=1}^r e_j^m - \omega_j^* - \pi^* = 0$, $z_j^* \ge 0$, $\omega_j^* \ge 0$, and $z_j^* \omega_j^* = 0$. Obviously, we also have $r\boldsymbol{z}^* - \sum_{m=1}^r \boldsymbol{e}^m - \boldsymbol{\omega}^* - \pi^* \mathbf{1} = \mathbf{0}$.

According to the constraint $\mathbf{1}^{\mathrm{T}} \boldsymbol{z}^* = 1$, $\pi^* = (r - \sum_{m=1}^r \mathbf{1}^{\mathrm{T}} \boldsymbol{e}^m - \mathbf{1}^{\mathrm{T}} \boldsymbol{\omega}^*)/n$ is obtained. So the optimal solution $\boldsymbol{z}^*$ is formulated as

$$\frac{\sum_{m=1}^r \boldsymbol{e}^m}{r} + \frac{\mathbf{1}}{n} - \frac{\sum_{m=1}^r \mathbf{1}^{\mathrm{T}} \boldsymbol{e}^m \mathbf{1}}{rn} - \frac{\mathbf{1}^{\mathrm{T}} \boldsymbol{\omega}^* \mathbf{1}}{rn} + \frac{\boldsymbol{\omega}^*}{r}. \tag{20}$$

Defining $\boldsymbol{g} = \sum_{m=1}^r \boldsymbol{e}^m / r + \mathbf{1}/n - \sum_{m=1}^r \mathbf{1}^{\mathrm{T}} \boldsymbol{e}^m \mathbf{1} / (rn)$ and $\hat{\omega}^* = \mathbf{1}^{\mathrm{T}} \boldsymbol{\omega}^* / (rn)$, (20) can then be simplified to $\boldsymbol{z}^* = \boldsymbol{g} - \hat{\omega}^* \mathbf{1} + [(\boldsymbol{\omega}^*)/r]$. Consequently, for $\forall j$, we have

$$z_j^* = g_j - \hat{\omega}^* + \frac{\omega_j^*}{r} = \lfloor g_j - \hat{\omega}^* \rfloor_+. \tag{21}$$

Similarly, we can derive $\omega_j^* = r \lfloor \hat{\omega}^* - g_j \rfloor_+$. Due to $\hat{\omega}^* = \mathbf{1}^{\mathrm{T}} \boldsymbol{\omega}^* / (rn)$, the optimal solution $\hat{\omega}^*$ is represented as $\hat{\omega}^* = (1/n) \sum_{j=1}^n \lfloor \hat{\omega}^* - g_j \rfloor_+$. To solve the self-dependent $\hat{\omega}^*$, we define an auxiliary function as

$$\Theta(\hat{\omega}) = \frac{1}{n} \sum_{j=1}^{n} \lfloor \hat{\omega}^* - g_j \rfloor_+ - \hat{\omega}. \tag{22}$$

Due to the fact that $\hat{\omega} \ge 0$ and $\Theta'(\hat{\omega}) \le 0$, and $\Theta'(\hat{\omega})$ is a piecewise linear and convex function, the Newton method can be employed to find the root $\hat{\omega}^*$ when $\Theta(\hat{\omega}) = 0$, that is

$$\hat{\omega}_{t+1} = \hat{\omega}_t - \frac{\Theta(\hat{\omega}_t)}{\Theta'(\hat{\omega}_t)} \tag{23}$$

where t is the iteration number. Hence, the closed-form solution $\boldsymbol{z}^*$ is (21). This completes the proof. ■

After obtaining $\boldsymbol{z}_i$, a top-k neighbors selection strategy is introduced to control the number of nearest neighbors of $\boldsymbol{x}_i$ that could have a chance to connect to $\boldsymbol{x}_i$. This guarantees that the number of nonzero entries in each row of $\boldsymbol{Z}$ is k ($k \in \{1, \ldots, n\}$). To this end, the resulting $\boldsymbol{Z}$ is naturally sparse.

Algorithm 1 Algorithm of the Proposed CAGL Method

Input: Data matrix $\boldsymbol{X}$, and parameters α and k.
Initialize: $\{w_m\}_{m=1}^r = \frac{1}{r}$, $\beta = \gamma = 1$, and $mit = 1e3$.
Learn r base kernel matrices $\{\boldsymbol{K}^m\}_{m=1}^r$ via (3).
Learn r candidate affinity graphs $\{\boldsymbol{Z}^m\}_{m=1}^r$ via (7).
Set $\boldsymbol{Z} = \frac{1}{r} \sum_{m=1}^r \boldsymbol{Z}^m$ and $t = 0$.
while $(\mathrm{rank}(\boldsymbol{L}_Z) ! = n - c)$ and $(t{+}{+} < mit)$ **do**
  Update $\boldsymbol{w}^{(t+1)}$ using (15).
  Update $\boldsymbol{Z}^{(t+1)}$ using (16) and compute $\boldsymbol{L}_Z$.
  Update $\boldsymbol{P}^{(t+1)}$ using (25) and self-tune γ.
end while
Accomplish spectral clustering using the affinity graph $\boldsymbol{Z}$.
Output: The clustering results: ACC, NMI, purity.

TABLE I
SUMMARY OF NOTATIONS FOR COMPLEXITY ANALYSIS

Notation	Meanings	Remark
n	The numbers of samples	Decided by dataset
m	The numbers of samples	Decided by dataset
c	The numbers of clusters	Decided by dataset
d	Sample dimensionality	Decided by dataset
r	The numbers of base kernels	r = 12
t	The numbers of iterations	Decided by convergence
L	The size of the problem encoded	Used in [22]

3) Update $\boldsymbol{P}$ as Given $\boldsymbol{w}$ and $\boldsymbol{Z}$: The subproblem for updating $\boldsymbol{P}$ becomes as follows:

$$\min_{\boldsymbol{P}} \gamma \operatorname{Tr}\left(\boldsymbol{P}^{\mathrm{T}} \boldsymbol{L}_Z \boldsymbol{P} \right) \quad \text{s.t. } \boldsymbol{P}^{\mathrm{T}} \boldsymbol{P} = \boldsymbol{I}. \tag{24}$$

The solution of (24) is the eigenvectors corresponding to the smallest c eigenvalues of the Laplacian matrix $\boldsymbol{L}_Z$.

For obtaining an optimal affinity graph with exact c diagonal blocks, in each iteration, we automatically double γ or halve γ when the connectivity of the resulting affinity graph $\boldsymbol{Z}$ is smaller or greater than the number of clusters c, respectively, until to achieve the convergence criterion $\mathrm{rank}(\boldsymbol{L}_Z) = n - c$

$$\gamma = \begin{cases} 2\gamma, & b < c \\ \frac{\gamma}{2}, & b > c \end{cases} \tag{25}$$

where b is the connectivity of graph $\boldsymbol{Z}$ in each iterator.

The pseudocode of our CAGL is depicted as in Algorithm 1. The demo code will be released on our GitHub homepage.

C. Complexity

Obviously, the computational complexity of our CAGL mainly depends on the complexity of Algorithm 1, which contains a candidate graph constructing step (i.e., CAGL-I) and three main iterator steps (i.e., CAGL-II). The notations that are used for complexity analysis are summarized in Table I.

First, constructing each candidate graph $\boldsymbol{Z}_i$ ($i = 1, \ldots, r$) mainly involves the matrix inversion, so the computational complexity of solving (11) is $\mathcal{O}(rn^3)$. Then, in each iterator, three subproblems need to solve, whose complexities are induced as follows: 1) the first iterator step is to solve the weight vector $\boldsymbol{w}$ using (15), whose computational complexity is $\mathcal{O}(rn^2)$; 2) the second iterator step is to calculate $\boldsymbol{z}_i$ ($i = 1, \ldots, n$) using (16), whose computational complexity is $\mathcal{O}(n)$; and 3) the third iterator step is to solve (11), which needs to solve the eigenvectors of the Laplacian matrix $\boldsymbol{L}_Z$,

TABLE II
SUMMARY OF COMPUTATIONAL COMPLEXITIES OF MANY EXISTING MKGC METHODS

Method	Ref.	Computational complexity
MVC-LFA	[35]	$O(t(nc^2 + rc^3))$
LLMKL	[23]	$O(t(2n^3 + (c+2)n^2 + r^3))$
JMKSC	[21]	$O(t(2n^3 + (c+2)n^2))$
SMKL	[14]	$O(t((r+1)n^3 + (c+1)n^2))$
SCMK	[19]	$O(t((r+1)n^3 + (r+c+1)n^2))$
LKGr	[20]	$O(t(2n^3 + (r+c+1)n^2 + r^3))$
RMKKM	[15]	$O(drn^3 + rt(n^3 + n^2 + n))$
GLSAMKC	[18]	$O(t((r+1)n^3 + rcn^2 + r^3))$
LKAMKC	[17]	$O(t((r^2 + r + c + 1)n^3 + r^3))$
Neighbor-Kernel MKC	[22]	$O(t(Ld^3 + n^3))$

whose computational complexity is $\mathcal{O}(cn^2)$ when partial SVD is used. To summarize, the overall complexity of Algorithm 1 is

$$\mathcal{O}\Big(rn^3 + t\Big(rn^2 + n + cn^2\Big)\Big) \approx \mathcal{O}\Big(rn^3 + t(r+c)n^2\Big). \quad (26)$$

In addition, for the sake of comparison, the complexity of some recently proposed MKGC methods is presented in Table II. It is clear that the complexity of our CAGL is significantly smaller than that of the existing MKGC methods. The main reasons are that, on the one hand, Algorithm 1 does not involve the high-dimensional parameter d, and on the other hand, Algorithm 1 has good convergence and the computational complexity is only $\mathcal{O}(n^2)$ in each iteration.

D. Convergence

For the convergence study, the convergence guarantee of Algorithm 1 is easy to verify by using Proposition 2. For this purpose, let the constrained solution set of $\boldsymbol{Z}$ be $\boldsymbol{S} = \{\boldsymbol{Z} | 0 \leq \boldsymbol{Z}, \boldsymbol{Z}\mathbf{1} = \mathbf{1}\}$ and $\iota_S(\boldsymbol{Z})$ be the indicator function of $\boldsymbol{S}$, and denote the objective function of (10) as $f(\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{w})$.

Proposition 2: The sequence $\{\boldsymbol{P}^t, \boldsymbol{Z}^t, \boldsymbol{w}^t\}$ produced by Algorithm 1 has some properties as follows.

1) The subproblems w.r.t. $\boldsymbol{P}$, $\boldsymbol{Z}$, and $\boldsymbol{w}$ are convex.
2) The objective function $f(\boldsymbol{P}^t, \boldsymbol{Z}^t, \boldsymbol{w}^t) + \iota_S(\boldsymbol{Z}^t)$ is monotonically decreasing. Indeed
$$f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^{t+1}, \boldsymbol{w}^{t+1}\Big) + \iota_S\Big(\boldsymbol{Z}^{t+1}\Big) \leq f\big(\boldsymbol{P}^t, \boldsymbol{Z}^t, \boldsymbol{w}^t\big) + \iota_S\big(\boldsymbol{Z}^t\big) - \lambda\|\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t\|_F^2 \quad (27)$$
where $\lambda = \sum_{m=1}^r w_m$.
3) $\boldsymbol{P}^{t+1} - \boldsymbol{P}^t \to 0$, $\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t \to 0$ and $\boldsymbol{w}^{t+1} - \boldsymbol{w}^t \to 0$.
4) The sequence $\{\boldsymbol{P}^t\}$, $\{\boldsymbol{Z}^t\}$, and $\{\boldsymbol{w}^t\}$ are bounded.

Proof: First, from the optimality of $\boldsymbol{P}^{t+1}$ to (25), it can be deemed as spectral clustering [8], [44], whose convergence has been stated and proved in [45]. From another perspective, the Hessian matrix w.r.t. $\boldsymbol{P}$ is

$$\boldsymbol{H}_P = \frac{\partial^2 \operatorname{Tr}\big(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P}\big)}{\partial \boldsymbol{P}\partial \boldsymbol{P}^{\mathrm{T}}} = \boldsymbol{L}_Z + \boldsymbol{L}_Z^{\mathrm{T}}. \quad (28)$$

Since $\boldsymbol{L}_Z$ is symmetric (i.e., $\boldsymbol{L}_Z = \boldsymbol{L}_Z^{\mathrm{T}}$) and positive semidefinite (i.e., $\boldsymbol{L}_Z \succeq 0$), so $\boldsymbol{H}_P$ is also a symmetric positive semidefinite matrix. Therefore, $\boldsymbol{L}_Z$ can be factorized as $\boldsymbol{L}_Z = \boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$ [46], that is, $\operatorname{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P}) = \|\boldsymbol{B}\boldsymbol{P}\|_F^2$. Obviously, (24) is a convex function w.r.t. $\boldsymbol{P}$ and can monotonically decrease the objective function. Thus, we have

$$f\Big(\boldsymbol{P}^{k+1}, \boldsymbol{Z}^k, \boldsymbol{w}^k\Big) \leq f\Big(\boldsymbol{P}^k, \boldsymbol{Z}^k, \boldsymbol{w}^k\Big). \quad (29)$$

Second, from the optimality of $\boldsymbol{w}^{t+1}$ to (8), such a problem is a linear convex function and can be solved in a closed-form solution [i.e., (15)]. Thus, we have

$$f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^t, \boldsymbol{w}^{t+1}\Big) \leq f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^t, \boldsymbol{w}^t\Big). \quad (30)$$

Third, from the updating rule of (16), the Hessian matrix w.r.t. $\boldsymbol{Z}$ is also a positive semidefinite matrix. Hence, (16) is a convex function w.r.t. $\boldsymbol{Z}$. Moreover, it monotonically decreases the objective function value based on Lemma 1.

Lemma 1 [47]: The following inequality holds for any nonzero matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$:

$$\|\boldsymbol{U}\|_F - \frac{\|\boldsymbol{U}\|_F^2}{2\|\boldsymbol{V}\|_F} \leq \|\boldsymbol{V}\|_F - \frac{\|\boldsymbol{V}\|_F^2}{2\|\boldsymbol{V}\|_F}. \quad (31)$$

Letting $\Theta(\boldsymbol{Z}) = \gamma \operatorname{Tr}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{P})$ and denoting the updated $\boldsymbol{Z}$ as $\widehat{\boldsymbol{Z}}$ in each iteration, then it is easy to know that

$$\sum_{m=1}^r \frac{\big\|\widehat{\boldsymbol{Z}} - \boldsymbol{Z}^m\big\|_F^2}{2\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F} + \Theta\big(\widehat{\boldsymbol{Z}}\big) \leq \sum_{m=1}^r \frac{\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F^2}{2\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F} + \Theta(\boldsymbol{Z}). \quad (32)$$

Based on Lemma 1, we have

$$\sum_{m=1}^r \big\|\widehat{\boldsymbol{Z}} - \boldsymbol{Z}^m\big\|_F - \sum_{m=1}^r \frac{\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F^2}{2\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F} \leq \sum_{m=1}^r \big\|\boldsymbol{Z} - \boldsymbol{Z}^m\big\|_F - \sum_{m=1}^r \frac{\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F^2}{2\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F}. \quad (33)$$

Sum over both sides of (32) and (33), we then have

$$\sum_{m=1}^r \big\|\widehat{\boldsymbol{Z}} - \boldsymbol{Z}^m\big\|_F + \Theta\big(\widehat{\boldsymbol{Z}}\big) \leq \sum_{m=1}^r \big\|\boldsymbol{Z} - \boldsymbol{Z}^m\big\|_F + \Theta\big(\widehat{\boldsymbol{Z}}\big). \quad (34)$$

The inequality (34) guarantees that the objective value in (10) decreases monotonically in each iterator step until it converges. Furthermore, due to $\sum_{m=1}^r w_m\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F^2$, f is λ-strongly convex w.r.t. $\boldsymbol{Z}$, where $\lambda = \sum_{m=1}^r w_i$. Thus, we have

$$f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^{t+1}, \boldsymbol{w}^{t+1}\Big) + \iota_S\Big(\boldsymbol{Z}^{t+1}\Big) \leq f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^t, \boldsymbol{w}^{t+1}\Big) + \iota_S\big(\boldsymbol{Z}^t\big) - \lambda\Big\|\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t\Big\|_F^2. \quad (35)$$

To sum up inequalities (29), (30), and (35), we have

$$f\Big(\boldsymbol{P}^{t+1}, \boldsymbol{Z}^{t+1}, \boldsymbol{w}^{t+1}\Big) + \iota_S\Big(\boldsymbol{Z}^{t+1}\Big) \leq f\big(\boldsymbol{P}^t, \boldsymbol{Z}^t, \boldsymbol{w}^t\big) + \iota_S\big(\boldsymbol{Z}^t\big) - \lambda\Big\|\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t\Big\|_F^2. \quad (36)$$

Note that f can be rewritten as $f = \sum_{m=1}^r w_m\|\boldsymbol{Z} - \boldsymbol{Z}^m\|_F^2 + \gamma\|\boldsymbol{B}\boldsymbol{P}\|_F^2 \geq 0$. Theoretically, Algorithm 1 can monotonically decrease the objective value of f and thus f is upper bounded. This suggests that $\{\boldsymbol{P}^t\}$, $\{\boldsymbol{Z}^t\}$, and $\{\boldsymbol{w}^t\}$ are bounded.

Now, summing (36) over $t = 0, 1, \ldots,$ we have

$$\sum_{t=0}^{+\infty} \lambda\Big\|\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^t\Big\|_F^2 \leq f\Big(\boldsymbol{P}^0, \boldsymbol{Z}^0, \boldsymbol{w}^0\Big). \quad (37)$$

Fig. 2. Example images in the seven image datasets. (a) Yale. (b) Jaffe. (c) AR. (d) ORL. (e) BA. (f) COIL20. (g) CIFAR-10.

This implies

$$\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^{t} \to 0. \tag{38}$$

By (38) and the updating of $\boldsymbol{P}^{t+1}$ and $\boldsymbol{w}^{t+1}$ in (24) and (15), we have $\boldsymbol{P}^{t+1} - \boldsymbol{P}^{t} \to 0$ and $\boldsymbol{w}^{t+1} - \boldsymbol{w}^{t} \to 0$, respectively. The proof of Proposition 2 is completed. ■

V. Experiments

In this section, we evaluate the proposed CAGL method on ten benchmark datasets and two synthetic datasets to demonstrate its effectiveness. Our platform is MATLAB 2016b on a Mac mini 2018 with a 3.2-GHz CPU and 16-GB RAM.

A. Datasets and Settings

Ten benchmark datasets are employed, which include seven image datasets [i.e., Yale,[1] Jaffe,[2] AR,[3] ORL,[4] COIL20,[5] binary alphabet (BA),[6] and CIFAR-10[7]] and three text corporas[8] (i.e., TR11, TR41, and TR45). Specifically, Yale, Jaffe, AR, and ORL are widely used face image datasets. BA consists of digits of letters of capital "A" through "Z" and "0" through "9." COIL20 is an object image dataset where the images of the object are taken at pose intervals of 5°. CIFAR-10 consisting of ten object classes, each of which contains 6000 color images. TR11, TR41, and TR45 are widely used text corporas datasets. Example images of the six-image dataset are illustrated in Fig. 2. More details of these datasets are given in [15] and [23]. Moreover, two synthetic datasets are employed, which consist of a two-moon dataset and a three-ring dataset, as shown in Figs. 3(a) and 4(a), respectively. The summaries of these datasets are listed in Table III.

For the CIFAR-10 dataset, we use a convolutional neural network (CNN) [48] pretrained on ILSVRC [49] as the feature extractor, which represents each image as a 2048-D vector. The data vectors are normalized to have a unit length. For each object class, we randomly select 100 samples for experiments.

Following [15], [21], and [23], 12 base kernels (i.e., $r = 12$) are constructed and these base kernels form a kernel pool by using MBKC. The kernel pool contains seven radial basis function (RBF) kernels $\boldsymbol{K}_{ij} = \exp(-\|x_i - x_j\|_2^2/(2\tau\sigma^2))$, where

[1]http://www.cvc.yale.edu/projects/yalefaces/yalefaces.html
[2]http://www.kasrl.org/jaffe.html
[3]http://www2.ece.ohio-state.edu/aleix/ARdataset.html
[4]https://www.cl.cam.ac.uk/research/dtg/attarchive/facedataset.html
[5]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[6]https://cs.nyu.edu/roweis/data/binaryalphadigs.mat
[7]https://www.cs.toronto.edu/kriz/cifar.html
[8]http://trec.nist.gov

TABLE III
Summary of the 12 Datasets

Dataset	Clusters (c)	Samples (n)	Features (d)	Type
Yale	15	165	1024 (32×32)	Image
Jaffe	10	213	676 (26×26)	Image
AR	10	213	768 (32×24)	Image
ORL	120	840	1024 (32×32)	Image
COIL	20	1440	1024 (32×32)	Image
BA	36	1404	320 (20×16)	Image
CIFAR-10	10	1000	1024(32×32)	Image
TR11	9	414	6429	Text
TR41	10	878	7454	Text
TR45	10	690	8261	Text
Two-moon	2	200	2	Synthetic
Three-ring	3	300	2	Synthetic

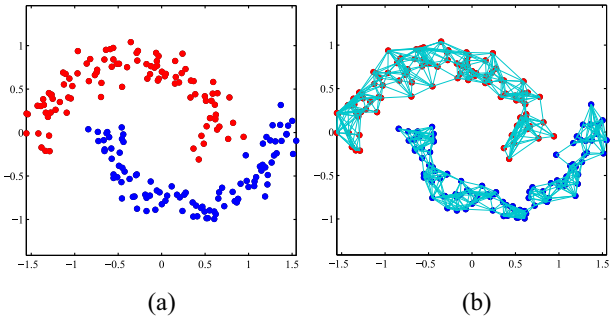


Fig. 3. Results on the two-moon synthetic data. (a) Original two-moon dataset. (b) Consensus affinity graph $\boldsymbol{Z}$.

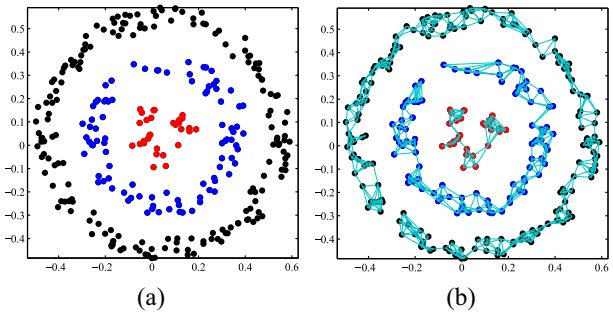


Fig. 4. Results on the three-ring synthetic data. (a) Original three-rings dataset. (b) Consensus affinity graph $\boldsymbol{Z}$.

τ varies in the range of $\{0.01, 0.05, 0.1, 1, 10, 50, 100\}$ and σ is the maximum distance between any two samples, four polynomial kernels $\boldsymbol{K}_{ij} = (a + x_i^{\mathrm{T}} x_j)^b$ with $a = \{0, 1\}$ and $b = \{2, 4\}$, and one cosine kernel $\boldsymbol{K}_{ij} = (x_i^{\mathrm{T}} x_j)/(\|x_i\| \cdot \|x_j\|)$. Finally, all the kernel Gram matrices are normalized to $[0, 1]$ range through $\boldsymbol{K}_{ij} = \boldsymbol{K}_{ij}/\sqrt{\boldsymbol{K}_{ii}\boldsymbol{K}_{jj}}$.

TABLE IV
PERFORMANCE OF THE COMPARED MKC METHODS ON TEN DATASETS. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD FONT

Dataset	Metric	MKKM	RMKKM	LKGr	SCMK	SMKL	JMKSC	LLMKL	MVC-LFA	Our CAGL
Yale	ACC	0.457(0.041)	0.521(0.034)	0.540(0.030)	0.582(0.025)	0.585(0.017)	0.630(0.006)	0.655(0.009)	0.618(0.011)	**0.703(0.000)**
	NMI	0.501(0.036)	0.556(0.025)	0.566(0.025)	0.576(0.012)	0.614(0.015)	0.631(0.006)	0.646(0.007)	0.609(0.009)	**0.668(0.000)**
	Purity	0.475(0.037)	0.536(0.031)	0.554(0.029)	0.610(0.014)	0.667(0.014)	0.673(0.007)	0.683(0.009)	0.624(0.010)	**0.709(0.000)**
Jaffe	ACC	0.746(0.069)	0.871(0.053)	0.861(0.052)	0.869(0.022)	0.967(0.000)	0.967(0.000)	**1.000(0.000)**	0.981(0.005)	**1.000(0.000)**
	NMI	0.798(0.058)	0.893(0.041)	0.869(0.031)	0.868(0.021)	0.951(0.000)	0.952(0.010)	**1.000(0.000)**	0.970(0.008)	**1.000(0.000)**
	Purity	0.768(0.062)	0.889(0.045)	0.859(0.038)	0.882(0.023)	0.967(0.000)	0.967(0.007)	**1.000(0.000)**	0.981(0.005)	**1.000(0.000)**
AR	ACC	0.286(0.014)	0.344(0.012)	0.314(0.015)	0.544(0.024)	0.465(0.012)	0.609(0.007)	0.853(0.013)	0.667(0.008)	**0.891(0.000)**
	NMI	0.592(0.014)	0.655(0.015)	0.648(0.007)	0.775(0.009)	0.681(0.014)	0.820(0.002)	0.935(0.003)	0.844(0.002)	**0.954(0.000)**
	Purity	0.305(0.012)	0.368(0.010)	0.330(0.014)	0.642(0.014)	0.611(0.011)	0.656(0.010)	0.897(0.003)	0.685(0.003)	**0.902(0.000)**
ORL	ACC	0.475(0.023)	0.556(0.024)	0.616(0.016)	0.656(0.015)	0.573(0.032)	0.725(0.014)	0.800(0.003)	0.692(0.005)	**0.863(0.000)**
	NMI	0.689(0.016)	0.748(0.018)	0.794(0.008)	0.808(0.008)	0.733(0.027)	0.852(0.012)	0.890(0.003)	0.836(0.003)	**0.932(0.000)**
	Purity	0.514(0.021)	0.602(0.024)	0.658(0.017)	0.699(0.015)	0.648(0.017)	0.753(0.012)	0.839(0.009)	0.732(0.004)	**0.878(0.000)**
COIL	ACC	0.548(0.058)	0.667(0.028)	0.618(0.051)	0.591(0.028)	0.487(0.031)	0.696(0.016)	0.636(0.010)	0.664(0.013)	**0.860(0.000)**
	NMI	0.707(0.033)	0.773(0.017)	0.766(0.023)	0.726(0.011)	0.628(0.018)	0.818(0.007)	0.806(0.004)	0.782(0.005)	**0.930(0.000)**
	Purity	0.590(0.053)	0.699(0.022)	0.650(0.039)	0.635(0.013)	0.683(0.004)	0.806(0.010)	0.714(0.010)	0.690(0.013)	**0.881(0.000)**
BA	ACC	0.405(0.019)	0.434(0.018)	0.444(0.018)	0.384(0.014)	0.246(0.012)	0.484(0.015)	0.482(0.010)	0.413(0.005)	**0.508(0.000)**
	NMI	0.569(0.008)	0.585(0.011)	0.604(0.009)	0.544(0.012)	0.486(0.011)	0.621(0.007)	0.619(0.007)	0.556(0.002)	**0.632(0.000)**
	Purity	0.435(0.014)	0.463(0.015)	0.479(0.017)	0.606(0.009)	0.623(0.011)	0.563(0.018)	0.593(0.009)	0.438(0.006)	**0.546(0.000)**
TR11	ACC	0.501(0.048)	0.577(0.094)	0.607(0.043)	0.549(0.015)	0.708(0.033)	0.737(0.002)	0.718(0.001)	0.572(0.026)	**0.761(0.000)**
	NMI	0.446(0.046)	0.561(0.118)	0.597(0.031)	0.371(0.018)	0.557(0.068)	0.673(0.002)	0.633(0.002)	0.582(0.012)	**0.689(0.000)**
	Purity	0.655(0.044)	0.729(0.096)	0.776(0.030)	0.783(0.011)	**0.835(0.048)**	0.819(0.001)	0.783(0.002)	0.768(0.009)	0.783(0.000)
TR41	ACC	0.561(0.068)	0.627(0.073)	0.595(0.020)	0.650(0.068)	0.671(0.002)	0.689(0.004)	0.689(0.004)	0.594(0.005)	**0.809(0.000)**
	NMI	0.578(0.042)	0.635(0.092)	0.604(0.023)	0.492(0.017)	0.625(0.004)	0.660(0.003)	0.666(0.003)	0.575(0.006)	**0.762(0.000)**
	Purity	0.728(0.042)	0.776(0.065)	0.759(0.031)	0.758(0.034)	0.761(0.003)	0.799(0.003)	0.817(0.003)	0.757(0.008)	**0.862(0.000)**
TR45	ACC	0.585(0.066)	0.640(0.071)	0.663(0.042)	0.634(0.058)	0.671(0.004)	0.687(0.036)	0.745(0.000)	0.721(0.002)	**0.800(0.000)**
	NMI	0.562(0.056)	0.627(0.092)	0.671(0.020)	0.584(0.051)	0.622(0.007)	0.690(0.022)	0.726(0.000)	0.681(0.001)	**0.768(0.000)**
	Purity	0.691(0.058)	0.752(0.074)	0.800(0.026)	0.728(0.048)	0.816(0.004)	0.822(0.031)	0.797(0.000)	0.806(0.001)	**0.857(0.000)**
Deep CIAR-10	ACC	0.499(0.014)	0.638(0.014)	0.659(0.012)	0.727(0.008)	0.712(0.005)	0.743(0.006)	0.760(0.002)	0.698(0.002)	**0.788(0.000)**
	NMI	0.467(0.010)	0.615(0.012)	0.626(0.008)	0.633(0.005)	0.611(0.002)	0.685(0.005)	0.646(0.002)	0.583(0.001)	**0.699(0.000)**
	Purity	0.691(0.008)	0.654(0.010)	0.677(0.011)	0.734(0.004)	0.714(0.003)	0.746(0.006)	0.760(0.001)	0.698(0.002)	**0.788(0.000)**
AVG.	ACC	0.506 (0.042)	0.588(0.042)	0.592(0.030)	0.619(0.028)	0.609(0.015)	0.697(0.011)	0.734(0.005)	0.660(0.008)	**0.798(0.000)**
	NMI	0.591 (0.032)	0.665(0.044)	0.675(0.019)	0.638(0.016)	0.651(0.017)	0.740(0.008)	0.757(0.003)	0.702(0.005)	**0.803(0.000)**
	Purity	0.585 (0.035)	0.647(0.039)	0.653(0.025)	0.708(0.019)	0.733(0.012)	0.760(0.011)	0.790(0.005)	0.718(0.006)	**0.821(0.000)**
Ranking		#9	#8	#7	#5	#6	#3	#2	#4	#1

* The average clustering performance on each separate dataset is denoted as "AVG." for short. "Ranking" indicates the performance rankings of the compared methods.

B. Compared Methods and Evaluation Metrics

We compare the clustering performance of the proposed CAGL method against eight state-of-the-art MKC methods, including MKKM [34], RMKKM [15], AASC [36], SCMK [19], LKGr [20], SMKL [14], JMKSC [21], LLMKL [23], and MVC-LFA [35]. MVC-LFA is not a graph learning method, and we take the kernels as views and fed into it. Besides, we do not compare with Neighbor-Kernel MKC [22] and GLSAMKC [18], as their source codes are not publicly available. To be fair, the parameters of these competitors are carefully tuned by following the recommended experimental settings provided by their respective authors.

Three widely used clustering metrics, accuracy (ACC), normalized mutual information (NMI), and purity, are applied. For these metrics, the larger value indicates the better clustering performance (referring to [1] and [31] for more details). To avoid the effect of randomness caused by the k-means algorithm, each experiment is repeated 20 times and the average clustering performance with standard deviation is reported.

C. Experimental Results

The sufficient experimental results of all the compared methods are presented in Table IV. It can be seen that our CAGL consistently obtains the best clustering performance. More precisely, CAGL improves by 6.4%, 4.6%, and 3.1%, respectively, compared to LLMKL [23] (the best comparison method) in terms of average ACC, NMI, and purity. Note that the performance ranking of these compared methods is shown in the last row of Table IV. For a better flow of this article, the detailed discussions are given in Section V-G.

D. Consensus Affinity Graph

To evaluate the quality of the consensus affinity matrix/graph produced by DCGL in a visual manner, we evaluate the resulting $\boldsymbol{Z}$ in the perspective of matrix and graph on the Jaffe dataset and the two synthetic datasets successively.

Taking the Jaffe dataset, for instance, which consists of ten clusters, and therefore ideally has ten diagonal blocks. As shown in Fig. 5, the matrix $\boldsymbol{Z}$ produced by our CAGL has better block-diagonal property and intercluster separability than all other MKGC methods.

From a graph perspective, the graph $\boldsymbol{Z}$ should have c connected subgraph in the ideal situation. Therefore, we also evaluate CAGL on the two-moon and three-ring datasets, as shown in Figs. 3(b) and 4(b), respectively. It can be seen that there is no edge between any two clusters, and all data points within the same cluster are connected together (i.e., each strongly connected subgraph corresponds to a cluster). Thus, as a kernel method, the proposed CAGL method can handle well the data with nonlinearity. Significantly, it can preserve the true local structure and obtain the desired numbers of graph connected components. Due to the exact c connected

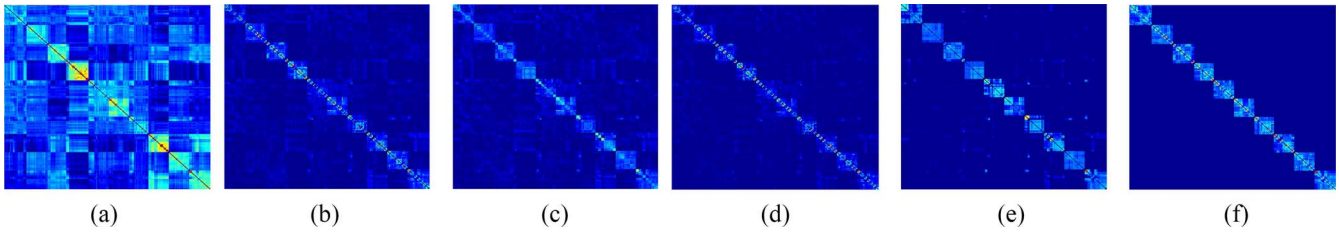


Fig. 5. Visualization of the resulting consensus affinity matrices produced by different MKGC methods on the Jaffe dataset. The Jaffe dataset consists of ten clusters. The darker the blue color, the value is closer to 0. (a) LKGr. (b) SMKL. (c) SCMK. (d) JMKSC. (e) LLMKL. (f) Our CAGL.

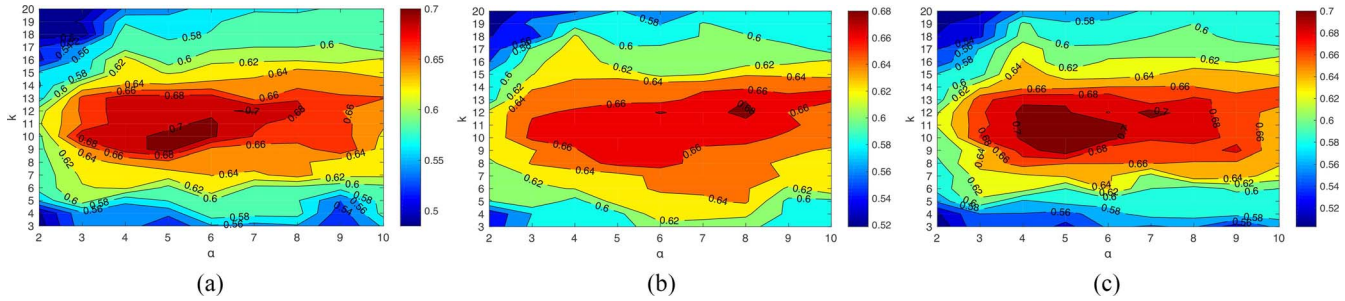


Fig. 6. Clustering performance (in terms of ACC, NMI, and purity) of our CAGL w.r.t. parameters α and k on the Yale dataset, where parameter α balances the effect of the proposed KALSL to preserve the local manifold structure of the input data in kernel space $\mathcal{H}$ and parameter k controls the number of neighbors when constructing the sparse consensus affinity graph for the clustering purpose (zoom in for best view). (a) ACC. (b) NMI. (c) Purity.

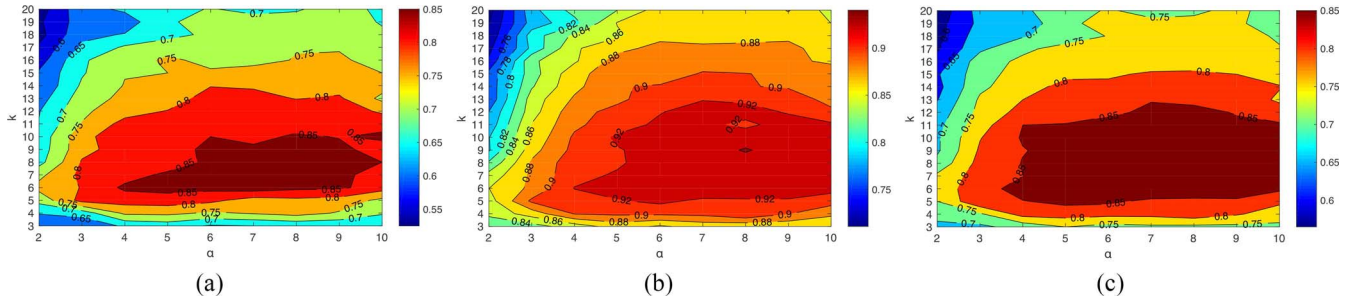


Fig. 7. Clustering performance (in terms of ACC, NMI, and purity) of CAGL w.r.t. parameters α and k on the ORL dataset. The notes are the same to Fig. 6. (a) ACC. (b) NMI. (c) Purity.

components, CAGL has a stable output, that is, the standard deviations of the clustering results are 0.

E. Parameter Sensitivity

The proposed CAGL method involves two essential parameters, α and k, which need to be tuned. α is the regularization parameter of the involved KALSL term [i.e., $-\mathrm{Tr}(\boldsymbol{KZ})$] and k is the threshold parameter to control the number of nearest neighbors of the ith graph vertex. For simplicity, both the regularization parameter β and the self-tuned parameter λ can be initiated as 1. By using a grid search strategy, the searching regions of α and k are selected from $\{2, \ldots, 10\}$ and $\{3, \ldots, 20\}$, respectively.

Taking the Yale and ORL datasets for example, we show the sensitivity of clustering performance to different α and k values in Figs. 6 and 7, respectively. It can be observed that CAGL works well for a wide range of α and k values.

F. Convergence Study and Computational Cost

The computational cost results of all the compared MKGC methods evaluated on the Yale, ORL, TR11, TR45, and CIAR-10 datasets are shown in Table V. We can see that the computational cost of our CAGL is lower than RMKKM, LKGr, SCMK, SMKL, JMKSC, and LLMKL, whose clustering performance is also worse than that of ours. Moreover, the cost of MVC-LFA is less than that of ours, but its clustering performance is poor.

TABLE V
COMPUTATIONAL TIME (IN SECONDS) COMPARISON

Method	Yale	ORL	TR11	TR45	CIAR-10
RMKKM	0.98(0.01)	3.82(0.08)	4.14(0.12)	6.93(0.12)	42.56(1.89)
LKGr	1.82(0.03)	7.54(0.12)	13.72(0.11)	7.56(0.08)	80.89(3.38)
SCMK	4.59(0.05)	43.13(1.14)	50.73(1.14)	208.82(5.44)	93.67(4.12)
SMKL	1.54(0.02)	13.73(0.57)	9.86(0.17)	154.66(4.32)	109.78(5.38)
JMKSC	0.73(0.01)	2.94(0.10)	3.84(0.11)	8.57(0.12)	16.87(1.27)
LLMKL	1.09(0.02)	2.85(0.02)	3.60(0.10)	7.99(0.22)	15.73(0.61)
MVC-LFA	0.25(0.00)	0.89(0.01)	1.12(0.44)	2.24(0.04)	6.95(0.42)
CAGL	0.25(0.00)	1.54(0.03)	2.34(0.03)	4.96(0.17)	15.83(0.55)

Furthermore, we investigate the convergence analysis of the proposed method by reporting the objective function value and the corresponding clustering ACC with increasing iteration. The results evaluated on the Yale, Jaffe, ORL, and TR41 datasets are shown in Fig. 8. From each convergence curve, it

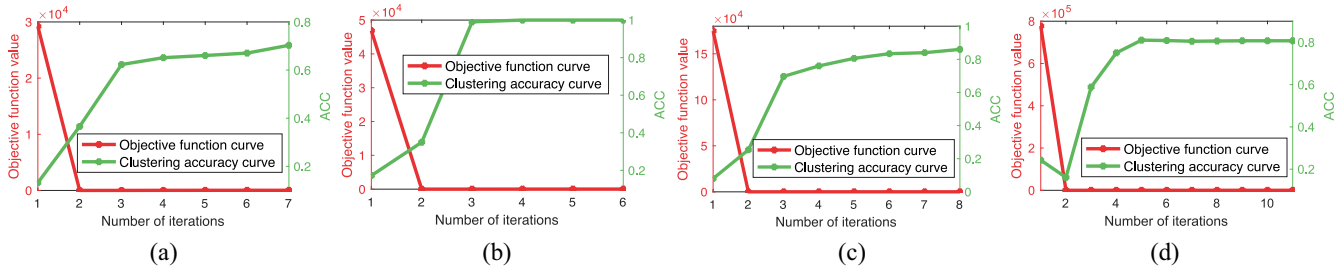


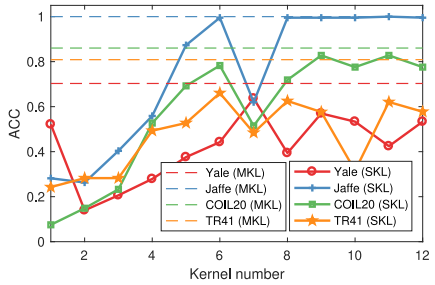
Fig. 8. Convergence property of CAGL on the (a) Yale, (b) Jaffe, (c) ORL, and (d) TR41 datasets. Note that the x-axis denotes the number of iterations, while the y-left-axis denotes the objective function value, and the y-right-axis denotes the clustering ACC.

is easy to see that the objective value is monotonically decreasing close to 0 after only a few iterations, and this phenomenon is consistent with the convergence analysis in Section IV-D. However, Algorithm 1 will continue to work until obtaining the desired c block-diagonal components within about 15 iterations. Especially, it spends 7, 6, 8, and 11 iterators for the four datasets, respectively. This validates that CAGL has very strong and stable convergence behavior.

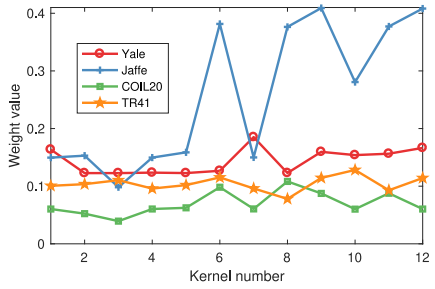
G. Discussion

Here, we discuss the above experimental results and the effects of the involved terms of the proposed CAGL method.

1) According to Table IV, we have the following. Overall, these data suggest that CAGL consistently obtains the best performance in all cases, which shows the effectiveness of CAGL in dealing with nonlinear clustering tasks. Moreover, the standard deviations of CAGL are 0, since the learned affinity graph $\boldsymbol{Z}$ has exact c connected components by self-tuning parameter γ. But, the competitors cannot satisfy exact c connected components due to fixed parameter γ.
2) According to Figs. 6 and 7, we have the following. These results suggest that CAGL works well for a wide range of the parameters α and k, which illustrate that CAGL is easy to be tamed. Furthermore, it can be seen that different k will produce different performance. This means that the top-k neighbors sparse strategy is indeed effective. For simplicity, k can be fixed to (n/c), and hence only α needs to be tuned. Importantly, CAGL contains a twin term [i.e., $\mathrm{Tr}(\boldsymbol{K}+\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Z})-\alpha\,\mathrm{Tr}(\boldsymbol{K}\boldsymbol{Z})$]. If we fix $\alpha = 2$, the twin term can be rewritten as $\mathrm{Tr}(\boldsymbol{K}-2\boldsymbol{K}\boldsymbol{Z}+\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{Z})$, which is the main part of problem (2). From Figs. 6 and 7, it is obvious that the best clustering performance is obtained when $\alpha > 2$ rather than $\alpha = 2$; therefore, the proposed KALSL term, $-\mathrm{Tr}(\boldsymbol{K}\boldsymbol{Z})$, is proven to be effective.
3) According to Figs. 3(b), 4(b), and 5 we have the following. Obviously, CAGL can produce a consensus affinity matrix $\boldsymbol{Z}$ with better block-diagonal property and inter-cluster separability than the compared methods. The main reasons are that: a) the constraint $\mathrm{rank}(\boldsymbol{L}_Z) = n-c$ forces $\boldsymbol{Z}$ to hold c diagonal blocks and b) only the first k important edges are preserved when solving $\boldsymbol{Z}$, this can remove the redundant graph edges between two graph vertexes. Especially Figs. 3(b) and 4(b), the affinity graphs demonstrate that CAGL can better explore the true cluster structures of different clusters. Thus, another important reason is that the KALSL term [i.e., $-\mathrm{Tr}(\boldsymbol{K}\boldsymbol{Z})$] can preserve the local manifold structure of data in the kernel space $\mathcal{H}$.
4) According to Fig. 8 and Table V, we have the following. Theoretically and experimentally, our algorithm has very strong and stable convergence behavior. It should be noted that the convergence criterion is $\mathrm{rank}(\boldsymbol{L}_Z) = n-c$ rather than the objective function residual. Therefore, the objective value quickly reduces close to 0 after only a few iterations, and then it will continue working until the desired numbers of connected components are obtained. In addition, the ACC curves indicate that the better the block-diagonal property, the higher the clustering accuracy. Thus, the term $\mathrm{Tr}(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{L}_Z\boldsymbol{F})$ and self-tuned strategy to λ are proven to be effective. In terms of running time, Table V demonstrates the time advantage of the proposed thin model than traditional fat models. The reasons are that: a) the top-k strategy reduces the computation; b) our algorithm has strong convergence; and c) the complexity is only $\mathcal{O}(n^2)$ in each iteration.
5) In order to further demonstrate the effectiveness of the autoweighted candidate graph fusion model, the clustering performance of SKL learning (i.e., CAGL-I) and their corresponding weight values are evaluated. Note that the results of SKL and MKL are produced by (7) and (10), respectively. As shown in Fig. 9, it is easy to know that: a) the clustering performance of SKL is worse than that of MKL's; b) the better the clustering performance of a kernel, the larger its weight value; and c) the most suitable kernel and associated parameters for a specific dataset are usually different. This result, again, demonstrates the effectiveness of the fusion model as well as the proposed method.
6) CAGL has a higher memory usage than its counterparts, blaming on storing extra r candidate affinity graphs. Fortunately, the number of base kernels, $r = 12$, is very small. In summary, the local manifold structure of the nonlinear data in kernel space is preserved and the goal of MKL is changed to multiple graph learning (see the second contribution that is summarized in Section I), thus CAGL can learn a high-quality affinity graph for clustering.



(a)



(b)

Fig. 9. Effect of the proposed autoweighted fusion model on four datasets. The solid lines stand for the SKL, while dashed lines stand for MKL. (a) SKL versus MKL. (b) Kernel weights.

VI. Conclusion

In this article, we proposed a new MKGC method called CAGL. Different from the existing MKGC methods, which learn a consensus kernel matrix and an affinity graph using a fat model, CAGL learns a consensus affinity graph directly. Accordingly, the goal was shifted from MKL to multiple graph learning, which is the first to the best of our knowledge. Sufficient experiments on ten datasets and two synthetic datasets demonstrated the effectiveness and efficiency of the proposed method. In the future study, we expect to devote our efforts to address the problem where some rows and columns of one or more base kernels are absent [16].

Appendix

Problem (16) can be dealt with individually for each different i using the following problem:

$$\min_{z_{ij}\geq 0, z_i^{\mathrm{T}}\mathbf{1}=1} \sum_{j=1}^{n}\sum_{m=1}^{r} w_m\left(z_{ij}-z_{ij}^m\right)^2+\gamma\sum_{j=1}^{n} f_{ij}z_{ij} \tag{39}$$

where $f_{ij}=\|\boldsymbol{p}_i-\boldsymbol{p}_j\|_2^2$. Then, (39) can be rewritten as follows:

$$\min_{z_{ij}\geq 0, z_i^{\mathrm{T}}\mathbf{1}=1} \sum_{j=1}^{n}\sum_{m=1}^{r} w_m\left(z_{ij}-z_{ij}^m\right)^2+\gamma\sum_{j=1}^{n}\sum_{m=1}^{r}\frac{1}{r} f_{ij}z_{ij} \tag{40}$$

$$\min_{z_{ij}\geq 0, z_i^{\mathrm{T}}\mathbf{1}=1} \sum_{m=1}^{r}\sum_{j=1}^{n}\left[\left(z_{ij}-z_{ij}^m\right)^2+2\frac{\gamma f_{ij}}{2rw_m}\left(z_{ij}-z_{ij}^m+z_{ij}^m\right)\right] \tag{41}$$

$$\min_{z_{ij}\geq 0, z_i^{\mathrm{T}}\mathbf{1}=1} \sum_{m=1}^{r}\sum_{j=1}^{n}\left[\left(z_{ij}-z_{ij}^m+\frac{\gamma}{2rw_m}f_{ij}\right)^2+\Delta\right] \tag{42}$$

where $\Delta=[(\lambda f_{ij}z_{ij}^m)/(rw_m)]-(\gamma f_{ij}/2rw_m)^2$ and can be seen as a constant. For ease of exploration, we denote $\boldsymbol{z}_i^m$ and $\boldsymbol{f}_i$ as the vectors with the jth entries as z_{ij}^m and f_{ij}, respectively. Thus, (42) can be rewritten in a vector shape as follows:

$$\min_{z_{ij}\geq 0, z_i^{\mathrm{T}}\mathbf{1}=1} \sum_{m=1}^{r}\left\|\boldsymbol{z}_i-\boldsymbol{z}_i^m+\frac{\lambda}{2rw_m}\boldsymbol{f}_i\right\|_2^2. \tag{43}$$

This completes the proof. ■

References

[1] K. Zhan, F. Nie, J. Wang, and Y. Yang, "Multiview consensus graph clustering," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1261–1270, Mar. 2019.

[2] K. Zhan, C. Niu, C. Chen, F. Nie, C. Zhang, and Y. Yang, "Graph structure fusion for multiview clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 1984–1993, Oct. 2019.

[3] F. Nie, X. Wang, C. Deng, and H. Huang, "Learning a structured optimal bipartite graph for co-clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4129–4138.

[4] Z. Kang *et al.*, "Multi-graph fusion for multi-view spectral clustering," *Knowl. Based Syst.*, vol. 189, Feb. 2020, Art. no. 105102.

[5] Z. Kang, H. Xu, B. Wang, H. Zhu, and Z. Xu, "Clustering with similarity preserving," *Neurocomputing*, vol. 365, pp. 211–218, Nov. 2019.

[6] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6323–6332, Dec. 2018.

[7] C.-D. Wang, J.-H. Lai, and P. S. Yu, "Multi-view clustering based on belief propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1007–1021, Apr. 2015.

[8] Y. Pang, J. Xie, F. Nie, and X. Li, "Spectral clustering by joint spectral embedding and spectral rotation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 247–258, Jan. 2020.

[9] J. Huang, F. Nie, and H. Huang, "A new simplex sparse learning model to measure data similarity for clustering," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3569–3575.

[10] C. Lu, J. Feng, Z. Lin, T. Mei, and S. Yan, "Subspace clustering by block diagonal representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 487–501, Feb. 2019.

[11] B.-Y. Liu, L. Huang, C.-D. Wang, S. Fan, and P. S. Yu, "Adaptively weighted multiview proximity learning for clustering," *IEEE Trans. Cybern.*, early access, Dec. 16, 2019, doi: 10.1016/j.neunet.2020.05.030.

[12] Y.-M. Xu, C.-D. Wang, and J.-H. Lai, "Weighted multi-view clustering with feature selection," *Pattern Recognit.*, vol. 53, pp. 25–35, May 2016.

[13] X. Xie, X. Guo, G. Liu, and J. Wang, "Implicit block diagonal low-rank representation," *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 477–489, Jan. 2018.

[14] Z. Kang, X. Lu, J. Yi, and Z. Xu, "Self-weighted multiple kernel learning for graph-based clustering and semi-supervised classification," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2312–2318.

[15] L. Du *et al.*, "Robust multiple kernel *k*-means using L21-norm," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3476–3482.

[16] X. Liu *et al.*, "Multiple kernel *k*-means with incomplete kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 5, pp. 1191–1204, May 2020.

[17] M. Li, X. Liu, L. Wang, Y. Dou, J. Yin, and E. Zhu, "Multiple kernel clustering with local kernel alignment maximization," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1704–1710.

[18] C. Wang, E. Zhu, X. Liu, L. Gao, J. Yin, and N. Hu, "Multiple kernel clustering with global and local structure alignment," *IEEE Access*, vol. 6, pp. 77911–77920, 2018.

[19] Z. Kang, C. Peng, Q. Cheng, and Z. Xu, "Unified spectral clustering with optimal graph," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3366–3373.

[20] Z. Kang, L. Wen, W. Chen, and Z. Xu, "Low-rank kernel learning for graph-based clustering," *Knowl. Based Syst.*, vol. 163, pp. 510–517, Jan. 2019.

[21] C. Yang, Z. Ren, Q. Sun, M. Wu, M. Yin, and Y. Sun, "Joint correntropy metric weighting and block diagonal regularizer for robust multiple kernel subspace clustering," *Inf. Sci.*, vol. 500, pp. 48–66, May 2019.

[22] S. Zhou *et al.*, "Multiple kernel clustering with neighbor-kernel subspace segmentation," *IEEE Trans. Neural Netw.*, vol. 31, no. 4, pp. 1351–1362, Apr. 2020.

[23] Z. Ren, H. Li, C. Yang, and Q. Sun, "Multiple kernel subspace clustering with local structural graph and low-rank consensus kernel learning," *Knowl. Based Syst.*, vol. 188, Jan. 2020, Art. no. 105040.
[24] Z. Ren and Q. Sun, "Simultaneous global and local graph structure preserving for multiple kernel clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 4, 2020, doi: 101109/TNNLS.2020.2991366.
[25] C. Zhang *et al.*, "Generalized latent multi-view subspace clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 1, pp. 86–99, Jan. 2020.
[26] M.-S. Chen, L. Huang, C.-D. Wang, and D. Huang, "Multi-view clustering in latent embedding space," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3513–3520.
[27] X. Zhu, S. Zhang, Y. Li, J. Zhang, L. Yang, and Y. Fang, "Low-rank sparse subspace for spectral clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1532–1543, Jul. 2019.
[28] Z. Ren, Q. Sun, B. Wu, X. Zhang, and W. Yan, "Learning latent low-rank and sparse embedding for robust image feature extraction," *IEEE Trans. Image Process.*, vol. 29, pp. 2094–2107, Jan. 2020.
[29] F. Nie, W. Zhu, and X. Li, "Unsupervised feature selection with structured graph optimization," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1302–1308.
[30] J. Wen, B. Zhang, Y. Xu, J. Yang, and N. Han, "Adaptive weighted nonnegative low-rank representation," *Pattern Recognit.*, vol. 81, pp. 326–340, Sep. 2018.
[31] Z. Kang, H. Pan, S. C. H. Hoi, and Z. Xu, "Robust graph learning from noisy data," *IEEE Trans. Cybern.*, vol. 28, no. 4, pp. 1007–1021, May 2020.
[32] X. Li, M. Chen, and Q. Wang, "Adaptive consistency propagation method for graph clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 797–802, Apr. 2020.
[33] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 977–986.
[34] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Multiple kernel fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 1, pp. 120–134, Feb. 2012.
[35] S. Wang *et al.*, "Multi-view clustering via late fusion alignment maximization," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3778–3784.
[36] H.-C. Huang, Y.-Y. Chuang, and C.-S. Chen, "Affinity aggregation for spectral clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 773–780.
[37] X. Liu, L. Wang, J. Zhang, J. Yin, and H. Liu, "Global and local structure preservation for feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1083–1095, Jun. 2013.
[38] Y. Li, C. Jia, X. Kong, L. Yang, and J. Yu, "Locally weighted fusion of structural and attribute information in graph clustering," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 247–260, Jan. 2019.
[39] C. Tang *et al.*, "Robust unsupervised feature selection via dual self-representation and manifold regularization," *Knowl. Based Syst.*, vol. 145, pp. 109–120, Apr. 2018.
[40] Z. Kang, X. Lu, Y. Lu, C. Peng, W. Chen, and Z. Xu, "Structure learning with similarity preserving," *Neural Netw.*, vol. 129, pp. 138–148, Sep. 2020, doi: 10.1016/j.neunet.2020.05.030.
[41] R. Wang, F. Nie, Z. Wang, H. Hu, and X. Li, "Parameter-free weighted multi-view projected clustering with structured graph learning," *IEEE Trans. Knowl. Data Eng.*, early access, Apr. 26, 2019, doi: 10.1109/TKDE.2019.2913377.
[42] F. Nie, J. Li, and X. Li, "Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1881–1887.
[43] M. Yin, J. Gao, Z. Lin, Q. Shi, and Y. Guo, "Dual graph regularized latent low-rank representation for subspace clustering," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4918–4933, Dec. 2015.
[44] J. Wen, Y. Xu, and H. Liu, "Incomplete multiview spectral clustering with adaptive graph learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 4, pp. 1418–1429, Apr. 2020.
[45] U. Von Luxburg, M. Belkin, and O. Bousquet, "Consistency of spectral clustering," *Ann. Stat.*, vol. 36, no. 2, pp. 555–586, 2008.
[46] M. Wei, J. Huang, X. Xie, L. Liu, J. Wang, and J. Qin, "Mesh denoising guided by patch normal co-filtering via kernel low-rank recovery," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 10, pp. 2910–2926, Oct. 2019.
[47] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint L_{21}-norms minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1813–1821.
[48] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[49] O. Russakovsky *et al.*, "ImageNet large-scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

Zhenwen Ren (Member, IEEE) received the B.Sc. and M.Sc. degrees in communication and information systems from the Southwest University of Science and Technology (SWUST), Mianyang, China, in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree in control science and engineering with the Nanjing University of Science and Technology, Nanjing, China.

He is working with the School of National Defence Science and Technology, SWUST. He has published 20+ peer-reviewed papers, including those in highly regarded journals and conferences, such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS. His research interests include wireless-sensor networks, computer vision, machine learning, deep learning, and industrial software.

Simon X. Yang (Senior Member, IEEE) received the B.Sc. degree in engineering physics from Beijing University, Beijing, China, in 1987, the first of two M.Sc. degrees in biophysics from the Chinese Academy of Sciences, Beijing, in 1990, the second M.Sc. degree in electrical engineering from the University of Houston, Houston, TX, USA, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1999.

He is currently a Professor and the Head of the Advanced Robotics and Intelligent Systems Laboratory, University of Guelph, Guelph, ON, Canada. His research interests include robotics, intelligent systems, sensors and multisensor fusion, wireless-sensor networks, control systems, machine learning, fuzzy systems, and computational neuroscience.

Prof. Yang has been very active in professional activities. He serves as the Editor-in-Chief for the *International Journal of Robotics and Automation*, and an Associate Editor for the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS OF ARTIFICIAL INTELLIGENCE, and several other journals. He has been involved in the organization of many international conferences.

Quansen Sun (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology (NJUST), Nanjing, China, in 2006.

He is a Professor with the Department of Computer Science, NJUST. He visited the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, in 2004 and 2005, respectively. His current interests include pattern recognition, image processing, remote sensing information systems, and medicine image analysis.

Tao Wang (Member, IEEE) received the B.E. degree in computer science and the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology (NUST), Nanjing, China, in 2012 and 2017, respectively.

He is currently an Associate Professor with the School of Computer Science and Engineering, NUST. His research interests include image segmentation and pattern recognition.