Fast Multi-View Clustering Via Ensembles: Towards Scalability, Superiority, and Simplicity

Dong Huang ⓘ, *Member, IEEE*, Chang-Dong Wang ⓘ, *Member, IEEE*, and Jian-Huang Lai ⓘ, *Senior Member, IEEE*

Abstract—**Despite significant progress, there remain three limitations to the previous multi-view clustering algorithms. First, they often suffer from high computational complexity, restricting their feasibility for large-scale datasets. Second, they typically fuse multi-view information via one-stage fusion, neglecting the possibilities in multi-stage fusions. Third, dataset-specific hyperparameter-tuning is frequently required, further undermining their practicability. In light of this, we propose a *fast multi-view clustering via ensembles* (FastMICE) approach. Particularly, the concept of random view groups is presented to capture the versatile view-wise relationships, through which the hybrid early-late fusion strategy is designed to enable efficient multi-stage fusions. With *multiple* views extended to *many* view groups, three levels of diversity (w.r.t. features, anchors, and neighbors, respectively) are jointly leveraged for constructing the view-sharing bipartite graphs in the early-stage fusion. Then, a set of diversified base clusterings for different view groups are obtained via fast graph partitioning, which are further formulated into a unified bipartite graph for final clustering in the late-stage fusion. Notably, FastMICE has almost linear time and space complexity, and is free of dataset-specific tuning. Experiments on 22 multi-view datasets demonstrate its advantages in scalability (for extremely large datasets), superiority (in clustering performance), and simplicity (to be applied) over the state-of-the-art. Code available: https://github.com/huangdonghere/FastMICE.**

Index Terms—**Data clustering, ensemble clustering, hybrid early-late fusion, large-scale clustering, linear time, multi-view clustering.**

I. INTRODUCTION

CLUSTERING analysis has been a fundamental yet challenging research topic in knowledge discovery and data mining [1]. It aims to partition a set of data samples into a certain number of homogeneous groups, each of which is referred to as a cluster. Among the various sub-topics in clustering analysis, multi-view clustering (MVC) has recently gained a considerable amount of attention due to its advantage in fusing common and complementary information from multiple views (or data sources) to enhance the clustering performance [2]. Despite the proposals of many MVC algorithms, there are still three crucial questions (with regard to scalability, information fusion, and hyperparameter tuning, respectively) that remain to be addressed.

First, how to enable MVC for very large-scale datasets? Though a large quantity of MVC algorithms have been developed in recent years, the high computational complexity is still a major hurdle for many of them to be applied in large-scale scenarios. In previous MVC works, there are several often-encountered complexity bottlenecks, such as affinity graph construction, graph partitioning, and some other expensive matrix computations. The affinity graph construction is a basic step in many MVC algorithms, which formulates the sample-wise relationship by computing an $N \times N$ affinity matrix and generally takes $O(N^2 d)$ time and $O(N^2)$ space, where N is the number of samples and d is the dimension. The graph partitioning (typically by spectral clustering) is another computationally expensive step in many MVC algorithms [3], [4], [5], [6], [7], [8], which often requires singular value decomposition (SVD) and takes $O(N^3)$ time and $O(N^2)$ space. Especially, the graph partitioning via spectral clustering is adopted as an important step in many MVC algorithms, such as multi-view spectral clustering [3], [8], multi-view subspace clustering [6], [9], and multi-view graph learning [4], [5], which, together with some other expensive matrix computations, contributes to the $O(N^3)$ complexity bottleneck in these MVC algorithms [3], [4], [5], [6], [7], [8].

Second, in which stage should multi-view information be fused? An essential task of MVC is to fuse the information from multiple views for robust clustering. The difference in various MVC algorithms is typically reflected by *in which stage* and *how* they conduct the multi-view fusion. A naive strategy is to directly concatenate the features from multiple views and then perform some single-view clustering algorithm on the concatenated features, which in practice is rarely adopted as it neglects the rich and complementary information across multiple views. In the MVC literature, the most widely-adopted strategy may be the *early fusion* [3], [5], [8], [9], which typically fuse the multi-view information in a unified clustering model via optimization or some heuristics (as illustrated in Fig. 1(a)). Besides the early fusion, another popular strategy is the *late fusion* [6], [10], [11],

Dong Huang is with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China, and also with the Key Laboratory of Smart Agricultural Technology in Tropical South China, Ministry of Agriculture and Rural Affairs, China (e-mail: huangdonghere@gmail.com).

Chang-Dong Wang and Jian-Huang Lai are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China, also with the Guangdong Key Laboratory of Information Security Technology, Guangzhou 510006, China, and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China (e-mail: changdongwang@hotmail.com; stsljh@mail.sysu.edu.cn).

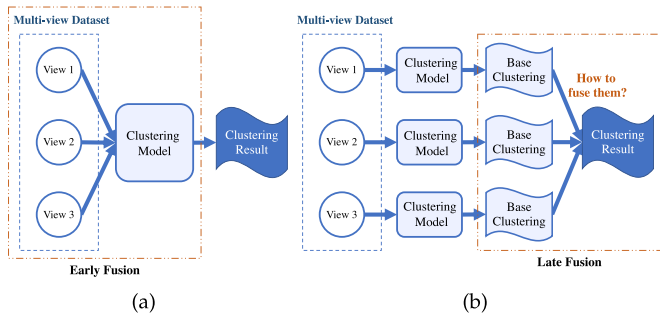Digital Object Identifier 10.1109/TKDE.2023.3236698

Fig. 1. In which stage can the multi-view information be fused? Most existing MVC works adopt either (a) early fusion or (b) late fusion (as illustrated on an example of three views).

Fig. 2. Illustration of the hybrid early-late fusion strategy on an example of three views. With the benefits brought in by the random view groups, the key questions arise as to how to diversify them, how to fuse them, and especially how to ensure the clustering robustness while maintaining the scalability for extremely large datasets.

[12], which first obtains multiple base clusterings by performing the clustering process on each view separately and then fuses these base clusterings into a more robust clustering result in the final stage (as illustrated in Fig. 1(b)). While most of the existing MVC algorithms adopt either early fusion or late fusion, it is surprising that few of them have gone beyond the single-stage fusion to explore the rich possibilities and potential benefits hidden in the multi-stage fusion formulation.

Third, is the dataset-specific fine-tuning necessary? The regularization hyperparameters or some other types of hyperparameters are often involved in previous MVC algorithms to adjust the influences of different terms (or components) [2], where dataset-specific fine-tuning is frequently required to seek the proper values of these hyperparameters in a probably extensive trial-and-error manner. However, unlike supervised or semi-supervised learning [13], [14], in the unsupervised situations it may be arguable whether the ground-truth labels can be used for guiding the fine-tuning process. Without the fine-tuning guided by partial or even all ground-truth labels, the practicality of these MVC algorithms may be significantly weakened. Moreover, when the number of the tuning-intensive hyperparameters goes to three or even more, their tuning costs (typically via grid-search) might become very expensive on large datasets (as reported in Table VII), which give rise to the critical question whether the dataset-specific tuning can be eliminated while maintaining robust clustering performance.

More recently several efforts have been made to deal with some of the above issues. In single-view clustering, it has proved to be an effective strategy to represent large-scale data samples via a set of anchors (also known as landmarks or representatives) [15], [16], which can substantially facilitate the computation of the graph construction and partitioning for large-scale datasets. When it goes from single-view to multi-view, the anchor-based formulation still shows its promising ability [17], [18], [19], [20], but also faces a series of new challenges, ranging from multi-view anchor selection to multi-view information fusion. Typically, Li et al. [17] selected a set of anchors by performing k-means on the concatenated multi-view features. With this unified anchor set, a bipartite graph is built between data samples and anchors on each view, and then multiple bipartite graphs are fused into a consensus graph for the final
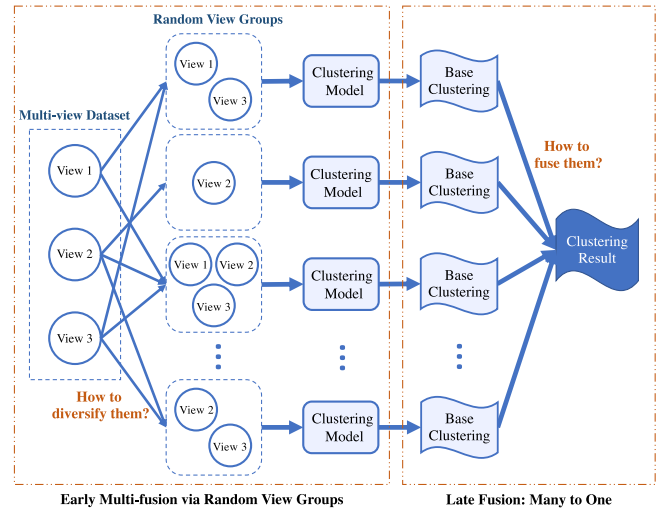
clustering [17]. Wang et al. [20] exploited the self-expressive loss to learn a set of unified anchors and a bipartite graph for all views. However, considering the diverse characteristics of multiple views, a single set of unified anchors may not sufficiently capture the rich and complementary information of all views. Different from the pursuit of a single anchor set for *all* views [17], [20], Kang et al. [18] learned a set of anchors on *each* view, and then built a bipartite graph on each view separately. Then the multiple bipartite graphs are combined into a unified graph for final clustering. However, since each anchor set on a view is learned independently of other views, the cross-view information is inherently neglected in the construction of each single-view graph, which may lead to a degraded capacity of multi-view expressiveness. Despite the progress, these methods [17], [18], [19], [20] either jointly construct a *single* anchor set *for all* views [17], [19], [20] or separately construct a *single* anchor set *for each* view [18], which, however, neglect the possibilities hidden between *single* and *all*, and dwell in the single-stage fusion strategy without the ability to capture the view-wise relationships in multiple stages. Furthermore, the requirement of dataset-specific hyperparameter-tuning in many of them [17], [18], [19], [21] also poses a practical hurdle for their real-world applications.

To jointly address the above-mentioned issues, in this paper, we propose a *fast m*ulti-v*iew c*lustering via *e*nsembles (Fast-MICE) approach. Different from previous MVC approaches that tend to work at either single views or all views in each stage, this paper first presents the concept of random view groups, which serves as the basic form of our flexible view-organizations. Specifically, with each view group consisting of a random number of view members, the *multiple* views can be extended to *many* random view groups for investigating the view-wise relationships in a diversified manner. Based on random view groups, a hybrid early-late fusion strategy is devised to enable efficient and robust fusions at multiple stages (as shown

in Fig. 2). In the early stage, multiple fusions are simultaneously performed in multiple view groups, where three levels of diversity, namely, feature-level diversity, anchor-level diversity, and neighborhood-level diversity, are jointly leveraged to construct a set of view-sharing bipartite graphs. By efficient bipartite graph partitioning, a set of diversified base clusterings are obtained, which are further utilized to construct a unified bipartite graph for achieving the final clustering in the late-stage fusion. It is noteworthy that our FastMICE approach has almost linear time and space complexity, and is capable of producing high-quality clustering results without dataset-specific tuning. Extensive experiments are conducted on 22 real-world multi-view datasets, including 10 general-scale datasets and 12 large-scale datasets, which demonstrate the scalability (for extremely large datasets), the superiority (in clustering performance), and the simplicity (to be applied) of our approach. For clarity, the contributions of this work are summarized below.

- This paper for the first time, to our knowledge, presents the concept of *random view group* to capture the versatile view-wise relationships, which extends *multiple* views to *many* random view groups and may significantly benefit the clustering robustness while maintaining high efficiency.
- A hybrid early-late fusion strategy is devised, which breaks through the conventional single-stage fusion paradigm and enables the multi-stage fusions to jointly explore different levels of the multi-view information.
- A novel large-scale MVC approach termed FastMICE is proposed, whose advantages are three-fold: (i) it has almost *linear time and space complexity* and is feasible for very large-scale datasets; (ii) it is able to achieve *superior clustering performance* over the state-of-the-art approaches as confirmed by extensive experimental results; (iii) it is simple to be applied, where *no dataset-specific hyperparameter-tuning* is required across various multi-view datasets.

The remainder of the paper is organized as follows. Section II reviews the related works on multi-view clustering and ensemble clustering. Section III describes the overall framework of our FastMICE approach. Section IV reports the experimental results. Finally, Section V concludes this paper.

II. RELATED WORK

In this paper, we propose a new large-scale MVC approach termed FastMICE, which involves both MVC and ensemble clustering (EC). In this section, the related works on MVC and EC will be reviewed in Sections II-A and II-B, respectively.

A. Multi-View Clustering

In recent years, many MVC methods have been developed from different technical perspectives [2]. In spite of the difference in their specific models, they typically share a common and essential task, that is, how to fuse the information from multiple views. A straightforward strategy is to concatenate the features from all views and then perform single-view clustering on the concatenated features, which, however, ignores the multi-view

complementariness and is rarely adopted. Besides feature concatenation, according to their fusion stage, most of the existing MVC methods can be classified into two categories, i.e., the early fusion methods [3], [5], [9] and the late fusion methods [6], [10], [11].

Early fusion is probably the most widely-adopted fusion strategy in MVC [3], [5], [8], [9], which typically formulates the information of all views in a unified optimization or heuristic model (as shown in Fig. 1(a)). In early fusion, the information of each view can be given via different representations, such as the original features [9], transition probability matrix [3], K-nearest-neighbor (K-NN) graph [5], and so forth. Xia et al. [3] constructed a shared low-rank transition probability matrix by exploiting multiple transition probability matrices from multiple views, and then performed spectral clustering on the shared matrix for final clustering. Zhang et al. [9] conducted multi-view subspace clustering by minimizing a self-expressive loss on each view with the tensorized low-rank constraint. Xie et al. [22] extended the tensorized multi-view subspace clustering by further incorporating a local structure constraint. Liang et al. [5], [23] performed graph fusion on multiple K-NN graphs from multiple views with cross-view consistency and inconsistency jointly modeled.

Late fusion is another popular fusion strategy in recent years [6], [10], [11], [12], which first builds a base clustering on each view (often separately) and then fuses them at the partition-level to obtain a consensus clustering (as shown in Fig. 1(b)). Wang et al. [11] built the base clusterings by performing kernel k-means clustering on each view, and learned a consensus clustering by maximizing alignment between the consensus clustering and the base clusterings. Kang et al. [6] performed spectral clustering on the subspace representation of each view to obtain a corresponding base clustering, and learned a consensus clustering by minimizing the distance between a unified cluster indicator matrix and the multiple base clusterings [6].

These MVC methods [3], [5], [6], [8], [9], [10], [11], [12] seek to fuse the multi-view information in different stages and through different techniques. Yet surprisingly, most of them perform the fusion in a single-stage manner (either in the early stage or in the late stage), which lack the ability to go beyond the single-stage fusion to investigate more possibilities in multi-stage fusions. Besides the limitation in their fusion strategy, another limitation is that many of them still suffer from quadratic or cubic computational complexity, which makes them almost infeasible for large-scale datasets. Recently some large-scale MVC methods have been proposed, among which the anchor-based methods have been one of the representative categories [17], [18], [19], [20]. However, in terms of anchor selection, these anchor-based methods either learn a unified anchor set for all views [17], [19], [20] or learn a separate anchor set for each view [18]. In terms of fusion stage, they still dwell in the single-stage fusion strategy. Moreover, for most of the previous MVC methods, including the general-scale methods [3], [5], [6], [8], [9], [10], [11] and the large-scale methods [17], [18], [19], [21], their requirement of dataset-specific hyperparameter-tuning also poses a major hurdle for their real-world applications.

B. Ensemble Clustering

The purpose of EC is to combine multiple base clusterings into a better and more robust consensus clustering [24], [25], [26], [27], [28], [29], [30], [31]. In the final stage of our FastMICE approach, multiple base clusterings are fused into a unified clustering result, which can be viewed as an EC process. Therefore, in this section, we will also review the related works on EC.

Previous EC methods can mostly be classified into three categories, i.e., the pair-wise co-occurrence based methods [24], [25], the median partition based methods [26], [27], and the graph partitioning based methods [16], [29], [30], [32]. The pair-wise co-occurrence based methods typically construct a co-association matrix by considering the pair-wise co-occurrence relationship in base clusterings, and then perform some clustering algorithm on the co-association matrix to obtain the consensus clustering. Fred and Jain [24] proposed the evidence accumulation clustering method which imposes hierarchical agglomerative clustering on the co-association matrix. Huang et al. [25] refined the co-association matrix by an entropy based local weighting strategy and presented the locally weighted evidence accumulation method. The median partition based methods treat the EC problem as an optimization problem, which aims to find a median clustering by maximizing the similarity between this clustering and the base clusterings. Topchy et al. [26] cast the EC problem as a maximum-likelihood problem and solved it via the EM algorithm. Huang et al. [27] formulated the EC problem as a binary linear programming problem and solved it via the factor graph model. The graph partitioning based methods represent the multiple base clusterings as a graph structure and obtain the consensus clustering by partitioning this graph. Strehl and Ghosh [32] considered the concept of hyper-edge and presented three graph partitioning algorithms. Ren et al. [29] took into account the importance of the objects and devised three graph partitioning based consensus functions for weighted-object ensemble clustering.

Despite the progress of these EC works [24], [25], [26], [27], [28], [29], [30], [31], most of them are devised for single-view datasets and lack the consideration of multi-view scenarios. Recently Tao et al. [33] proposed a multi-view ensemble clustering (MVEC) method, which learns a consensus clustering from the multiple co-association matrices built in multiple views with low-rank and sparse constraints. Tao et al. [7] further incorporated marginalized denoising autoencoder into MVEC, and presented a marginalized multi-view ensemble clustering (M ^{2} VEC) method. However, in MVEC and M ^{2} VEC, the base clusterings in different views are generated separately, without leveraging multi-view complementariness in their ensemble generation. Moreover, the high computational complexity also restricts their applications in large-scale scenarios.

III. PROPOSED FRAMEWORK

In this section, we describe the proposed FastMICE approach in detail. Specifically, the notations are introduced in Section III-A. The formation of random view groups is provided in Section III-B. The view-sharing bipartite graph is described in Section III-C. The generation of the diversified base clusterings

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

$\mathcal{X}$	A multi-view dataset
N	# of samples in the dataset
V	# of views in the dataset
$View_v$	The v-th view in the dataset
$\mathcal{X}_v \in \mathbb{R}^{N \times d_v}$	Data matrix associated with $View_v$
$\mathcal{VG}$	The set of view groups
M	# of view groups
$\mathcal{VG}^{(m)}$	The m-th view group
$V^{(m)}$	# of view members in $\mathcal{VG}^{(m)}$
$View_v^{(m)}$	The v-th view members in $\mathcal{VG}^{(m)}$
$\tau_v^{(m)}$	Feature sampling ratio for $View_v^{(m)}$
$\tilde{\mathcal{X}}_v^{(m)} \in \mathbb{R}^{N \times \tilde{d}_v^{(m)}}$	Data matrix after feature sampling in $View_v^{(m)}$
$\tilde{x}_{v,i}^{(m)}$	The i-th sample in $\tilde{\mathcal{X}}_v^{(m)}$
p	# of anchors for each view group
$\bar{p}^{(m)}$	# of anchors for a view member
K	# of nearest neighbors for each view group
$\bar{K}^{(m)}$	# of nearest neighbors for a view member
$\mathcal{A}_v^{(m)}$	The anchor set in $View_v^{(m)}$
$a_{v,i}^{(m)}$	The i-th anchor in $\mathcal{A}_v^{(m)}$
$\mathcal{G}_v^{(m)}$	The bipartite sub-graph for $View_v^{(m)}$
$\mathcal{L}_v^{(m)}$	The left node set of $\mathcal{G}_v^{(m)}$
$\mathcal{R}_v^{(m)}$	The right node set of $\mathcal{G}_v^{(m)}$
$B_v^{(m)}$	The cross-affinity matrix of $\mathcal{G}_v^{(m)}$
$\mathcal{G}^{(m)}$	The view-sharing bipartite graph for $\mathcal{VG}^{(m)}$
$\mathcal{L}^{(m)}$	The left node set of $\mathcal{G}^{(m)}$
$\mathcal{R}^{(m)}$	The right node set of $\mathcal{G}^{(m)}$
$B^{(m)}$	The cross-affinity matrix of $\mathcal{G}^{(m)}$
$E^{(m)}$	Full affinity matrix of graph $\mathcal{G}^{(m)}$
$L^{(m)}$	Graph Laplacian of graph $\mathcal{G}^{(m)}$
$D^{(m)}$	Degree matrix of graph $\mathcal{G}^{(m)}$
$\lambda_i^{(m)}$	The i-th eigenvalue of graph $\mathcal{G}^{(m)}$
$y_i^{(m)}$	The i-th eigenvector of graph $\mathcal{G}^{(m)}$
$\mathcal{G}_s^{(m)}$	A small graph with $\mathcal{R}^{(m)}$ as the node set
$E_s^{(m)}$	Affinity matrix of graph $\mathcal{G}_s^{(m)}$
$L_s^{(m)}$	Graph Laplacian of graph $\mathcal{G}_s^{(m)}$
$D_s^{(m)}$	Degree matrix of graph $\mathcal{G}_s^{(m)}$
$\delta_i^{(m)}$	The i-th eigenvalue of graph $\mathcal{G}_s^{(m)}$
$u_i^{(m)}$	The i-th eigenvector of graph $\mathcal{G}_s^{(m)}$
Π	The ensemble of M base clusterings
π^m	The m-th base clustering in Π
$\mathcal{C}$	The set of all clusters in Π
C_i	The i-th cluster in $\mathcal{C}$
k_c	Number of clusters in $\mathcal{C}$
$\mathcal{G}$	A bipartite graph between $\mathcal{X}$ and $\mathcal{C}$
B	Cross-affinity matrix of graph $\mathcal{G}$.
$\mathcal{G}_s$	A small graph with $\mathcal{C}$ as the node set
E_s	Affinity matrix of graph $\mathcal{G}_s$

is introduced in Section III-D. Then the base clusterings are fused into a unified clustering via a highly efficient consensus function in Section III-E. The time and space complexity of FastMICE is analyzed in Section III-F.

A. Notations

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ be a dataset with N data samples, where x_i is the i-th sample. For a multi-view dataset, each data sample can be represented by features from different views. Thus, the multi-view dataset can be denoted as $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_V\}$, where $\mathcal{X}_v \in \mathbb{R}^{N \times d_v}$ is the data matrix of the v-th view, V is the number of views, and d_v is

the dimension of the v-th view. For convenience of the later view group formation, we denote the set of V views as $\mathcal{V} = \{View_1, View_2, \ldots, View_V\}$, where $View_v$ is the v-th view in the dataset. Note that $\mathcal{X}_v \in \mathbb{R}^{N \times d_v}$ is the data matrix associated with the v-th view (i.e., $View_v$). For clarity, the notations used throughout the paper are given in Table I.

The purpose of MVC is to fuse the information of multiple views for enhanced clustering. In large-scale multi-view scenario, where the data size N can be very large, it becomes a critical challenge how to robustly fuse the multi-view information while ensuring high efficiency and practicality, which will be the focus of our following sections.

B. Early-Stage View Group Formation

An essential task of MVC is to fuse the information of multiple views for robust clustering result. The previous MVC algorithms differ from each other mainly in their fusion stages and fusion techniques [5], [6], [8], [11], [12], but they generally have two characteristics in common. First, they tend to perform the multi-view fusion in a single stage, either early or late. Second, they often implicitly comply with a single-or-all paradigm, where each stage of them involves *either* a single view *or* all views. For example, in the late fusion algorithms [6], [11], [12], each base clustering is constructed on a *single* view independently of other views, while the final fusion process utilizes the base clusterings from *all* views in a one-shot manner. However, the vast middle ground between single-views-independently and all-views-together has rarely be explored by previous works.

Different from the conventional single-or-all paradigm, in the section, we present the concept of random view groups, each of which encapsules a random number of views and serves as a basic unit for the view-wise diversification and the hybrid early-late fusion in our FastMICE framework. Formally, let $\mathcal{VG}^{(m)}$ be the m-th view group, with a randomly selected subset of views, and $V^{(m)}$ be the number of selected views in $\mathcal{VG}^{(m)}$. The number of views in each view group can be randomly chosen in the range of $[V_{min}, V_{max}]$, where V_{min} and V_{max} are respectively the lower bound and the upper bound of the number of selected views such that $1 \leq V_{min} \leq V_{max} \leq V$.

Each selected view in the view group is called a *view member*. To enhance the diversity of view groups, we set the lower bound $V_{min} = 1$ and the upper bound $V_{max} = V$, through which each view group can have at least one view member and at most V view members. Thus, the m-th view group $\mathcal{VG}^{(m)}$ can be formed by randomly selected $V^{(m)}$ views from the set of all views, denoted as

$$\mathcal{VG}^{(m)} = \{View_1^{(m)}, View_2^{(m)}, \ldots, View_{V^{(m)}}^{(m)}\}, \quad (1)$$

where $View_v^{(m)}$ is the v-th view member in $\mathcal{VG}^{(m)}$. The data matrix associated with $View_v^{(m)}$ is denoted as $\mathcal{X}_v^{(m)} \in \mathbb{R}^{N \times d_v^{(m)}}$, where $d_v^{(m)}$ is the dimension of this view member. By performing the randomization process repeatedly, a set of random view groups can be obtained, that is

$$\mathcal{VG} = \{\mathcal{VG}^{(1)}, \mathcal{VG}^{(2)}, \ldots, \mathcal{VG}^{(M)}\}, \quad (2)$$

where M is the number of the generated random view groups. As each view group leads to the generation of a base clustering, the number of view groups is also the number of base clusterings, also known as the ensemble size.

It is worth mentioning that the early-fusion MVC methods (as shown in Fig. 1(a)) can be viewed as a special instance of our view group based formulation with all views selected into a single view group by setting $V_{min} = V_{max} = V$ and $M = 1$. Similarly, the late fusion MVC methods (as shown in Fig. 1(b)) can also be viewed as special instance of our view group based formulation with each view being a single view group by setting $V_{min} = V_{max} = 1$ and $M = V$.

C. View-Sharing Bipartite Graph Construction

With multiple random view groups obtained, our next goal is to perform the early-stage fusion in the view groups. Note that the early-stage fusion is not meant to achieve a single optimal solution in a one-shot manner. Instead, it performs fusions on multiple view groups, and builds multiple view-sharing bipartite graphs with multi-level diversities. Besides the diversity, to enable the scalability for very large datasets, the efficiency is another of our key concerns during the early-stage multi-fusion process.

Particularly, in each view group, the bipartite graph structure is exploited to formulate the information of multiple view member. In recent years, the bipartite graph structure has shown its advantage in handling large-scale datasets [15], [16], [19], [20]. From the perspective of topology, the bipartite graph is built between the N data samples and a set of p anchors (or representatives), typically with $p \ll N$ for large-scale datasets. From the perspective of matrix notation, the bipartite graph can be represented by an $N \times p$ cross-affinity matrix with its (i, j)-th entry being the affinity between the i-th sample and the j-th anchor, which can also be regarded as encoding the data samples via this small set of anchors.

To enable the diversity of the bipartite graph construction in multiple view groups, we simultaneously leverage three levels of diversification, corresponding to the *feature*-level, the *anchor*-level, and the *neighborhood*-level, respectively.

In the feature-level diversification, we first perform random feature sampling on each view member in a view group. Let $\tau_v^{(m)}$ denote the feature sampling ratio for the v-th view member in the m-th view group (i.e., $View_v^{(m)}$). To inject the diversity, the sampling ratio for each view member is randomly chosen in the range of $[\tau_{min}, \tau_{max}]$, where τ_{min} and τ_{max} are respectively the lower and the upper bounds of the sampling ratio such that $0 < \tau_{min} \leq \tau_{max} \leq 1$. By performing feature sub-sampling with a randomized sampling ratio, we can obtain a subset of features for each view member. For $View_v^{(m)}$, its data matrix *after* random feature sampling can be denoted as $\tilde{\mathcal{X}}_v^{(m)} \in \mathbb{R}^{N \times \tilde{d}_v^{(m)}}$, where $\tilde{d}_v^{(m)} = \lceil \tau_v^{(m)} \cdot d_v^{(m)} \rceil$ is the reduced dimension, and $\lceil \cdot \rceil$ obtains the ceiling of a real value.

After random feature sampling, we proceed to construct a bipartite sub-graph on each view member, and then combine these bipartite sub-graphs into a unified view-sharing bipartite graph, where a set of p anchors are required and the K-NN

sparsification is performed. In prior works, the anchor set for multiple views can be obtained by performing k-means clustering on the concatenated features of all views [17] or by optimizing some objective function to learn consensus anchors [20]. However, on the one hand, they typically aim to find a set of anchors suitable for all views, without sufficient consideration to view-specific characteristics. On the other hand, their anchor selection or learning process may also be computationally expensive when facing very large datasets.

Instead of pursuing a set of consensus anchors, we *distribute* the task of finding p anchors to the multiple view members. Without prior knowledge, we expect the multiple view members in the same view group to contribute equally. Specifically, each view member is expected to make a contribution of $\bar{p}^{(m)} = \lceil p/V^{(m)} \rceil$ anchors. Similarly, in terms of the neighborhood, each view member is expected to contribute $\bar{K}^{(m)} = \lceil K/V^{(m)} \rceil$ nearest neighbors. Thereafter, on each view member in $\mathcal{VG}^{(m)}$, we aim to build a bipartite sub-graph between N samples and $\bar{p}^{(m)}$ anchors, with each sample connected to $\bar{K}^{(m)}$ nearest anchors.

For the m-th view group $\mathcal{VG}^{(m)}$, the anchor selection on the v-th view member $View_v^{(m)}$ (associated with the data matrix $\tilde{\mathcal{X}}_v^{(m)}$) is performed via the hybrid representative selection strategy [16], which efficiently obtains a set of $\bar{p}^{(m)}$ anchors, denoted as

$$\mathcal{A}_v^{(m)} = \{a_{v,1}^{(m)}, a_{v,2}^{(m)}, \ldots, a_{v,\bar{p}^{(m)}}^{(m)}\}, \qquad (3)$$

where $a_{v,i}^{(m)} \in \mathbb{R}^{\tilde{d}_v^{(m)}}$ is the i-th anchor selected from $\tilde{\mathcal{X}}_v^{(m)}$.

Then we define the bipartite sub-graph for the view member $View_v^{(m)}$ as follows:

$$\mathcal{G}_v^{(m)} = \{\mathcal{L}_v^{(m)}, \mathcal{R}_v^{(m)}, B_v^{(m)}\}, \qquad (4)$$

where $\mathcal{L}_v^{(m)} = \mathcal{X}$ and $\mathcal{R}_v^{(m)} = \mathcal{A}_v^{(m)}$ are the left and right node sets of the bipartite sub-graph, respectively, and $B_v^{(m)} \in \mathbb{R}^{N \times \bar{p}^{(m)}}$ is the cross-affinity matrix. An edge between two nodes exists if and only if one node is a data sample, another node is an anchor, and this anchor is one of the sample's $\bar{K}^{(m)}$-nearest anchors. Formally, the (i, j)-th entry of the cross-affinity matrix $B_v^{(m)}$ can be defined as

$$b_{v,ij}^{(m)} = \begin{cases} Sim(\tilde{x}_{v,i}^{(m)}, a_{v,j}^{(m)}), & \text{if } a_{v,j}^{(m)} \in \mathcal{N}_{\bar{K}^{(m)}}(\tilde{x}_{v,i}^{(m)}), \\ 0, & \text{otherwise}, \end{cases} \qquad (5)$$

where $\tilde{x}_{v,i}^{(m)}$ denotes the i-th sample in this view member (with reduced dimension), $Sim(\cdot)$ computes the similarity between two vectors, and $\mathcal{N}_K(x)$ denotes the set of K-nearest anchors of sample x. Note that $Sim(\cdot)$ can be any similarity measure. Typically, we utilize the Gaussian kernel similarity, which maps the euclidean distance to a similarity measure via a Gaussian kernel. Thus, with each sample linked to $\bar{K}^{(m)}$ nearest anchors, the cross-affinity matrix $B_v^{(m)}$ can be represented as a sparse matrix with only $N \cdot \bar{K}^{(m)}$ non-zeros entries, whose sparsity can significantly benefit the later matrix computations.

For the v-th view member in the m-th view group, a bipartite sub-graph $\mathcal{G}_v^{(m)}$ (with the cross-affinity matrix $B_v^{(m)}$) can be obtained. From the perspective of information encoding,

the cross-affinity matrix $B_v^{(m)}$ can be regarded as a representation matrix to encode the N data samples by the $\bar{p}^{(m)}$ anchors, with each row of $B_v^{(m)}$ being a low-dimensional feature vector for a data sample. As there are $V^{(m)}$ view members in $\mathcal{VG}^{(m)}$, each data sample can be represented (or encoded) by the totally $\bar{p}^{(m)} \cdot V^{(m)}$ anchors, which lead to the view-sharing bipartite graph for the entire view group, that is

$$\mathcal{G}^{(m)} = \{\mathcal{L}^{(m)}, \mathcal{R}^{(m)}, B^{(m)}\}, \qquad (6)$$

where $\mathcal{L}^{(m)} = \mathcal{X}$ is the left node set of the bipartite graph, and $\mathcal{R}^{(m)} = \mathcal{A}_1^{(m)} \bigcup, \ldots, \bigcup \mathcal{A}_{V^{(m)}}^{(m)}$ is the right node set, which is the union of the anchor sets of the $V^{(m)}$ view members from $\mathcal{VG}^{(m)}$. The cross-affinity matrix for $\mathcal{G}^{(m)}$ is defined as $B^{(m)} = [\bar{B}_1^{(m)}, \ldots, \bar{B}_{V^{(m)}}^{(m)}]$, where $\bar{B}_v^{(m)}$ is computed by normalizing each row of $B_v^{(m)}$ to unit norm, so as to adjust multiple bipartite sub-graphs into similar scales.

The time complexity of building the view-sharing bipartite graph $\mathcal{G}^{(m)}$ mainly comes from the anchor selection and the cross-affinity matrix construction. For the v-th view member in $\mathcal{VG}^{(m)}$, the anchor selection via hybrid representative selection takes $O((\bar{p}^{(m)})^2 \tilde{d}_v^{(m)} t)$ time, where t is the number of iterations. Then it takes $O(N(\bar{p}^{(m)})^{1/2})$ time to build the cross-affinity matrix $B_v^{(m)}$ via approximate K-NN computation [16]. Thus the construction of a bipartite sub-graph (for a view member) takes $O(N(\bar{p}^{(m)})^{1/2} + (\bar{p}^{(m)})^2 \tilde{d}_v^{(m)} t)$ time, which is dominated by $O(N(\bar{p}^{(m)})^{1/2})$ with $\bar{p}^{(m)} \ll N$. The construction of a view-sharing bipartite graph for $\mathcal{VG}^{(m)}$ involves the construction of $V^{(m)}$ bipartite sub-graphs, which in total take $O(N(\bar{p}^{(m)})^{1/2} V^{(m)})$ time. With $\bar{p}^{(m)} = \lceil p/V^{(m)} \rceil$ and $V^{(m)} \leq V$, it is easy to know that $\bar{p}^{(m)} V^{(m)} \in [p, p + V^{(m)})$, leading to $(\bar{p}^{(m)})^{1/2} V^{(m)} = (\bar{p}^{(m)}(V^{(m)})^2)^{1/2} < ((p+V)V)^{1/2}$. With $V \ll p$, the time complexity of constructing a view-sharing bipartite graph can be obtained as $O(Np^{1/2}V^{1/2})$.

As $\bar{B}_v^{(m)}$ is a sparse matrix with $N \cdot \bar{K}^{(m)}$ non-zero entries and $B^{(m)}$ is a sparse matrix with $N \cdot \bar{K}^{(m)} \cdot V^{(m)}$ non-zero entries, the space complexity of constructing a view-sharing bipartite graph is $O(N\bar{K}^{(m)}V^{(m)})$. With $\bar{K}^{(m)} = \lceil K/V^{(m)} \rceil$ and $V^{(m)} \leq V$, we have $\bar{K}^{(m)}V^{(m)} \in [K, K+V)$. Therefore the space complexity can be written as $O(N(K+V))$.

D. Ensemble Generation in View Groups

In this section, we describe the ensemble generation, i.e., the generation of a set of diversified base clusterings, based on the view-sharing bipartite graphs in multiple view groups.

Specifically, with a view-sharing bipartite graph constructed for each view group, a set of M view-sharing bipartite graphs, ranging from $\mathcal{G}^{(1)}$ to $\mathcal{G}^{(M)}$, can be built for the M view groups. For the base clustering in each view group, the number of clusters, say, $k^{(m)}$, will be randomly selected in the range of $[k_{min}, k_{max}]$, where k_{min} and k_{max} are respectively the lower bound and the upper bound of the cluster number. In the following, we proceed to describe the fast partitioning of the m-th view-sharing bipartite graph into $k^{(m)}$ clusters.

For the m-th view group, there are totally $N + \bar{p}^{(m)} \cdot V^{(m)}$ nodes in the view-sharing bipartite graph $\mathcal{G}^{(m)}$, with N data samples in the left node set and $\bar{p}^{(m)} \cdot V^{(m)}$ anchors in the right node set. By treating $\mathcal{G}^{(m)}$ as a general graph, its full affinity matrix $E^{(m)} \in \mathbb{R}^{(N+\bar{p}^{(m)}V^{(m)}) \times (N+\bar{p}^{(m)}V^{(m)})}$ can be represented as

$$E^{(m)} = \begin{bmatrix} 0 & B^{(m)} \\ (B^{(m)})^\top & 0 \end{bmatrix} \tag{7}$$

To partition this graph via conventional spectral clustering [34], the following generalized eigen-decomposition problem should be solved

$$L^{(m)} y^{(m)} = \lambda^{(m)} D^{(m)} y^{(m)}, \tag{8}$$

where $L^{(m)} = D^{(m)} - E^{(m)}$ is the graph Laplacian, $D^{(m)}$ is the degree matrix with its (i, i)-th entry being the sum of the i-row in $E^{(m)}$. It takes $O((N + \bar{p}^{(m)} V^{(m)})^3)$ time to solve this problem via SVD, which is computationally expensive for large-scale datasets.

Due to the imbalanced sizes of the left and right node sets in the bipartite graph, with $\bar{p}^{(m)} V^{(m)} \approx p \ll N$, the eigen-decomposition problem in (8) can be reduced to an eigen-decomposition problem on a smaller graph $\mathcal{G}_s^{(m)}$ with a node set of $\bar{p}^{(m)} V^{(m)}$ anchors and an affinity matrix of $E_s^{(m)} = (B^{(m)})^\top (\hat{D}^{(m)})^{-1} B^{(m)}$, whose computation takes $O(NK^2)$ time, where $\hat{D}^{(m)}$ is a diagonal matrix with its (i, i)-th entry being the sum of the i-th row in the cross-affinity matrix $B^{(m)}$. The generalized eigen-decomposition problem on the reduced graph $\mathcal{G}_s^{(m)}$ can be formulated as

$$L_s^{(m)} u^{(m)} = \delta^{(m)} D_s^{(m)} u^{(m)}, \tag{9}$$

where $L_s^{(m)} = D_s^{(m)} - E_s^{(m)}$ is the graph Laplacian, $D_s^{(m)}$ is the degree matrix with its (i, i)-th entry being the sum of the i-th row in $E_s^{(m)}$.

By solving the eigen-decomposition problem in (9) with $O((\bar{p}^{(m)} V^{(m)})^3) \approx O(p^3)$ time, the first $k^{(m)}$ eigen-pairs can be obtained, denoted as $\{\delta_i^{(m)}, u_i^{(m)}\}_{i=1}^{k^{(m)}}$ with $0 = \delta_1^{(m)} \leq \delta_2^{(m)} \leq \cdots \leq \delta_{k^{(m)}}^{(m)} < 1$. Let $\{\lambda_i^{(m)}, y_i^{(m)}\}_{i=1}^{k^{(m)}}$ be the first k eigen-pairs for the graph $\mathcal{G}^{(m)}$ with $0 = \lambda_1^{(m)} \leq \lambda_2^{(m)} \leq \cdots \leq \lambda_{k^{(m)}}^{(m)} < 1$, which can be efficiently computed from $\{\delta_i^{(m)}, u_i^{(m)}\}_{i=1}^{k^{(m)}}$ with the following properties [35]

$$\lambda_i^{(m)} (2 - \lambda_i^{(m)}) = \delta_i^{(m)}, \tag{10}$$

$$y_i^{(m)} = \begin{bmatrix} h_i^{(m)} \\ u_i^{(m)} \end{bmatrix} \tag{11}$$

where $h_i^{(m)} = (\hat{D}^{(m)})^{-1} B^{(m)} u_i^{(m)} / (1 - \lambda_i^{(m)})$.

The time complexity of solving the eigen-decomposition problem in (9) is $O(p^3)$. Due to the sparsity of $B^{(m)}$, it takes $O(NK)$ time to compute $h_i^{(m)}$ and obtain each eigenvector $\{y_i^{(m)}\}_{i=1}^{k^{(m)}}$ from $\{u_i^{(m)}\}_{i=1}^{k^{(m)}}$. Then, the $k^{(m)}$ eigenvectors will be stacked as a matrix with each row treating as a new feature vector, upon which the k-means discretization can be performed

to obtain the m-th base clustering with $O(N(k^{(m)})^2 t)$ time. Since $k^{(m)}$ and k are generally at similar scales, the time complexity of generating the base clustering from a view-sharing bipartite graph is $O(N(k^2 t + K^2 + Kk) + p^3)$, and the space complexity is $O(N(k + K))$.

E. Late-Stage Consensus Function

By partitioning the view-sharing bipartite graph of the m-th view group, we obtain a base clustering with $k^{(m)}$ clusters. Then the set of base clusterings generated for the M view groups can be represented as

$$\Pi = \{\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(M)}\}, \tag{12}$$

where $\pi^{(m)} = \{C_1^{(m)}, C_2^{(m)}, \ldots, C_{k^{(m)}}^{(m)}\}$ is the m-th base clustering, and $C_i^{(m)}$ is the i-th cluster in $\pi^{(m)}$. Each base clustering consists of a certain number of clusters. For convenience, we represent the set of clusters in *all* base clusterings as

$$\mathcal{C} = \{C_1, C_2, \ldots, C_{k_c}\}, \tag{13}$$

where C_i is the i-th cluster, and $k_c = \sum_{m=1}^M k^{(m)}$ is the total number of clusters in all base clusterings.

To jointly formulate the information of multiple base clusterings, we construct a unified bipartite graph by treating both the data samples and the base clusters as graph nodes, denoted as

$$\mathcal{G} = \{\mathcal{L}, \mathcal{R}, B\}, \tag{14}$$

where $\mathcal{L} = \mathcal{X}$ is the left node set with N data samples, $\mathcal{R} = \mathcal{C}$ is the right node set with k_c base clusters, and $B \in \mathbb{R}^{N \times k_c}$ is the cross-affinity matrix. A link between two nodes exists if and only if one of them is a data sample and the other one is a base cluster that contains the sample. Thus, the (i, j)-th entry of the cross-affinity matrix B can be defined as

$$b_{ij} = \begin{cases} 1, & \text{if } x_i \in C_j, \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

Note that each data sample belongs to one and only one cluster in a base clustering. With a total of M base clusterings, each data sample will be linked to exactly M clusters in the unified bipartite graph. That is, in each row of B, there are exactly M non-zero entries. Therefore, B is a matrix with $N \cdot M$ non-zero entries.

Similar to the partitioning of the view-sharing bipartite graph $\mathcal{G}^{(m)}$, the eigen-decomposition problem of the unified bipartite graph $\mathcal{G}$ can also be tackled by conducting the eigen-decomposition on a smaller graph $\mathcal{G}_s$ with an affinity matrix $E_s = B^\top \hat{D}^{-1} B$, whose computation takes $O(NM^2)$ time, where $\hat{D}$ is a diagonal matrix with its (i, i)-th entry being the sum of the i-th row in B.

To obtain the final clustering result with k clusters, solving the eigen-decomposition of the graph $\mathcal{G}_s$ takes $O(k_c^3)$ time. Then it takes $O(NM)$ time to compute *each* of the k eigenvectors of $\mathcal{G}$ from the eigenvectors of $\mathcal{G}_s$. Thereafter, it takes $O(Nk^2 t)$ time to perform the k-means discretization. Thus, the time complexity of the overall consensus function with the unified bipartite graph is $O(N(k^2 t + M^2 + Mk) + k_c^3)$, and the space complexity is $O(N(k + M))$.

TABLE II
DESCRIPTION OF THE BENCHMARK DATASETS

	Dataset	#Sample	#View	#Class	Dimension
General	MSRCv1	210	4	7	CM (24), GIST (512), LBP (256), GENT (254)
	Yale	165	3	15	Intensity 4,096, LBP (3,304), Gabor (6,750)
	ORL	400	3	40	Intensity (4,096), LBP (3,304), Gabor (6,750)
	Movies	617	2	17	Keywords (1,878), Actors (1,398)
	BBCSport	544	2	5	View1 (3,183), View2 (3,203)
	NH-550	550	3	5	Intensity (2,000), LBP (3,304), Gabor (6,750)
	Out-Scene	2,688	4	8	Color (432), GIST (512), HOG (256), LBP (48)
	Citeseer	3,312	2	6	Content (3,703), Citation (3,312)
	NH-4660	4,660	3	5	Intensity (2,000), LBP (3,304), Gabor (6,750)
	ALOI	10,800	4	100	Similarity (77), Haralick (13), HSV (64), RGB (125)
Large	NUS-WIDE	30,000	5	31	CH (64), CM (225), CORR (144), EDH (73), WT(128)
	VGGFace2-50	34,027	4	50	LBP (944), HOG (576), GIST(512), Gabor (640)
	VGGFace2-100	65,342	4	100	LBP (944), HOG (576), GIST(512), Gabor (640)
	VGGFace2-200	124,880	4	200	LBP (944), HOG (576), GIST(512), Gabor (640)
	CIFAR-10	60,000	4	10	LBP (944), HOG (576), GIST(512), Gabor (640)
	CIFAR-100	60,000	4	100	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-10	38,654	4	10	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-20	63,896	4	20	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-50	126,054	4	50	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-100	195,537	4	100	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-200	286,006	4	200	LBP (944), HOG (576), GIST(512), Gabor (640)
	YTF-400	398,191	4	400	LBP (944), HOG (576), GIST(512), Gabor (640)

F. Complexity Analysis

In the following, we will analyze the time and space complexity of the proposed FastMICE approach.

1) Time Complexity: This section analyzes the time complexity of the FastMICE approach. The generation of random view groups in Section III-B takes $O(MV)$ time. Then it takes $O(Np^{1/2}V^{1/2})$ time to construct a view-sharing bipartite graph in each view group, and totally $O(NMp^{1/2}V^{1/2})$ time to construct the view-sharing bipartite graphs in all the M view groups in Section III-C. Thereafter, the ensemble generation in Section III-D (to produce M base clusterings) takes $O(NM(k^2t + K^2 + Kk) + Mp^3)$ time. Finally, the consensus function in Section III-E takes $O(N(k^2t + M^2 + Mk) + k_c^3)$ time.

In large-scale scenarios, the number of anchors may range from hundreds to thousands, which is much smaller than the data size N, but much larger than the number of clusters k or the number of nearest neighbors K. With $k, K, M, V \ll p \ll N$, the overall time complexity of the FastMICE approach can be written as $O(NMp^{1/2}V^{1/2})$, which is linear to the data size N.

2) Space Complexity: This section analyzes the space complexity of FastMICE. The construction of a view-sharing bipartite graph takes $O(N(K + V))$ space, while its partitioning takes $O(N(k + K))$ space. Since the base clusterings can be generated in a serial processing manner, the space complexity of generating multiple base clusterings is still $O(N(k + K + V))$. The consensus function takes $O(N(k + M))$ space. Thus, the overall space complexity of FastMICE is $O(N(k + K + V + M))$, which is also linear to the data size N.

IV. EXPERIMENTS

In this section, we evaluate the proposed FastMICE approach against the state-of-the-art MVC approaches on a variety of general-scale and large-scale multi-view datasets. The experiments are conducted on a PC with an Intel i5-6600 CPU and 16 GB of RAM.

A. Benchmark Datasets

In the experiments, 22 real-world multi-view datasets are used, including 10 general-scale datasets and 12 large-scale datasets (as shown in Table II).

Specifically, the 10 general-scale datasets are *MSRCv1* [36], *Yale* [36], *ORL* [36], *Movies* [37], *BBCSport* [38], *NH-550* [38], *Out-Scene* [39], *Citeseer* [12], *NH-4660* [40], and *ALOI* [41], and the 12 large-scale datasets are *NUS-WIDE* [21], *VGGFace2-50*, *VGGFace2-100*, *VGGFace2-200*, *CIFAR-10*, *CIFAR-100*, *YTF-10*, *YTF-20*, *YTF-50*, *YTF-100*, *YTF-200*, and *YTF-400*. Note that *VGGFace2-50*, *VGGFace2-100*, and *VGGFace2-200* are three versions of the *VGGFace2*[1] dataset. *CIFAR-10* and *CIFAR-100* are two versions of the *CIFAR*[2] dataset. And *YTF-10*, *YTF-20*, *YTF-50*, *YTF-100*, *YTF-200*, and *YTF-400* are six versions of the *YouTube-Faces*[3] (YTF) dataset. The purpose of testing different versions (of different data sizes) of these large-scale datasets is to better evaluate the MVC algorithms with different levels of scalability. The details of the datasets are given in Table II.

B. Baseline Methods and Experimental Settings

Our proposed FastMICE method is experimentally compared against ten MVC methods, which are listed below.
- *MVSC* [17]: multi-view spectral clustering.
- *AMGL* [4]: auto-weighted multiple graph learning.
- *SwMC* [42]: self-weighted multi-view clustering.

[1]www.robots.ox.ac.uk/~vgg/data/vgg_face2/
[2]https://www.cs.toronto.edu/~kriz/cifar.html
[3]https://www.cs.tau.ac.il/~wolf/ytfaces/

TABLE III
AVERAGE NMI(%) SCORES OVER 20 RUNS BY DIFFERENT MULTI-VIEW CLUSTERING ALGORITHMS. ON EACH DATASET, THE BEST TWO SCORES ARE HIGHLIGHTED IN **BOLD**, WHILE THE BEST ONE IN [**BOLD AND BRACKETS**]

Dataset	MVSC	AMGL	SwMC	MVEC	M^2VEC_{km}	M^2VEC_{spec}	BMVC	LMVSC	SMVSC	FPMVS-CAG	FastMICE
MSRCv1	$49.34_{\pm3.78}$	$58.84_{\pm6.33}$	$\mathbf{62.69}_{\pm0.00}$	$62.44_{\pm1.99}$	$59.88_{\pm1.79}$	$60.09_{\pm2.27}$	$53.62_{\pm0.00}$	$27.94_{\pm0.00}$	$61.09_{\pm0.00}$	$56.62_{\pm0.00}$	$[\mathbf{71.04}_{\pm3.77}]$
Yale	$64.10_{\pm3.25}$	$61.78_{\pm2.21}$	$\mathbf{66.28}_{\pm0.00}$	$64.61_{\pm1.11}$	$48.48_{\pm9.28}$	$50.08_{\pm3.73}$	$27.57_{\pm0.00}$	$61.64_{\pm0.00}$	$61.68_{\pm0.00}$	$49.76_{\pm0.00}$	$[\mathbf{67.85}_{\pm2.58}]$
ORL	$84.69_{\pm2.06}$	$\mathbf{85.29}_{\pm1.78}$	$83.31_{\pm0.00}$	$83.29_{\pm1.04}$	$77.59_{\pm1.94}$	$75.16_{\pm2.93}$	$39.29_{\pm0.00}$	$76.42_{\pm0.00}$	$75.26_{\pm0.00}$	$74.30_{\pm0.00}$	$[\mathbf{87.01}_{\pm1.15}]$
Movies	$22.85_{\pm1.50}$	$22.72_{\pm0.88}$	$16.75_{\pm0.00}$	$25.03_{\pm1.15}$	$28.27_{\pm0.42}$	$28.19_{\pm1.12}$	$[\mathbf{29.07}_{\pm0.00}]$	$24.65_{\pm0.00}$	$16.76_{\pm0.00}$	$14.51_{\pm0.00}$	$\mathbf{28.59}_{\pm1.33}$
BBCSport	$69.29_{\pm14.02}$	$87.32_{\pm6.91}$	$46.93_{\pm0.00}$	$[\mathbf{90.57}_{\pm1.16}]$	$\mathbf{89.30}_{\pm1.29}$	$89.18_{\pm1.37}$	$75.97_{\pm0.00}$	$82.88_{\pm0.00}$	$14.83_{\pm0.00}$	$10.37_{\pm0.00}$	$88.95_{\pm2.84}$
NH-550	$77.09_{\pm8.11}$	$59.40_{\pm12.08}$	$82.04_{\pm0.00}$	$\mathbf{84.93}_{\pm3.65}$	$79.62_{\pm5.17}$	$79.22_{\pm4.81}$	$4.14_{\pm0.00}$	$72.59_{\pm0.00}$	$82.49_{\pm0.00}$	$71.35_{\pm0.00}$	$[\mathbf{93.17}_{\pm4.99}]$
OutdoorScene	$23.39_{\pm14.73}$	$44.19_{\pm3.96}$	$43.69_{\pm0.00}$	$48.90_{\pm0.55}$	$47.99_{\pm0.42}$	$47.84_{\pm0.59}$	$\mathbf{53.66}_{\pm0.00}$	$45.97_{\pm0.00}$	$51.67_{\pm0.00}$	$53.04_{\pm0.00}$	$[\mathbf{58.38}_{\pm1.29}]$
Citeseer	$0.40_{\pm0.11}$	$0.53_{\pm0.06}$	$1.22_{\pm0.00}$	$29.12_{\pm0.42}$	$\mathbf{29.69}_{\pm0.48}$	$28.56_{\pm0.36}$	$25.29_{\pm0.00}$	$8.75_{\pm0.00}$	$17.86_{\pm0.00}$	$14.73_{\pm0.00}$	$[\mathbf{31.71}_{\pm2.03}]$
NH-4660	$53.98_{\pm19.66}$	$8.19_{\pm2.82}$	$8.77_{\pm0.00}$	$85.01_{\pm4.17}$	$\mathbf{85.05}_{\pm0.38}$	$84.11_{\pm2.91}$	$74.13_{\pm0.00}$	$68.06_{\pm0.00}$	$66.64_{\pm0.00}$	$67.15_{\pm0.00}$	$[\mathbf{93.39}_{\pm5.58}]$
ALOI	$30.48_{\pm2.14}$	$68.07_{\pm1.72}$	N/A	N/A	N/A	N/A	$75.30_{\pm0.00}$	$74.95_{\pm0.00}$	$57.38_{\pm0.00}$	$55.51_{\pm0.00}$	$[\mathbf{83.39}_{\pm0.75}]$
NUS-WIDE	N/A	N/A	N/A	N/A	N/A	N/A	$[\mathbf{15.48}_{\pm0.00}]$	$8.55_{\pm0.00}$	$11.43_{\pm0.00}$	$12.32_{\pm0.00}$	$\mathbf{13.40}_{\pm0.48}$
VGGFace2-50	N/A	N/A	N/A	N/A	N/A	N/A	$13.64_{\pm0.00}$	$13.62_{\pm0.00}$	$11.16_{\pm0.00}$	$12.33_{\pm0.00}$	$[\mathbf{14.75}_{\pm0.34}]$
VGGFace2-100	N/A	N/A	N/A	N/A	N/A	N/A	$14.30_{\pm0.00}$	$\mathbf{15.15}_{\pm0.00}$	$11.93_{\pm0.00}$	$11.82_{\pm0.00}$	$[\mathbf{15.71}_{\pm0.29}]$
VGGFace2-200	N/A	N/A	N/A	N/A	N/A	N/A	$14.42_{\pm0.00}$	$\mathbf{15.35}_{\pm0.00}$	$11.13_{\pm0.00}$	$9.58_{\pm0.00}$	$[\mathbf{15.50}_{\pm0.36}]$
CIFAR-10	N/A	N/A	N/A	N/A	N/A	N/A	$17.51_{\pm0.00}$	$13.54_{\pm0.00}$	$18.37_{\pm0.00}$	$\mathbf{18.41}_{\pm0.00}$	$[\mathbf{21.42}_{\pm0.52}]$
CIFAR-100	N/A	N/A	N/A	N/A	N/A	N/A	$15.34_{\pm0.00}$	$14.50_{\pm0.00}$	$13.93_{\pm0.00}$	$13.75_{\pm0.00}$	$[\mathbf{18.37}_{\pm0.34}]$
YTF-10	N/A	N/A	N/A	N/A	N/A	N/A	$79.80_{\pm0.00}$	$71.70_{\pm0.00}$	$77.91_{\pm0.00}$	$76.86_{\pm0.00}$	$[\mathbf{81.69}_{\pm2.72}]$
YTF-20	N/A	N/A	N/A	N/A	N/A	N/A	$74.17_{\pm0.00}$	$75.06_{\pm0.00}$	$\mathbf{79.81}_{\pm0.00}$	$77.82_{\pm0.00}$	$[\mathbf{80.03}_{\pm1.73}]$
YTF-50	N/A	N/A	N/A	N/A	N/A	N/A	$82.29_{\pm0.00}$	$80.92_{\pm0.00}$	$80.59_{\pm0.00}$	$79.59_{\pm0.00}$	$[\mathbf{83.15}_{\pm0.78}]$
YTF-100	N/A	N/A	N/A	N/A	N/A	N/A	$82.97_{\pm0.00}$	$79.62_{\pm0.00}$	$75.97_{\pm0.00}$	$67.27_{\pm0.00}$	$[\mathbf{83.09}_{\pm0.69}]$
YTF-200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$\mathbf{79.05}_{\pm0.00}$	$70.64_{\pm0.00}$	$65.10_{\pm0.00}$	$[\mathbf{82.01}_{\pm0.23}]$
YTF-400	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$[\mathbf{79.81}_{\pm0.25}]$
Average Rank	6.55	6.27	6.14	4.77	5.32	5.68	4.41	4.95	4.77	5.82	1.23

* Note that N/A indicates the out-of-memory error.

TABLE IV
AVERAGE ARI(%) SCORES OVER 20 RUNS BY DIFFERENT MULTI-VIEW CLUSTERING ALGORITHMS. ON EACH DATASET, THE BEST TWO SCORES ARE HIGHLIGHTED IN **BOLD**, WHILE THE BEST ONE IN [**BOLD AND BRACKETS**]

Dataset	MVSC	AMGL	SwMC	MVEC	M^2VEC_{km}	M^2VEC_{spec}	BMVC	LMVSC	SMVSC	FPMVS-CAG	FastMICE
MSRCv1	$37.89_{\pm4.62}$	$46.27_{\pm9.59}$	$52.37_{\pm0.00}$	$\mathbf{54.81}_{\pm2.44}$	$49.90_{\pm2.06}$	$49.93_{\pm2.36}$	$42.16_{\pm0.00}$	$14.82_{\pm0.00}$	$50.18_{\pm0.00}$	$43.31_{\pm0.00}$	$[\mathbf{65.49}_{\pm5.21}]$
Yale	$40.15_{\pm5.28}$	$34.35_{\pm4.13}$	$\mathbf{44.82}_{\pm0.00}$	$42.67_{\pm1.60}$	$20.39_{\pm9.86}$	$23.60_{\pm4.62}$	$1.77_{\pm0.00}$	$37.25_{\pm0.00}$	$37.56_{\pm0.00}$	$25.30_{\pm0.00}$	$[\mathbf{46.65}_{\pm3.90}]$
ORL	$\mathbf{56.95}_{\pm6.96}$	$51.86_{\pm6.76}$	$41.39_{\pm0.00}$	$50.26_{\pm4.93}$	$33.40_{\pm7.19}$	$35.11_{\pm5.33}$	$0.61_{\pm0.00}$	$41.68_{\pm0.00}$	$42.39_{\pm0.00}$	$39.99_{\pm0.00}$	$[\mathbf{65.16}_{\pm2.76}]$
Movies	$5.80_{\pm1.23}$	$5.89_{\pm1.20}$	$1.86_{\pm0.00}$	$10.44_{\pm0.65}$	$10.10_{\pm0.53}$	$9.91_{\pm0.67}$	$[\mathbf{13.30}_{\pm0.00}]$	$8.33_{\pm0.00}$	$4.89_{\pm0.00}$	$2.19_{\pm0.00}$	$\mathbf{12.36}_{\pm1.52}$
BBCSport	$67.10_{\pm18.72}$	$89.82_{\pm10.05}$	$37.39_{\pm0.00}$	$[\mathbf{91.71}_{\pm1.06}]$	$90.49_{\pm1.05}$	$90.51_{\pm1.11}$	$69.88_{\pm0.00}$	$87.39_{\pm0.00}$	$6.01_{\pm0.00}$	$10.28_{\pm0.00}$	$\mathbf{91.04}_{\pm3.36}$
NH-550	$74.52_{\pm9.88}$	$53.80_{\pm15.83}$	$\mathbf{84.30}_{\pm0.00}$	$80.35_{\pm8.34}$	$74.71_{\pm9.44}$	$74.73_{\pm8.71}$	$2.66_{\pm0.00}$	$73.18_{\pm0.00}$	$83.95_{\pm0.00}$	$62.25_{\pm0.00}$	$[\mathbf{92.55}_{\pm8.92}]$
OutdoorScene	$18.95_{\pm12.56}$	$33.84_{\pm3.90}$	$33.82_{\pm0.00}$	$41.01_{\pm0.88}$	$38.55_{\pm0.63}$	$38.19_{\pm0.96}$	$42.94_{\pm0.00}$	$38.04_{\pm0.00}$	$42.91_{\pm0.00}$	$\mathbf{43.72}_{\pm0.00}$	$[\mathbf{49.79}_{\pm1.95}]$
Citeseer	$0.06_{\pm0.05}$	$0.09_{\pm0.03}$	$0.10_{\pm0.00}$	$25.34_{\pm0.70}$	$\mathbf{26.69}_{\pm0.60}$	$24.27_{\pm0.71}$	$24.61_{\pm0.00}$	$1.15_{\pm0.00}$	$14.86_{\pm0.00}$	$11.74_{\pm0.00}$	$[\mathbf{31.18}_{\pm2.26}]$
NH-4660	$48.17_{\pm23.18}$	$2.76_{\pm2.15}$	$2.88_{\pm0.00}$	$\mathbf{83.05}_{\pm5.29}$	$82.23_{\pm0.46}$	$81.26_{\pm4.20}$	$68.54_{\pm0.00}$	$62.11_{\pm0.00}$	$61.11_{\pm0.00}$	$60.93_{\pm0.00}$	$[\mathbf{92.58}_{\pm7.71}]$
ALOI	$5.97_{\pm1.23}$	$11.69_{\pm2.23}$	N/A	N/A	N/A	N/A	$42.71_{\pm0.00}$	$\mathbf{44.95}_{\pm0.00}$	$16.98_{\pm0.00}$	$16.07_{\pm0.00}$	$[\mathbf{66.40}_{\pm1.50}]$
NUS-WIDE	N/A	N/A	N/A	N/A	N/A	N/A	$\mathbf{6.17}_{\pm0.00}$	$2.40_{\pm0.00}$	$5.31_{\pm0.00}$	$[\mathbf{6.39}_{\pm0.00}]$	$5.07_{\pm0.41}$
VGGFace2-50	N/A	N/A	N/A	N/A	N/A	N/A	$\mathbf{3.68}_{\pm0.00}$	$3.35_{\pm0.00}$	$2.70_{\pm0.00}$	$3.21_{\pm0.00}$	$[\mathbf{4.32}_{\pm0.15}]$
VGGFace2-100	N/A	N/A	N/A	N/A	N/A	N/A	$2.00_{\pm0.00}$	$\mathbf{2.46}_{\pm0.00}$	$1.98_{\pm0.00}$	$2.06_{\pm0.00}$	$[\mathbf{2.47}_{\pm0.09}]$
VGGFace2-200	N/A	N/A	N/A	N/A	N/A	N/A	$1.06_{\pm0.00}$	$\mathbf{1.28}_{\pm0.00}$	$0.92_{\pm0.00}$	$0.86_{\pm0.00}$	$[\mathbf{1.29}_{\pm0.05}]$
CIFAR-10	N/A	N/A	N/A	N/A	N/A	N/A	$10.95_{\pm0.00}$	$8.09_{\pm0.00}$	$12.56_{\pm0.00}$	$\mathbf{12.63}_{\pm0.00}$	$[\mathbf{13.56}_{\pm0.49}]$
CIFAR-100	N/A	N/A	N/A	N/A	N/A	N/A	$3.39_{\pm0.00}$	$2.14_{\pm0.00}$	$3.22_{\pm0.00}$	$3.04_{\pm0.00}$	$[\mathbf{3.55}_{\pm0.14}]$
YTF-10	N/A	N/A	N/A	N/A	N/A	N/A	$69.92_{\pm0.00}$	$56.96_{\pm0.00}$	$66.04_{\pm0.00}$	$65.04_{\pm0.00}$	$[\mathbf{70.79}_{\pm5.57}]$
YTF-20	N/A	N/A	N/A	N/A	N/A	N/A	$52.56_{\pm0.00}$	$56.35_{\pm0.00}$	$[\mathbf{66.02}_{\pm0.00}]$	$61.50_{\pm0.00}$	$63.50_{\pm4.48}$
YTF-50	N/A	N/A	N/A	N/A	N/A	N/A	$60.32_{\pm0.00}$	$56.97_{\pm0.00}$	$60.10_{\pm0.00}$	$58.05_{\pm0.00}$	$[\mathbf{61.24}_{\pm2.34}]$
YTF-100	N/A	N/A	N/A	N/A	N/A	N/A	$58.51_{\pm0.00}$	$52.22_{\pm0.00}$	$48.48_{\pm0.00}$	$24.21_{\pm0.00}$	$[\mathbf{60.00}_{\pm2.27}]$
YTF-200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$49.30_{\pm0.00}$	$35.82_{\pm0.00}$	$19.43_{\pm0.00}$	$[\mathbf{56.83}_{\pm0.95}]$
YTF-400	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$[\mathbf{51.49}_{\pm1.14}]$
Average Rank	6.55	6.45	6.27	4.64	5.73	5.77	4.55	5.00	4.41	5.27	1.27

* Note that N/A indicates the out-of-memory error.

- *MVEC* [33]: multi-view ensemble clustering.
- M^2VEC_{km} [7]: marginalized multi-view ensemble clustering with k-means
- M^2VEC_{spec} [7]: marginalized multi-view ensemble clustering with spectral clustering
- *BMVC* [21]: binary multi-view clustering.
- *LMVSC* [18]: large-scale multi-view subspace clustering.
- *SMVSC* [19]: scalable multi-view subspace clustering.
- *FPMVS-CAG* [20]: fast parameter-free multi-view subspace clustering with consensus anchor guidance.

Among these baseline methods, BMVC, LMVSC, SMVSC, and FPMVS-CAG are four large-scale MVC methods, MVEC, M^2VEC_{km}, and M^2VEC_{spec} are three EC based MVC methods, and AMGL, SwMC, and FPMVS-CAG are three tuning-free MVC methods. For the baseline methods as well as the proposed method, if the distance metric can be customized, then the cosine distance will be adopted for the document datasets, such as *Movies*, *BBCSport*, and *Citeseer*. Otherwise, their suggested distance (mostly euclidean distance) will be adopted.

For each of the baseline methods, if the dataset-specific tuning is needed, then each of its hyperparameters will be tuned in the range of $\{10^{-5}, 10^{-4}, \ldots, 10^5\}$, unless the tuning range is specifically suggested by the corresponding paper. To

TABLE V
AVERAGE ACC(%) SCORES OVER 20 RUNS BY DIFFERENT MULTI-VIEW CLUSTERING ALGORITHMS. ON EACH DATASET, THE BEST TWO SCORES ARE HIGHLIGHTED IN **BOLD**, WHILE THE BEST ONE IN [**BOLD AND BRACKETS**]

Dataset	MVSC	AMGL	SwMC	MVEC	M^2VEC_{km}	M^2VEC_{spec}	BMVC	LMVSC	SMVSC	FPMVS-CAG	FastMICE
MSRCv1	$56.98_{\pm5.33}$	$64.90_{\pm8.10}$	$70.48_{\pm0.00}$	$70.81_{\pm1.75}$	$67.81_{\pm2.22}$	$67.90_{\pm2.21}$	$64.76_{\pm0.00}$	$36.67_{\pm0.00}$	$67.14_{\pm0.00}$	$60.00_{\pm0.00}$	$[79.64_{\pm5.18}]$
Yale	$59.03_{\pm4.26}$	$60.12_{\pm3.74}$	$61.21_{\pm0.00}$	$62.21_{\pm1.79}$	$45.18_{\pm8.23}$	$44.03_{\pm3.86}$	$22.42_{\pm0.00}$	$55.15_{\pm0.00}$	$56.97_{\pm0.00}$	$44.24_{\pm0.00}$	$[64.21_{\pm3.69}]$
ORL	$72.36_{\pm3.81}$	$72.61_{\pm2.25}$	$70.75_{\pm0.00}$	$68.60_{\pm1.43}$	$57.55_{\pm2.38}$	$56.45_{\pm4.81}$	$16.75_{\pm0.00}$	$57.00_{\pm0.00}$	$56.25_{\pm0.00}$	$56.00_{\pm0.00}$	$[73.05_{\pm2.63}]$
Movies	$25.19_{\pm1.31}$	$26.17_{\pm1.39}$	$20.91_{\pm0.00}$	$27.07_{\pm0.83}$	$27.49_{\pm0.75}$	$27.78_{\pm0.98}$	$[29.34_{\pm0.00}]$	$26.74_{\pm0.00}$	$21.39_{\pm0.00}$	$17.02_{\pm0.00}$	$28.77_{\pm2.06}$
BBCSport	$77.67_{\pm12.79}$	$94.43_{\pm7.15}$	$62.87_{\pm0.00}$	$[96.96_{\pm0.41}]$	$96.50_{\pm0.41}$	$96.50_{\pm0.43}$	$75.74_{\pm0.00}$	$94.67_{\pm0.00}$	$32.35_{\pm0.00}$	$40.63_{\pm0.00}$	$95.15_{\pm5.54}$
NH-550	$81.20_{\pm5.86}$	$67.94_{\pm11.39}$	$87.82_{\pm0.00}$	$89.95_{\pm4.43}$	$84.43_{\pm9.12}$	$84.55_{\pm8.81}$	$27.27_{\pm0.00}$	$80.73_{\pm0.00}$	$91.45_{\pm0.00}$	$70.55_{\pm0.00}$	$[93.98_{\pm9.26}]$
OutdoorScene	$31.82_{\pm10.36}$	$49.91_{\pm5.12}$	$45.46_{\pm0.00}$	$62.55_{\pm0.92}$	$59.58_{\pm0.50}$	$59.69_{\pm1.17}$	$63.62_{\pm0.00}$	$59.86_{\pm0.00}$	$63.32_{\pm0.00}$	$63.58_{\pm0.00}$	$[69.85_{\pm3.57}]$
Citeseer	$21.24_{\pm0.17}$	$21.48_{\pm0.06}$	$21.62_{\pm0.00}$	$55.57_{\pm0.48}$	$56.20_{\pm0.42}$	$54.84_{\pm0.47}$	$54.41_{\pm0.00}$	$29.35_{\pm0.00}$	$41.94_{\pm0.00}$	$37.53_{\pm0.00}$	$[59.32_{\pm1.60}]$
NH-4660	$62.93_{\pm14.53}$	$36.29_{\pm1.84}$	$33.88_{\pm0.00}$	$88.09_{\pm4.27}$	$86.00_{\pm1.43}$	$85.70_{\pm2.74}$	$85.94_{\pm0.00}$	$73.76_{\pm0.00}$	$71.61_{\pm0.00}$	$71.31_{\pm0.00}$	$[94.38_{\pm7.13}]$
ALOI	$12.36_{\pm0.83}$	$60.00_{\pm2.06}$	N/A	N/A	N/A	N/A	$60.39_{\pm0.00}$	$56.61_{\pm0.00}$	$34.30_{\pm0.00}$	$31.53_{\pm0.00}$	$[75.62_{\pm1.58}]$
NUS-WIDE	N/A	N/A	N/A	N/A	N/A	N/A	$14.32_{\pm0.00}$	$12.06_{\pm0.00}$	$17.75_{\pm0.00}$	$[19.41_{\pm0.00}]$	$14.44_{\pm0.58}$
VGGFace2-50	N/A	N/A	N/A	N/A	N/A	N/A	$11.86_{\pm0.00}$	$11.55_{\pm0.00}$	$9.83_{\pm0.00}$	$10.74_{\pm0.00}$	$[12.65_{\pm0.26}]$
VGGFace2-100	N/A	N/A	N/A	N/A	N/A	N/A	$7.37_{\pm0.00}$	$[8.46_{\pm0.00}]$	$6.56_{\pm0.00}$	$7.21_{\pm0.00}$	$8.09_{\pm0.26}$
VGGFace2-200	N/A	N/A	N/A	N/A	N/A	N/A	$4.68_{\pm0.00}$	$[5.14_{\pm0.00}]$	$4.16_{\pm0.00}$	$4.22_{\pm0.00}$	$5.08_{\pm0.16}$
CIFAR-10	N/A	N/A	N/A	N/A	N/A	N/A	$28.03_{\pm0.00}$	$25.81_{\pm0.00}$	$29.20_{\pm0.00}$	$29.55_{\pm0.00}$	$[33.28_{\pm1.24}]$
CIFAR-100	N/A	N/A	N/A	N/A	N/A	N/A	$8.95_{\pm0.00}$	$7.96_{\pm0.00}$	$8.14_{\pm0.00}$	$9.06_{\pm0.00}$	$[10.31_{\pm0.26}]$
YTF-10	N/A	N/A	N/A	N/A	N/A	N/A	$[78.62_{\pm0.00}]$	$69.32_{\pm0.00}$	$73.81_{\pm0.00}$	$71.49_{\pm0.00}$	$76.74_{\pm4.39}$
YTF-20	N/A	N/A	N/A	N/A	N/A	N/A	$60.79_{\pm0.00}$	$63.88_{\pm0.00}$	$[74.79_{\pm0.00}]$	$68.87_{\pm0.00}$	$71.47_{\pm3.78}$
YTF-50	N/A	N/A	N/A	N/A	N/A	N/A	$[70.06_{\pm0.00}]$	$69.49_{\pm0.00}$	$67.44_{\pm0.00}$	$63.73_{\pm0.00}$	$69.21_{\pm2.30}$
YTF-100	N/A	N/A	N/A	N/A	N/A	N/A	$65.67_{\pm0.00}$	$60.71_{\pm0.00}$	$59.47_{\pm0.00}$	$53.49_{\pm0.00}$	$[66.83_{\pm1.67}]$
YTF-200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$55.00_{\pm0.00}$	$52.88_{\pm0.00}$	$50.45_{\pm0.00}$	$[62.79_{\pm0.86}]$
YTF-400	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$[56.40_{\pm0.82}]$
Average Rank	6.59	6.14	6.27	4.68	5.41	5.64	4.45	4.82	4.82	5.55	1.55

* Note that N/A indicates the out-of-memory error.

TABLE VI
AVERAGE PUR(%) SCORES OVER 20 RUNS BY DIFFERENT MULTI-VIEW CLUSTERING ALGORITHMS. ON EACH DATASET, THE BEST TWO SCORES ARE HIGHLIGHTED IN **BOLD**, WHILE THE BEST ONE IN [**BOLD AND BRACKETS**]

Dataset	MVSC	AMGL	SwMC	MVEC	M^2VEC_{km}	M^2VEC_{spec}	BMVC	LMVSC	SMVSC	FPMVS-CAG	FastMICE
MSRCv1	$59.31_{\pm5.06}$	$68.10_{\pm7.17}$	$73.33_{\pm0.00}$	$74.29_{\pm1.96}$	$71.79_{\pm2.05}$	$71.69_{\pm2.31}$	$66.67_{\pm0.00}$	$40.95_{\pm0.00}$	$69.05_{\pm0.00}$	$62.86_{\pm0.00}$	$[81.98_{\pm3.13}]$
Yale	$60.33_{\pm3.59}$	$61.27_{\pm2.86}$	$61.82_{\pm0.00}$	$63.70_{\pm1.58}$	$46.94_{\pm7.71}$	$46.39_{\pm3.32}$	$24.24_{\pm0.00}$	$56.97_{\pm0.00}$	$56.97_{\pm0.00}$	$46.67_{\pm0.00}$	$[65.27_{\pm3.10}]$
ORL	$77.05_{\pm2.47}$	$[78.08_{\pm1.87}]$	$76.75_{\pm0.00}$	$75.41_{\pm1.08}$	$67.85_{\pm1.60}$	$62.78_{\pm3.65}$	$18.00_{\pm0.00}$	$63.50_{\pm0.00}$	$60.25_{\pm0.00}$	$60.00_{\pm0.00}$	$77.33_{\pm2.03}$
Movies	$28.11_{\pm1.37}$	$28.50_{\pm1.19}$	$23.34_{\pm0.00}$	$30.26_{\pm0.77}$	$30.86_{\pm0.57}$	$31.15_{\pm1.25}$	$[32.25_{\pm0.00}]$	$28.36_{\pm0.00}$	$23.50_{\pm0.00}$	$18.64_{\pm0.00}$	$32.00_{\pm1.87}$
BBCSport	$81.53_{\pm9.87}$	$94.99_{\pm5.92}$	$64.15_{\pm0.00}$	$[96.96_{\pm0.41}]$	$96.50_{\pm0.41}$	$96.50_{\pm0.43}$	$85.29_{\pm0.00}$	$94.67_{\pm0.00}$	$42.83_{\pm0.00}$	$48.16_{\pm0.00}$	$95.89_{\pm3.27}$
NH-550	$83.55_{\pm6.19}$	$69.02_{\pm10.93}$	$87.82_{\pm0.00}$	$89.95_{\pm4.43}$	$86.89_{\pm4.76}$	$87.01_{\pm4.54}$	$35.64_{\pm0.00}$	$84.00_{\pm0.00}$	$91.45_{\pm0.00}$	$82.73_{\pm0.00}$	$[96.02_{\pm4.77}]$
OutdoorScene	$32.03_{\pm10.40}$	$50.47_{\pm5.32}$	$45.57_{\pm0.00}$	$62.80_{\pm0.32}$	$62.00_{\pm0.47}$	$61.77_{\pm0.47}$	$63.62_{\pm0.00}$	$59.86_{\pm0.00}$	$66.00_{\pm0.00}$	$67.08_{\pm0.00}$	$[70.50_{\pm2.73}]$
Citeseer	$21.39_{\pm0.11}$	$21.57_{\pm0.05}$	$22.04_{\pm0.00}$	$57.22_{\pm0.46}$	$57.90_{\pm0.41}$	$56.44_{\pm0.47}$	$56.40_{\pm0.00}$	$31.76_{\pm0.00}$	$47.01_{\pm0.00}$	$42.81_{\pm0.00}$	$[61.48_{\pm1.78}]$
NH-4660	$65.24_{\pm13.91}$	$37.07_{\pm2.10}$	$37.10_{\pm0.00}$	$88.23_{\pm3.68}$	$87.24_{\pm0.01}$	$86.78_{\pm1.96}$	$85.94_{\pm0.00}$	$81.20_{\pm0.00}$	$78.41_{\pm0.00}$	$78.50_{\pm0.00}$	$[95.41_{\pm4.40}]$
ALOI	$13.57_{\pm0.90}$	$63.44_{\pm1.55}$	N/A	N/A	N/A	N/A	$63.27_{\pm0.00}$	$59.76_{\pm0.00}$	$35.28_{\pm0.00}$	$32.13_{\pm0.00}$	$[76.98_{\pm1.35}]$
NUS-WIDE	N/A	N/A	N/A	N/A	N/A	N/A	$[28.57_{\pm0.00}]$	$20.09_{\pm0.00}$	$22.81_{\pm0.00}$	$23.61_{\pm0.00}$	$24.75_{\pm0.57}$
VGGFace2-50	N/A	N/A	N/A	N/A	N/A	N/A	$12.87_{\pm0.00}$	$12.29_{\pm0.00}$	$10.28_{\pm0.00}$	$11.15_{\pm0.00}$	$[13.67_{\pm0.27}]$
VGGFace2-100	N/A	N/A	N/A	N/A	N/A	N/A	$8.19_{\pm0.00}$	$[9.28_{\pm0.00}]$	$6.74_{\pm0.00}$	$7.27_{\pm0.00}$	$9.04_{\pm0.24}$
VGGFace2-200	N/A	N/A	N/A	N/A	N/A	N/A	$5.40_{\pm0.00}$	$5.71_{\pm0.00}$	$4.37_{\pm0.00}$	$4.26_{\pm0.00}$	$[5.79_{\pm0.18}]$
CIFAR-10	N/A	N/A	N/A	N/A	N/A	N/A	$30.55_{\pm0.00}$	$27.48_{\pm0.00}$	$32.40_{\pm0.00}$	$32.85_{\pm0.00}$	$[35.28_{\pm0.73}]$
CIFAR-100	N/A	N/A	N/A	N/A	N/A	N/A	$10.05_{\pm0.00}$	$9.37_{\pm0.00}$	$8.64_{\pm0.00}$	$9.45_{\pm0.00}$	$[11.71_{\pm0.26}]$
YTF-10	N/A	N/A	N/A	N/A	N/A	N/A	$81.40_{\pm0.00}$	$73.09_{\pm0.00}$	$78.89_{\pm0.00}$	$76.22_{\pm0.00}$	$[81.95_{\pm3.03}]$
YTF-20	N/A	N/A	N/A	N/A	N/A	N/A	$69.33_{\pm0.00}$	$72.33_{\pm0.00}$	$[77.71_{\pm0.00}]$	$75.06_{\pm0.00}$	$77.08_{\pm2.61}$
YTF-50	N/A	N/A	N/A	N/A	N/A	N/A	$[75.37_{\pm0.00}]$	$73.61_{\pm0.00}$	$70.84_{\pm0.00}$	$66.90_{\pm0.00}$	$75.18_{\pm1.36}$
YTF-100	N/A	N/A	N/A	N/A	N/A	N/A	$[74.76_{\pm0.00}]$	$69.37_{\pm0.00}$	$62.21_{\pm0.00}$	$54.76_{\pm0.00}$	$73.59_{\pm1.43}$
YTF-200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$65.14_{\pm0.00}$	$57.19_{\pm0.00}$	$51.55_{\pm0.00}$	$[70.94_{\pm0.55}]$
YTF-400	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	$[68.11_{\pm0.39}]$
Average Rank	6.68	6.00	6.23	4.68	5.27	5.64	4.41	4.95	4.95	5.64	1.45

* Note that N/A indicates the out-of-memory error.

avoid the expensive or even unaffordable computational costs of hyperparameter-tuning on the entire large-scale datasets, for all the baseline methods except M^2VEC_{km} and M^2VEC_{spec}, if $N > 10,000$, the tuning will be conducted on a random subset of 10,000 samples. For M^2VEC_{km} and M^2VEC_{spec}, whose computational costs rapidly increase with larger data sizes, if $N > 1,000$, the tuning will be conducted on a random subset of 1,000 samples.

Note that the proposed FastMICE method does not require dataset-specific tuning. Though there exist several parameters in FastMICE, yet these parameters can safely be set to some common values (or randomized in some common ranges) across

various datasets. Specifically, the feature sampling ratio $\tau_v^{(m)}$ for each view member is randomized in [0.2, 0.8]. The number of clusters $k^{(m)}$ in each base clustering is randomized in $[k, 2k]$, where k is the desired number of clusters in the final clustering. Besides these randomizations, the number of base clusterings $M = 20$, the number of anchors $p = \min\{1,000, N\}$, and the number of nearest neighbors $K = 5$ are used in the experiments on all benchmark datasets.

To evaluate the clustering performances of different MVC methods, we adopt four wide-used evaluation metrics in our experiments, namely, normalized mutual information (NMI) [32], adjusted Rand index (ARI) [31], accuracy (ACC) [38], and
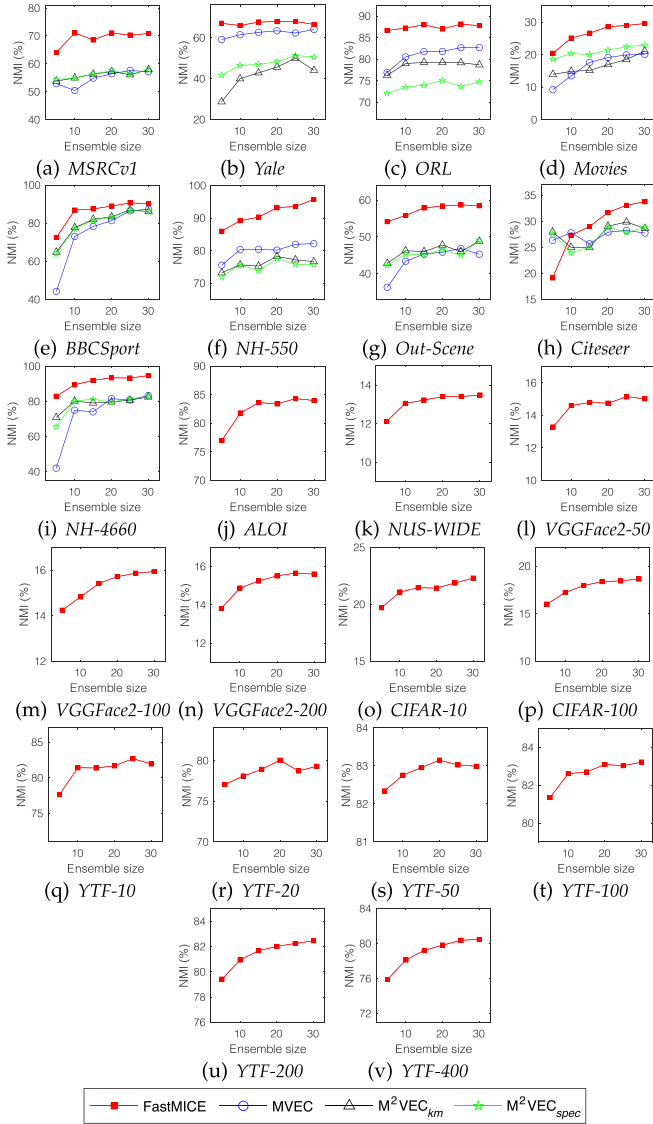
Fig. 3. The average NMI (%) scores over 20 runs by different EC based MVC algorithms with varying ensemble sizes M. Note that if an algorithm is not computationally feasible on a dataset, then its curve will not appear in the corresponding sub-figure.

Fig. 4. The average NMI (%) scores over 20 runs by the proposed algorithm with varying number of anchors p.

purity (PUR) [38]. Larger values of these metrics indicate better clustering results.

C. Performance Comparison and Analysis

In this section, we experimentally compare our FastMICE method against ten baseline MVC methods. The comparison results w.r.t. NMI, ARI, ACC, and PUR are reported in Tables III, IV, V, and VI, respectively.

As shown in Table III, FastMICE achieves the best performance w.r.t. NMI on 19 out of the 22 datasets. Though the MVEC method yields a higher NMI score than FastMICE on the *BBCSport* dataset, yet FastMICE outperforms or significantly outperforms MVEC on all the other datasets. In comparison with the four large-scale baseline methods, namely, BMVC, LMVSC, SMVSC, and FPMVS-CAG, our FastMICE method

outperforms these methods on all benchmark datasets except *Movies* and *NUS-WIDE*.

Further, we report the average ranks (across the twenty datasets) of the proposed method and the ten baseline methods in Table III. Note that, for a dataset, if four MVC methods are computationally feasible and seven MVC methods are computationally infeasible due to the out-of-memory error, then the seven infeasible methods will equally rank in the fifth position on this dataset. As can be seen in Table III, the proposed FastMICE method achieves an average rank of 1.23, which significantly outperforms the second best method with an average rank of 4.41.

Similar advantages of the proposed method over the baselines can also be observed in Tables IV, V, and VI. In terms of ARI, our FastMICE method achieves the best scores on 18 out of the 22 datasets, with an average rank of 1.27. In terms of ACC, our FastMICE method achieves the best scores on 14 out of the 22 datasets, with an average rank of 1.55. In terms of PUR, our
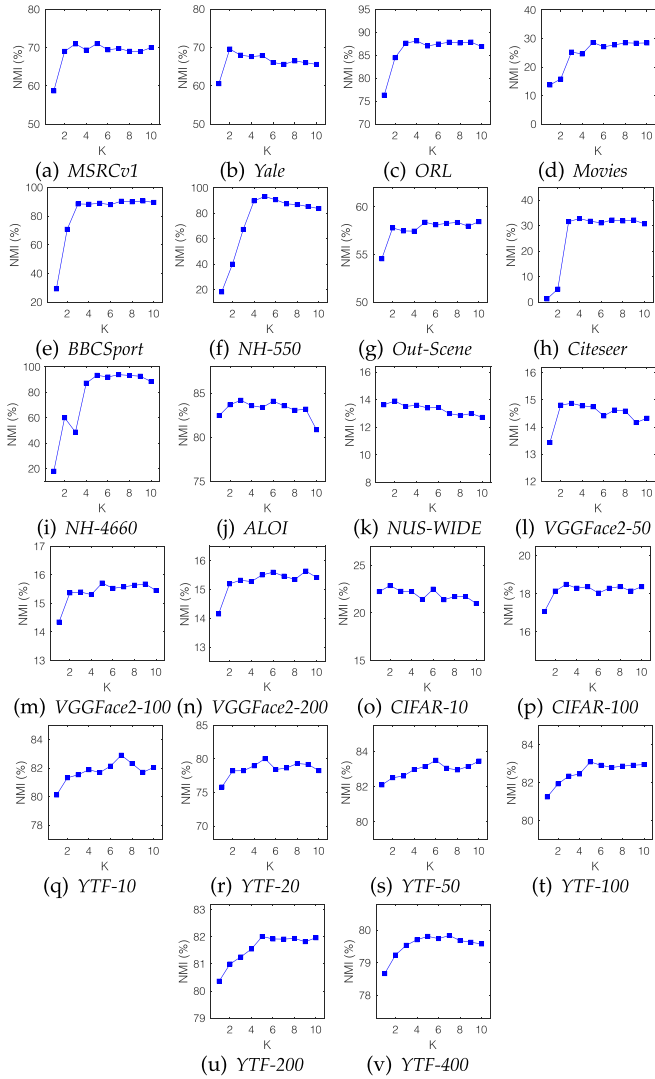
Fig. 5.   The average NMI (%) scores over 20 runs by the proposed algorithm with varying number of nearest neighbors K.

Fig. 6.   The comparison (w.r.t. NMI (%)) of the proposed algorithm using random view groups against treating each single view as a group and treating all views as a group.

FastMICE method achieves the best scores on 14 out of the twenty datasets, with an average rank of 1.45. The experimental results in Tables III, IV, V, and VI have confirmed the robust performance of our FastMICE method when compared with the other MVC methods.

D. Influence of Ensemble Size M

In this section, we test the influence of the ensemble size M, which corresponds to the number of base clusterings and also the number of random view groups in FastMICE. The three EC based baseline methods, namely, MVEC, M ^{2} VEC$_{km}$, and M ^{2} VEC$_{spec}$, similarly involve the ensemble size M, which will also be tested in the comparison.

The performances of FastMICE and the three baselines are illustrated in Fig. 3. Note that the three EC based baseline methods are not computationally feasible for datasets larger than *NH-4660*, so their curves will be absent in the corresponding sub-figures. As shown in Fig. 3, the proposed FastMICE method

yields consistently high-quality clustering performance with varying ensemble sizes. When compared with the other EC based methods, FastMICE achieves overall better performance than the baselines on the benchmark datasets. Empirically, a relatively larger ensemble size is beneficial. In our experiments, we use the ensemble size $M = 20$ on all benchmark datasets.

E. Influence of Number of Anchors p

In this section, we test the influence of the number of anchors p in the FastMICE method. As can be seen in Fig. 4, the proposed FastMICE method shows consistent performance as the number of anchors goes from 100 to 1400. Empirically, a larger number of anchors can be beneficial on most of the datasets, especially on the large-scale ones such as *YTF-200* and *YTF-400*, probably due to the fact that a larger number of anchors may better reflect the overall structure of the data. As the number of anchors cannot exceed the number of original samples, in our experiments, we use the number of anchors $p = \min\{1000, N\}$ on all benchmark datasets.

TABLE VII
TIME COSTS (IN SECONDS) OF DIFFERENT MULTI-VIEW CLUSTERING ALGORITHMS. **TUNETIME**: TIME COST FOR DATASET-SPECIFIC HYPERPARAMETER TUNING.
RUNTIME: TIME COST FOR RUNNING THE ALGORITHM

Dataset		MVSC	AMGL	SwMC	MVEC	M^2VEC_{km}	M^2VEC_{spec}	BMVC	LMVSC	SMVSC	FPMVS-CAG	FastMICE
MSRCv1	TuneTime	2.10	-	-	226.19	8,252.72		54.95	8.65	3.75	-	-
	RunTime	0.26	0.16	2.46	3.19	12.83	12.86	0.06	0.46	1.17	2.31	0.33
Yale	TuneTime	5.07	-	-	141.30	3,663.22		98.64	7.75	8.55	-	-
	RunTime	0.53	0.08	1.40	4.78	4.65	4.70	0.08	0.56	2.57	3.21	1.81
ORL	TuneTime	27.36	-	-	1,442.16	55,123.16		233.75	16.63	56.55	-	-
	RunTime	2.77	0.40	7.39	31.40	116.87	116.92	0.24	1.50	10.73	22.90	6.67
Movies	TuneTime	13.86	-	-	3,823.81	94,912.10		297.74	16.96	13.84	-	-
	RunTime	1.33	1.59	15.54	54.12	178.80	179.01	0.21	0.74	4.83	7.54	3.18
BBCSport	TuneTime	18.33	-	-	2,267.14	111,283.96		234.11	13.20	11.18	-	-
	RunTime	1.81	0.91	16.48	32.82	178.83	179.00	0.22	0.75	3.94	7.09	6.85
NH-550	TuneTime	33.17	-	-	2,350.40	117,688.03		364.87	19.12	12.75	-	-
	RunTime	3.23	1.53	22.92	36.27	208.96	209.08	0.46	1.78	4.10	7.98	6.38
Out-Scene	TuneTime	92.35	-	-	218,154.40	1,490,922.44		1,857.73	79.59	47.82	-	-
	RunTime	9.20	53.79	1,644.44	4,495.78	34,118.99	34,140.06	1.31	4.97	14.63	29.12	4.93
Citeseer	TuneTime	179.54	-	-	37,1374.01	1,131,867.93		2,209.50	117.07	66.03	-	-
	RunTime	17.50	191.80	816.35	7,984.33	46,788.20	46,826.12	1.25	8.10	21.41	42.38	28.26
NH-4660	TuneTime	522.87	-	-	1,256,609.02	1,418,874.08		3,962.23	249.21	103.99	-	-
	RunTime	52.98	388.50	5,709.74	19,621.66	142,392.16	142,472.40	2.20	23.75	33.95	66.69	27.60
ALOI	TuneTime	2,073.97	-	N/A	N/A	N/A		5,822.19	286.58	1,355.84	-	-
	RunTime	205.71	4,760.38	N/A	N/A	N/A	N/A	4.23	25.46	124.94	241.03	10.32
NUS-WIDE	TuneTime	N/A	N/A	N/A	N/A	N/A		6,478.61	375.52	267.60	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	12.73	91.99	203.15	386.47	14.43
VGGFace2-50	TuneTime	N/A	N/A	N/A	N/A	N/A		5,677.58	348.02	587.33	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	13.34	249.97	491.59	571.76	30.69
VGGFace2-100	TuneTime	N/A	N/A	N/A	N/A	N/A		5,870.30	400.46	2,181.60	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	26.12	663.49	3,989.61	1,788.38	62.63
VGGFace2-200	TuneTime	N/A	N/A	N/A	N/A	N/A		6,334.40	853.80	9,613.44	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	53.58	1,091.23	34,208.76	14,987.48	157.55
CIFAR-10	TuneTime	N/A	N/A	N/A	N/A	N/A		5,930.36	388.57	200.48	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	22.75	303.92	376.17	719.82	47.53
CIFAR-100	TuneTime	N/A	N/A	N/A	N/A	N/A		6,649.10	681.23	2,056.21	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	24.00	625.90	3,569.92	1,680.07	58.84
YTF-10	TuneTime	N/A	N/A	N/A	N/A	N/A		5,786.88	315.67	198.01	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	13.74	198.22	258.70	460.84	23.53
YTF-20	TuneTime	N/A	N/A	N/A	N/A	N/A		5,537.66	335.58	236.48	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	22.69	324.86	591.16	831.06	35.05
YTF-50	TuneTime	N/A	N/A	N/A	N/A	N/A		5,864.39	261.78	549.79	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	47.57	751.22	1,107.12	2,143.80	70.89
YTF-100	TuneTime	N/A	N/A	N/A	N/A	N/A		6,131.04	323.94	1,970.67	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	72.69	1,242.44	2,863.95	5,415.22	124.48
YTF-200	TuneTime	N/A	N/A	N/A	N/A	N/A		N/A	816.28	8,587.07	-	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3,184.79	74,063.30	32,974.03	263.81
YTF-400	TuneTime	N/A	N/A	N/A	N/A	N/A		N/A	N/A	N/A	N/A	-
	RunTime	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	652.05

[*] Note that N/A indicates the out-of-memory error, and the symbol "-" indicates that no dataset-specific tuning is needed.
[**] If the dataset-specific tuning is needed, the hyperparameters will be tuned in $\{10^{-5}, 10^{-4}, \cdots, 10^{5}\}$, unless the tuning range is specifically given by the corresponding paper.
[***] For the baseline methods except M^2VEC_{km} and M^2VEC_{spec}, if $N > 10,000$, the tuning (if needed) will be conducted on a random subset of 10,000 samples. For M^2VEC_{km} and M^2VEC_{spec}, if $N > 1,000$, the tuning will be conducted on a random subset of 1,000 samples.
[****] The difference between M^2VEC_{km} and M^2VEC_{spec} lies in the last step, where either k-means or spectral clustering is used. Thus we tune them together.

F. Influence of Number of Nearest Neighbors K

In this section, we test the influence of the number of nearest neighbors K in the FastMICE method, which corresponds to the number of (nearest) anchors that are linked to each data sample. Specifically, we illustrate the performance of the proposed Fast-MICE method as the number of nearest neighbors goes from 1 to 10 in Fig. 5. As shown in Fig. 5, a moderate value of K can often be beneficial on most of the benchmark datasets. Empirically, it is suggested that the number of nearest neighbors be set in the range of [4,8]. In our experiments, we use the number of nearest neighbors $K = 5$ on all benchmark datasets.

G. Influence of Random View Groups

In this section, we test the influence of the random view groups in our framework. As discussed in Section III-B, the previous early fusion methods can be regarded as a special instance of our view group formation with all views in a single group, while the late fusion methods can also be regarded as a special instance of our view group formation with a single view in each group. In this section, we compare our FastMICE method using random view groups against using a single view in each group and using all views in a group. As shown in Fig. 6, the use of random view groups leads to substantial improvements on most of the datasets. Though the use of all views in a group leads to comparable performance to the use of random view groups on the *Yale* dataset, yet on most of the other datasets the FastMICE method using random view groups outperforms or significantly outperforms the variants using a single view in each group or using all views in a group, which verifies the benefits brought in by the random view groups.

H. Execution Time

In this section, we evaluate the time costs of the FastMICE method and the other MVC methods on the benchmark datasets. Note that for the proposed FastMICE method, no dataset-specific hyperparameter-tuning is needed. For the baseline methods, if the dataset-specific hyperparameter-tuning is required, then the time costs of tuning and running will be respectively reported.

As shown in Table VII, more than half of the baseline methods cannot go beyond the *ALOI* dataset due to the computational complexity bottleneck. Though the large-scale baseline methods, including BMVC, LMVSC, SMVSC, and FPMVS-CAG, have shown their scalability for some larger datasets, yet they still encounter heavy computational burdens or even the out-of-memory error when they process the *YTF-200* or *YTF-400* datasets. Remarkably, on the *YTF-200* dataset with 286,006 samples, our FastMICE method only consumes 263.81 seconds of running time, while the LMVSC, SMVSC, and FPMVS-CAG methods respectively consume 3,184.79 seconds, 74,063.30 seconds, and 32,974.03 seconds of running times. In terms of the *YTF-400* dataset with 398,191 samples, FastMICE is the only method that is computationally feasible on this large-scale dataset, which demonstrates the clear advantage of our Fast-MICE method in scalability for very large-scale datasets.

To summarize, as shown in Tables III, IV, V, VI, and VII, the proposed FastMICE method is capable of yielding highly competitive clustering performance over the state-of-the-art, while showing advantageous scalability on very large-scale multi-view datasets.

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a new large-scale MVC approach termed FastMICE, which is featured by its scalability (for extremely large-scale datasets), superiority (in clustering performance), and simplicity (to be applied without dataset-specific tuning). In particular, different from the previous approaches that mostly adopt some types of single-stage fusion strategies, this paper presents a hybrid early-late fusion strategy based on the random view groups. A large number of random view groups are first formed to serve as a flexible view-organizations to investigate the view-wise relationships. Then, three levels of diversity, i.e., the feature-level diversity, the anchor-level diversity, and the neighborhood-level diversity, are jointly leveraged to explore the rich and versatile information in the random view groups and thereby enable the highly efficient construction of the view-sharing bipartite graphs. By fast partitioning of these view-sharing bipartite graphs from different view groups, a set of diversified base clusterings can be generated, which are further formulated into a unified bipartite graph for achieving the final clustering result.

It is noteworthy that our FastMICE approach has almost linear time and space complexity, and is able to perform robustly and accurately on various general-scale and large-scale datasets without requiring dataset-specific hyperparameter-tuning. Extensive experimental results on 22 multi-view datasets have demonstrated the superiority of our FastMICE approach over the state-of-the-art. In the future work, the concept of random view groups and the diversification-and-fusion strategy may also be investigated for more MVC tasks, such as the incomplete MVC task [43] and the deep MVC task [44], [45], so as to promote their robustness while ensuring scalability for very large datasets.

REFERENCES

[1] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[2] G. Chao, S. Sun, and J. Bi, "A survey on multiview clustering," *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 146–168, Apr. 2021.

[3] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2149–2155.

[4] 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. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1881–1887.

[5] Y. Liang, D. Huang, and C.-D. Wang, "Consistency meets inconsistency: A unified graph learning framework for multi-view clustering," in *Proc. IEEE Int. Conf. Data Mining*, 2019, pp. 1204–1209.

[6] Z. Kang et al., "Partition level multiview subspace clustering," *Neural Netw.*, vol. 122, pp. 279–288, 2020.

[7] 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.

[8] W. Liang et al., "Multi-view spectral clustering with high-order optimal neighborhood laplacian matrix," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3418–3430, Jul. 2022.

[9] C. Zhang, H. Fu, S. Liu, G. Liu, and X. Cao, "Low-rank tensor constrained multiview subspace clustering," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1582–1590.

[10] 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.

[11] S. Wang et al., "Multi-view clustering via late fusion alignment maximization," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3778–3784.

[12] X. Liu et al., "One pass late fusion multi-view clustering," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6850–6859.

[13] Z. Li, J. Tang, and T. Mei, "Deep collaborative embedding for social image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2070–2083, Sep. 2019.

[14] X. Wang, L. Fu, Y. Zhang, Y. Wang, and Z. Li, "MMatch: Semi-supervised discriminative representation learning for multi-view classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 6425–6436, Sep. 2022.

[15] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1669–1680, Aug. 2015.

[16] 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.

[17] Y. Li, F. Nie, H. Huang, and J. Huang, "Large-Scale Multi-View Spectral Clustering Via Bipartite Graph," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 2750–2756.

[18] Z. Kang, W. Zhou, Z. Zhao, J. Shao, M. Han, and Z. Xu, "Large-Scale Multi-View Subspace Clustering in Linear Time," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4412–4419.

[19] M. Sun et al., "Scalable multi-view subspace clustering with unified anchors," in *Proc. ACM Int. Conf. Multimedia*, 2021, pp. 3528–3536.

[20] S. Wang et al., "Fast parameter-free multi-view subspace clustering with consensus anchor guidance," *IEEE Trans. Image Process.*, vol. 31, pp. 556–568, 2022.

[21] Z. Zhang, L. Liu, F. Shen, H. T. Shen, and L. Shao, "Binary multi-view clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1774–1782, Jul. 2019.

[22] Y. Xie, W. Zhang, Y. Qu, L. Dai, and D. Tao, "Hyper-laplacian regularized multilinear multiview self-representations for clustering and semisu-pervised learning," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 572–586, Feb. 2020.

[23] Y. Liang, D. Huang, C.-D. Wang, and P. S. Yu, "Multi-view graph learning by joint modeling of consistency and inconsistency," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 27, 2022, doi: 10.1109/TNNLS.2022.3192445.

[24] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.

[25] D. Huang, C. D. Wang, and J. H. Lai, "Locally weighted ensemble clustering," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1460–1473, May 2018.

[26] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.

[27] D. Huang, J. Lai, and C.-D. Wang, "Ensemble clustering using factor graph," *Pattern Recognit.*, vol. 50, pp. 131–142, 2016.

[28] D. Huang, J.-H. Lai, and C.-D. Wang, "Robust ensemble clustering using probability trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1312–1326, May 2016.

[29] Y. Ren, C. Domeniconi, G. Zhang, and G. Yu, "Weighted-object ensemble clustering: Methods and analysis," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 661–689, 2017.

[30] D. Huang, C.-D. Wang, H. Peng, J.-H. Lai, and C.-K. Kwoh, "Enhanced ensemble clustering via fast propagation of cluster-wise similarities," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 1, pp. 508–520, Jan. 2021.

[31] D. Huang, C.-D. Wang, J.-H. Lai, and C.-K. Kwoh, "Toward multi-diversified ensemble clustering of high-dimensional data: From subspaces to metrics and beyond," *IEEE Trans. Cybern.*, vol. 52, no. 11, pp. 12 231–12 244, Nov. 2022.

[32] A. Strehl and J. Ghosh, "Cluster ensembles: A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2003.

[33] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "From ensemble clustering to multi-view clustering," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 2843–2849.

[34] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2002, pp. 849–856.

[35] Z. Li, X.-M. Wu, and S.-F. Chang, "Segmentation using superpixels: A bipartite graph partitioning approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 789–796.

[36] M.-S. Chen, L. Huang, C.-D. Wang, D. Huang, and J.-H. Lai, "Relaxed multi-view clustering in latent embedding space," *Inf. Fusion*, vol. 68, pp. 8–21, 2021.

[37] X. Xie and S. Sun, "Multi-view support vector machines with the consensus and complementarity information," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 12, pp. 2401–2413, Dec. 2020.

[38] M.-S. Chen, L. Huang, C.-D. Wang, D. Huang, and P. S. Yu, "Multiview subspace clustering with grouping effect," *IEEE Trans. Cybern.*, in vol. 52, no. 8, pp. 7655–7668, Aug. 2022.

[39] Z. Hu, F. Nie, R. Wang, and X. Li, "Multi-view spectral clustering via integrating nonnegative embedding and spectral embedding," *Inf. Fusion*, vol. 55, pp. 251–259, 2020.

[40] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang, "Diversity-induced multi-view subspace clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 586–594.

[41] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.

[42] F. Nie, J. Li, and X. Li, "Self-weighted multiview clustering with multiple graphs," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 2564–2570.

[43] M. Yang, Y. Li, P. Hu, J. Bai, J. C. Lv, and X. Peng, "Robust multi-view clustering with incomplete information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1055–1069, Jan. 2023.

[44] Y. Lin, Y. Gou, Z. Liu, B. Li, J. Lv, and X. Peng, "Completer: Incomplete multi-view clustering via contrastive prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11169–11178.

[45] Y. Lin, Y. Gou, X. Liu, J. Bai, J. Lv, and X. Peng, "Dual contrastive prediction for incomplete multi-view representation learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Aug. 08, 2022, doi: 10.1109/TPAMI.2022.3197238.
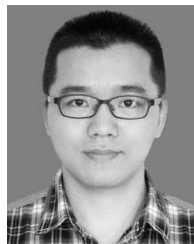
Dong Huang (Member, IEEE) received the BS degree in computer science from the South China University of Technology, Guangzhou, China, in 2009, and the MSc and PhD degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2011 and 2015, respectively. He joined South China Agricultural University, in 2015, where he is currently an associate professor with the College of Mathematics and Informatics. From July 2017 to July 2018, he was a visiting fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include data mining and machine learning. He has published more than 60 papers in refereed journals and conferences, such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Systems, Man, and Cybernetics*, *ACM Transactions on Knowledge Discovery from Data*, SIGKDD, AAAI, and ICDM. He was the recipient of the 2020 ACM Guangzhou Rising Star Award.

Chang-Dong Wang (Member, IEEE) received the BS degree in applied mathematics, and the MSc and PhD degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2008, 2010 and 2013, respectively. He was a visiting student with the University of Illinois at Chicago from January 2012 to November 2012. He is currently an associate professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His current research interests include machine learning and data mining. He has published more than 100 scientific papers in international journals and conferences such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, *ACM Transactions on Knowledge Discovery from Data*, *Pattern Recognition*, KDD, ICDM and SDM. His ICDM 2010 paper won the Honorable Mention for Best Research Paper Award. He was awarded 2015 Chinese Association for Artificial Intelligence (CAAI) Outstanding Dissertation.

Jian-Huang Lai (Senior Member, IEEE) received the MSc degree in applied mathematics and the PhD degree in mathematics from Sun Yat-sen University, China, in 1989 and 1999, respectively. He joined Sun Yat-sen University in 1989 as an assistant professor, where he is currently a professor with the School of Data and Computer Science. His current research interests include the areas of digital image processing, pattern recognition, multimedia communication, wavelet and its applications. He has published more than 200 scientific papers in the international journals and conferences on image processing and pattern recognition, such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *Pattern Recognition*, ICCV, CVPR, IJCAI, ICDM and SDM. He serves as a standing member of the Image and Graphics Association of China, and also serves as a standing director of the Image and Graphics Association of Guangdong.