2018 IEEE International Conference on Data Mining

Partial Multi-View Clustering via Consistent GAN

Qianqian Wang[†], Zhengming Ding[‡], Zhiqiang Tao[§], Quanxue Gao*[†], and Yun Fu[§]
[†]*State Key Laboratory of ISN, Xidian University, Xi'an China.*
[‡]*Department of Computer, Information and Technology, Indiana University-Purdue University Indianapolis, USA.*
[§]*Department of Electrical and Computer Engineering, Northeastern University, USA.*

***Abstract*—Multi-view clustering, as one of the most important methods to analyze multi-view data, has been widely used in many real-world applications. Most existing multi-view clustering methods perform well on the assumption that each sample appears in all views. Nevertheless, in real-world application, each view may well face the problem of the missing data due to noise, or malfunction. In this paper, a new consistent generative adversarial network is proposed for partial multi-view clustering. We learn a common low-dimensional representation, which can both generate the missing view data and capture a better common structure from partial multi-view data for clustering. Different from the most existing methods, we use the common representation encoded by one view to generate the missing data of the corresponding view by generative adversarial networks, then we use the encoder and clustering networks. This is intuitive and meaningful because encoding common representation and generating the missing data in our model will promote mutually. Experimental results on three different multi-view databases illustrate the superiority of the proposed method.**

***Keywords*-partial multi-view, clustering, generative adversarial network**

I. Introduction

Nowadays, with the advance of hardware technology, multi-view data, which are come from different sources for one subject, are common in real-world [1], [2]. For example, one image can be represented either by visual feature or text annotation. In general, different views provide complementary information to describe the data, which lets multi-view learning achieve promising performance, and draws research efforts in many fields such as data analysis, image classification, multimedia and information retrieval [3], [4].

As one of the most representative methods of multi-view learning, multi-view clustering has attracted considerable attention [5], [6]. However, in practice, incomplete view data are ubiquitous due to many unforeseeable reasons [7], such as noise, or malfunction of the data-collecting equipment. traditional multi-view clustering methods cannot directly handle such data, because they learn a shared representation based on an assumption that all the views are complete. For convenience, people always put away all the relevant data, which results in a huge waste of data.

*Corresponding author: Quanxue Gao (e-mail: qxgao@xidian.edu.cn).

Motivated by this problem, two kinds of approaches have been proposed [7], [8]. The first kind of method is based on kernel technique. For example, Shao *et al.* [8] completed the kernel matrices of the incomplete views according to that of the complete views and then doing clustering task. However, this method can only deal with the kernel-based multi-view clustering algorithms. To solve this problem, researchers proposed non-negative matrix factorization (NMF) based methods. For example, Li *et al.* [7] proposed NMF based partial multi-View Clustering (PVC) approach to learn a latent subspace over two views, which achieves better clustering performance. Motivated by these methods, a lot of partial multi-view clustering approaches based on NMF have been proposed [9]–[11]. However, these methods have several limitations restricting its application. (1) It is not straightforward to employ NMF based methods for the large-scale datasets, due to its heave computation caused by a large amount of inverse operations within matrix factorization. (2) NMF methods did not explicitly compensate the missing data in each view, since they exploited some regularizes to constrain on the new representation.

An intuitive way to learn better representation for partial multi-view data is synthesizing the missing data. Recently, generative adversarial networks (GANs) [2], [12], [13], has attracted lots of attention for its generating function. Vanilla GAN [12] creates desired data from random noise, while the latest GANs [14], [15] learn the relationship between two domains. For example, Isola *et al.* [16] used the conditional GANs on paired training data to transfer images from one distribution to another and developed pix2pix GAN. Zhu *et al.* proposed Cycle GAN [14] by using a cycle consistent adversarial network to train unpaired image, which achieves better performance than pix2pix GAN. [15] and [17] use GANs on multi-view data generation. However, current research on GAN mainly focuses on data generation, while few works explore its application in data analysis, like data clustering. There have not been thoroughly studied for partial multi-view clustering based on GANs.

In this paper, we propose a novel deep generative model, termed as consistent GAN, for the partial multi-view clustering task, which consists of two encoders, two GAN networks, and one deep clustering layer. In our model, we adopt two encoders to learn the shared latent representations among multiple views, and naturally leverage the common representation given by one view to infer the missing data of the corresponding view via a GAN network. In details, the generator of GAN tries to recover the missing-view data by using the encoded codes from another view; while the discriminator pushes the fake data towards the real ones. By this means, we fully utilize the adversarial training to explore the complementary information shared by each view. Moreover, to explicitly guide the representation learning for the clustering task, we add one clustering layer to further highlight the cluster structure existing in the common representation. The main contributions of our method are summarized as follows:

1. We propose a novel partial multi-view clustering method called consistent GAN, which not only captures a better clustering structure, but also infers the missing view.

2. Compared with some GAN models which use random noises to generate data, our model fully utilize the complementary information among multi-view data. We use the common representation encoded by one view to generate the missing data. The generated missing view data promote to achieve an optimal clustering result for each view.

3. Extensive experiments have been conducted on several multi-view databases which illustrate the superiority of our method compared with several state-of-the-art methods.

II. Partial Multi-View Clustering via Consistent GAN

A. Motivation

As aforementioned analysis, most existing partial multi-view clustering are based on kernel and non-negative matrix factorization techniques to infer the missing data and learn a common clustering structure. However, these methods are not the best solution for partial multi-view clustering problem: 1) They cannot be applied to large-scale data; 2) They do not consider learning a latent space which is suitable for clustering and simultaneously infers the missing view well. From this point and motivated by GANs, we propose a novel model named partial multi-view clustering via consistent GAN. The proposed model utilizes the combination of GAN and deep embedding clustering layer to learn a shared embedding structure for partial multi-view data, which can capture a better clustering structure and infer the missing view at the same time.

B. Framework

Notations: To introduce easily, we use two-view data as an example. Let two views data $\mathbf{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}\right\}$, $\mathbf{X}^{(v)} \in \mathbf{R}^{N \times d_v} (v = 1, 2)$ be the data matrix of each view, $\mathbf{X}^{(1)} = \{a_1, a_2, \cdots, a_N\}$, $\mathbf{X}^{(2)} = \{b_1, b_2, \cdots, b_N\}$, v be the number of views, $\mathbf{N}$ be the number of samples, and d_v be the feature dimension of v-th view. Considering the partial multi-view setting, we separate the data to two parts: paired data $\{a_i, b_i\}$ which has complete view, unpaired data $\{a'_j, b'_j\}$ which only has view 1 or view 2 data, as shown in Fig. 1. $\tilde{a}_i$ and $\tilde{b}_i$ respectively denote the missing data or generated data of view 1 and view 2.

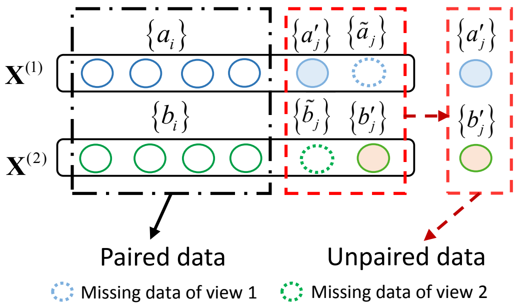

Figure 1: Notation for partial multi-view data.

Network architecture: The model consists of seven sub-networks, two stack fully-connected encoders $\mathbf{E}_1/\mathbf{E}_2$, one deep embedding clustering layer, two stack fully-connected generators $\mathbf{G}_1/\mathbf{G}_2$, two fully-connected discriminators $\mathbf{D}_1/\mathbf{D}_2$. Then we will give detailed introduction for the framework.

1. Encoder $\mathbf{E}_1/\mathbf{E}_2$: $\mathbf{R}^{d_v} \rightarrow \mathbf{R}^m$. $\mathbf{E}_1$, made up of L stacked fully connected layers, aims to learn a latent representation $\mathbf{Z}_1 = \left\{z_1^{(1)}, z_2^{(1)}, \cdots, z_N^{(1)}\right\}$ ($\mathbf{Z}_1 \in \mathbf{R}^{N \times m}$) for all input data $\mathbf{X}^{(1)}$. Specifically, it maps the d_1-dimensional input data a_i to a low-dimensional representation $z_i^{(1)}$, $z_i^{(1)} = f_1(a_i; \theta)$, where f is non-linear function, and θ is the parameter of $\mathbf{E}_1$. $\mathbf{E}_2$ has the same theory as $\mathbf{E}_1$. It learns the latent space $\mathbf{Z}_2 = \left\{z_1^{(2)}, z_2^{(2)}, \cdots z_N^{(2)}\right\}$ ($\mathbf{Z}_2 \in \mathbf{R}^{N \times m}$) from $\mathbf{X}^{(2)}$ and maps the d_2-dimensional input data b_i to a m-dimensional representation $z_i^{(2)}$, $z_i^{(2)} = f_2(b_i; \theta)$. In order to obtain common information from two views, there we partially share parameters of $\mathbf{E}_1$ and $\mathbf{E}_2$, i.e. θ.

2. Generator $\mathbf{G}_1/\mathbf{G}_2$: $\mathbf{R}^m \rightarrow \mathbf{R}^{d_v}$, which recover the two views by latent space $\mathbf{Z}_1/\mathbf{Z}_2$. In our model, $\mathbf{G}_1/\mathbf{G}_2$ have a symmetrical structure with $\mathbf{E}_1/\mathbf{E}_2$, and consist of L stacked fully connected layers, because $\mathbf{G}_1/\mathbf{G}_2$ play roles of generator and decoder at the same time. For $\mathbf{G}_1$, the output $\tilde{a}_i$, is generated by the low-dimensional representation $z_i^{(1)}$, and $z_i^{(2)}$, i.e., $\tilde{a}_i = \mathbf{G}_1(z_i^{(1)})$, and $\tilde{a}_i = \mathbf{G}_1(z_i^{(2)})$. To make $z_i^{(1)}$ be similar to $z_i^{(2)}$, we introduce a common space $\mathbf{Z}$ for these two view data in deep embedding clustering layer. The same for $\mathbf{G}_2$, which generates the second view data by low-dimensional representation $z_i^{(1)}$, and $z_i^{(2)}$, i.e., $\tilde{b}_i = \mathbf{G}_2(z_i^{(1)})$ and $\tilde{b}_i = \mathbf{G}_2(z_i^{(2)})$.

3. Discriminator $\mathbf{D}_1/\mathbf{D}_2$: $\mathbf{R}^{d_v} \rightarrow \{0, 1\}$. Each discriminator consists of 3 stacked fully connected layers, and their function is to distinguish the generated samples $\left\{\tilde{a}_i, \tilde{b}_i\right\}$ and real samples $\{a_i, b_i\}$. For $\mathbf{D}_1$, it should distinguish that $\tilde{a}_i$ is a generated sample and a_i is a real instance. Then it feeds back the result to generator network and updates the parameters of generator. Until the generator can create such realistic sample, the discriminator cannot distinguish which

input is real sample. Similar operation with $\mathbf{D}_2$.

4. Deep embedded clustering layer: this layer is used to change the data distribution and adopt the encoder network and generator network. We computer the current data distribution and target data distribution based on the common space $\mathbf{Z}$ and the cluster centroids $\{\mu_j\}_{j=1}^k$. Then, we use the target data distribution to modify the current data distribution and update the parameter of encoder $\mathbf{E}_1/\mathbf{E}_2$ and generator $\mathbf{G}_1/\mathbf{G}_2$, and refine the cluster centroids.

C. Objective function

The overall objective function of our model contains three terms: Auto-encoder loss, GAN loss, and KL clustering loss.

$$L = \min_{\theta, f_1, f_2, \mathbf{G}_1, \mathbf{G}_2} \max_{\mathbf{D}_1, \mathbf{D}_2} \quad L_{AE} + \lambda_1 L_{cycleGAN} + \lambda_2 L_{KL} \quad , \tag{1}$$

where λ_1 and λ_2 are two parameters for maintaining the impact of GAN and KL clustering loss.

Auto-encoder Loss w.r.t. $\theta, f_1, f_2, \mathbf{G}_1, \mathbf{G}_2$**.** The Auto-encoder loss is minimizing the Euclidean distance of generated sample and the original sample

$$\begin{aligned} L_{AE} \ = \min_{\theta, f_1, f_2, \mathbf{G}_1, \mathbf{G}_2} & \left\|\mathbf{X}^{(1)} - \mathbf{G}_1(f_1(\mathbf{X}^{(1)};\theta))\right\|_F^2 \\ & + \left\|\mathbf{X}^{(2)} - \mathbf{G}_2(f_2(\mathbf{X}^{(2)};\theta))\right\|_F^2 . \end{aligned} \tag{2}$$

This loss is used for the sub-networks of encoder $\mathbf{E}_1/\mathbf{E}_2$ and generator $\mathbf{G}_1/\mathbf{G}_2$. It aims to ensure the encoder can catch the essential structure from two views data, and the latent representation can exactly recover the real data. The encoders take two views $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$ as input and learn two latent representations $\mathbf{Z}_1 = f_1(\mathbf{X}^{(1)};\theta)$, and $\mathbf{Z}_2 = f_2(X^{(2)};\theta)$ for these two views. The generators reconstruct the two views from the latent representation. The output are $\mathbf{G}_1(f_1(\mathbf{X}^{(1)};\theta))$ and $\mathbf{G}_2(f_2(\mathbf{X}^{(2)};\theta))$.

If all the input data are paired data, only the Auto-encoder Loss is enough. The encoder and generator networks can work well. However, our setting is partial multi-view data, the unpaired data will degrade the performance of encoder and generator networks. In order to improve the performance when database contains unpaired data, we add cycle GAN Loss in our objective function to refine the network.

Cycle GAN Loss w.r.t. $\mathbf{G}_1, \mathbf{G}_2, \mathbf{D}_1, \mathbf{D}_2$. In our model, we use cycle GAN, due to we have large amount of unpaired data. A cycle GAN model is composed of two GAN models and trained on unpaired data. It aims to use one distribution data to generate another. Assume that the data distribution of two views are $a \sim P(\mathbf{X}^{(1)})$, $b \sim P(\mathbf{X}^{(2)})$, and let $\mathbf{G}_1 \circ f_2 = \mathbf{G}_1(f_2(b,\theta))$ denote the mapping of the second view sample to the first view data distribution, i.e., using view 2 data b to generate the corresponding view 1 data $\tilde{a} \sim P(\mathbf{X}^{(1)})$. The same for $\mathbf{G}_2 \circ f_1 = \mathbf{G}_2(f_1(a,\theta))$, which denotes the transformation of the first view sample to the second view sample. $\mathbf{D}_1$ is used to discriminate between the generated data $\tilde{a}$ by $\mathbf{G}_1$ and the real data a, while $\mathbf{D}_2$ is used to discriminate between the generated data $\tilde{b}$ by $\mathbf{G}_2$ and the real data b. The cycle GAN loss is:

$$\begin{aligned} & L_{cycleGAN}(\mathbf{G}_1, \mathbf{D}_1, \mathbf{G}_2, \mathbf{D}_2) \\ & \quad = \min_{\mathbf{G}_1, \mathbf{G}_2} \max_{\mathbf{D}_1, \mathbf{D}_2} L_{GAN}(\mathbf{G}_1, \mathbf{D}_1) \\ & \qquad + L_{GAN}(\mathbf{G}_2, \mathbf{D}_2) + \lambda_3 L_{cyc}(\mathbf{G}_1, \mathbf{G}_2). \end{aligned} \tag{3}$$

where the loss of GAN in our model is

$$\begin{aligned} L_{GAN} \ = \min_{\mathbf{G}_1, \mathbf{G}_2} \max_{\mathbf{D}_1, \mathbf{D}_2} & \ \mathbb{E}_{a \sim P(\mathbf{X}^{(1)})} \left[\log \ \mathbf{D}_1(a)\right] \\ & + \mathbb{E}_{b \sim P(\mathbf{X}^{(2)})} \left[\log \ (1 - \mathbf{D}_1(\mathbf{G}_1 \circ f_2(b)))\right] \\ & + \mathbb{E}_{b \sim P(\mathbf{X}^{(2)})} \left[\log \ \mathbf{D}_2(b)\right] \\ & + \mathbb{E}_{a \sim P(\mathbf{X}^{(1)})} \left[\log \ (1 - \mathbf{D}_2(\mathbf{G}_2 \circ f_1(a)))\right] . \end{aligned} \tag{4}$$

The cycle consistency loss is

$$\begin{aligned} L_{cyc}(\mathbf{G}_1, \mathbf{G}_2) = & \mathbb{E}_{a \sim P(\mathbf{X}^{(1)})} \|\mathbf{G}_2(\mathbf{G}_1 \circ f_2(b)) - b\|_1 \\ & + \mathbb{E}_{b \sim P(\mathbf{X}^{(2)})} \|\mathbf{G}_1(\mathbf{G}_2 \circ f_1(a)) - a\|_1 . \end{aligned} \tag{5}$$

The generator is trained to generate fake data which are similar to real data. The discriminators are trained to distinguish the fake data from the real data. They play a min-max game until convergence. However, the GANs are trained to map a same input to any random permutation of sample which are in the target data distribution. Hence, the GAN loss alone cannot ensure a desired output. To reduce the space of possible mapping functions, cycle GAN introduces cycle-consistent loss to update the learned mapping, i.e., for each image, after passing the image translation cycle, should be brought back to the itself. The cycle consistency loss can assist the generator in mapping a given sample a to a desired output b. Thus, the combination of GAN loss and cycle consistency loss ensure the generator map the input to a desired output.

KL Clustering Loss w.r.t. $\theta, f_1, f_2, \mathbf{G}_1, \mathbf{G}_2, \mathbf{D}_1, \mathbf{D}_2$. According the aforementioned analysis, cycle GAN loss will update the generator and discriminator networks. However, it does not modify the encoders which learn the common representation $\mathbf{Z}$ for the final clustering task, while unpaired data have a bad effect on the common representation learned by paired data. In order to obtain an optimal clustering structure, we add a clustering loss which is measured by Kullback-Leibler divergence (KL-divergence). We will learn two latent subspaces for the two views $\mathbf{Z}_1 = f_1(\mathbf{X}^{(1)};\theta)$, and $\mathbf{Z}_2 = f_2(\mathbf{X}^{(2)};\theta)$. Then we get a common latent representation based on these two subspaces

$$\mathbf{Z} = h(\mathbf{Z}_1, \mathbf{Z}_2), \tag{6}$$

$h(\cdot)$ represents a function of concatation or summation. Define $\{\mu_j\}_{j=1}^k$ as k initial clustering centroids. According to [18], we use the Student's t-distribution as a kernel to measure the similarity between common latent representation point z_i and centroid μ_j. The probability of assigning sample i to cluster j can be calculated by

$$q_{ij} = \frac{\left(1 + \left\|z_i - \mu_j\right\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'} \left(1 + \left\|z_i - \mu_{j'}\right\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}. \tag{7}$$

It is also named soft assignment. α is the degree of freedom of the Student's t-distribution. In order to improve clustering performance, and lay special stress on data points assigned with high confidence, we computer target distribution p_i by first raising q_i to the squared and then normalizing by frequency per cluster

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \tag{8}$$

where $f_j = \sum_i q_{ij}$ is soft cluster frequency. The clustering loss is defined as minimizing the KL-divergence between a data distribution and a target distribution.

$$L_{KL} = \mathrm{KL(P|Q)} = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{9}$$

Our aim is to match the soft assignment q_i to the target distribution p_i. In this way, we can sharpen the data distribution and concentrate the same class data. In addition, we will get a more effective and common representation for partial multi-view clustering.

D. Implementation

Step 1: Training encoder $\mathbf{E}_1, \mathbf{E}_2$ and generator $\mathbf{G}_1, \mathbf{G}_2$ on paired data. We use paired data to train encoder and generator. Since generator can be seen as the decoder corresponding to encoder, we just use the AE loss to train these networks and update the parameter of $\theta, f_1, f_2, \mathbf{G}_1, \mathbf{G}_2$. Specifically, we take $\{a_i, b_i\}$ as input for encoder $\mathbf{E}_1, \mathbf{E}_2$ and get two latent spaces $\mathbf{Z}_1, \mathbf{Z}_2$ and common representation $\mathbf{Z}$. Then we $\mathbf{Z}_1, \mathbf{Z}_2$ as the input of generator $\mathbf{G}_1, \mathbf{G}_2$ and get four outputs. $\mathbf{Z}_1$ can generate $\left\{\tilde{a}_i, \tilde{b}_i\right\}$, and $\mathbf{Z}_2$ can also generate $\left\{\tilde{a}_i, \tilde{b}_i\right\}$. Then we compute the AE loss and updating encoder and generator network until convergence. After step 1, we save the clustering centroids $\{\mu_j\}_{j=1}^k$ for the following training. These clustering centroids learned by paired data can instruct sample which has missing view to be assigned to the right group.

Step 2: Training generator $\mathbf{G}_1, \mathbf{G}_2$ and discriminator $\mathbf{D}_1/\mathbf{D}_2$ on all data. In this step, we train the generator and discriminator network on all data. For paired data $\{a_i, b_i\}$, we directly take them as the input of generators $\mathbf{G}_1, \mathbf{G}_2$. For unpaired data $\left\{a'_j\right\}$, in order to increase the number of unpaired data, we randomly choose one sample from all the second view $\left\{b_i, b'_j\right\}$ as the input of generator. The same operation for unpaired data $\left\{b'_j\right\}$. After step 2, we save the output of generator $\tilde{a}_j, \tilde{b}_j$ when inputting a'_j, b'_j, i.e., the inferred missing data. Then we use the complete database to compute the common representation $\mathbf{Z}$.

Step 3: Training encoder $\mathbf{E}_1, \mathbf{E}_2$, generator $\mathbf{G}_1, \mathbf{G}_2$, discriminator $\mathbf{D}_1/\mathbf{D}_2$, and clustering layer on inferred data by step 2. Finally, we use the clustering centroids $\{\mu_j\}_{j=1}^k$ from step 1, the common representation $\mathbf{Z}$, and the completed data $\left\{\{a_i, b_i\}, \left\{a'_j, \tilde{b}_j\right\}, \{\tilde{a}_j, b'_j\}\right\}$ from step 2 as input for encoders and clustering layer to train all model. In each iteration, we update the the clustering centroids, the common representation and the inferred missing data again.

III. Experimental Analysis

In this section, we evaluate our method over three kinds of databases image feature with image feature (HW), image feature with text feature (BDGP), image pixels with image pixels (MNIST). To prove the effectiveness of our method, we compare it with some state-of-the-art partial multi-view clustering methods (Incomplete Multi-Modal Visual Data Grouping (IMG) [11], Partial Multi-View Clustering using Graph Regularized NMF (GPVC) [7], [10]). Additionally, some multi-view clustering approaches: Feature Concatenation Spectral Clustering (ConSC) [19], Robust Multi-view Spectral Clustering (RMSC) [20], Auto-weighted Multiple Graph Learning (AMGL) [21], and spectral clustering for single view are also conducted in our experiments for comparison.

A. Experimental Setting

1) Dataset: In the experiment, we validate the clustering performance of our method on three different kinds of multi-view data, each of which is briefly introduced as follows.

Image feature with image feature: Handwritten numerals (HW) [22] database consists of 2,000 images for 10 classes from 0 to 9 digit. Each class contains 200 samples. Each sample has 2 kinds of features: 76 Fourier coefficients and 216 profile correlations.

Image feature with text feature: BDGP [23] is a two-view database. One is visual view and the other is textual view. It contains 2,500 images about drosophila embryos belonging to 5 categories. Each image is represented by a 1,750-D visual vector and a 79-D textual feature vector. In our experiment, we use all data on BDGP database, and evaluate the performance on both visual feature and textual feature.

Image pixels with image pixels: MNIST [24] is a handwritten digits image database, and the image size is 28×28 pixels. MNIST consists of 60,000 training examples and 10,000 testing examples. In our experiments, we use two views, the first view is the original black and white image of MNIST, and the second view is the corresponding edge image of the first view [25]. On account of the comparison methods cannot be conducted on large scale database, we randomly sample 4000 images to consist a new sampled MNIST database to do the partial multi-view clustering experiment.

Table I: The average Clustering Accuracy vs. different impartial ratio on the HW Database, BDGP Database, and the sampled MNIST Database.

Methods	**HW**					**BDGP**					**sampled MNIST**				
	0.1	**0.3**	**0.5**	**0.7**	**0.9**	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
SC1	44.58	50.97	57.42	64.32	72.12	32.96	35.39	38.45	41.03	44.04	34.83	39.56	44.29	47.74	52.77
SC2	29.81	32.79	35.27	37.66	41.73	47.48	51.69	56.92	61.39	67.16	26.45	29.44	32.07	35.04	38.87
AMGL	34.26	42.21	50.24	60.37	67.83	25.24	23.57	25.38	28.07	29.58	15.58	14.12	15.24	24.15	33.46
RMSC	40.16	46.25	55.64	63.30	69.78	33.95	36.83	39.07	42.33	44.99	34.92	41.50	45.75	49.60	51.44
ConSC	44.57	48.25	54.53	64.54	77.97	27.81	22.30	21.39	21.06	28.84	37.04	35.81	36.74	41.37	50.88
GPVC	32.38	30.77	34.19	42.36	57.30	50.15	54.24	62.77	68.33	75.46	35.25	38.64	42.38	44.01	46.44
IMG	53.50	54.55	54.57	55.29	56.33	43.73	45.08	48.68	50.55	51.76	**46.55**	46.40	46.13	45.92	46.22
Ours	**69.82**	**83.80**	**88.06**	**90.30**	**92.34**	**52.10**	**67.11**	**86.31**	**91.54**	**94.98**	45.17	**48.36**	**52.80**	**52.02**	**53.40**

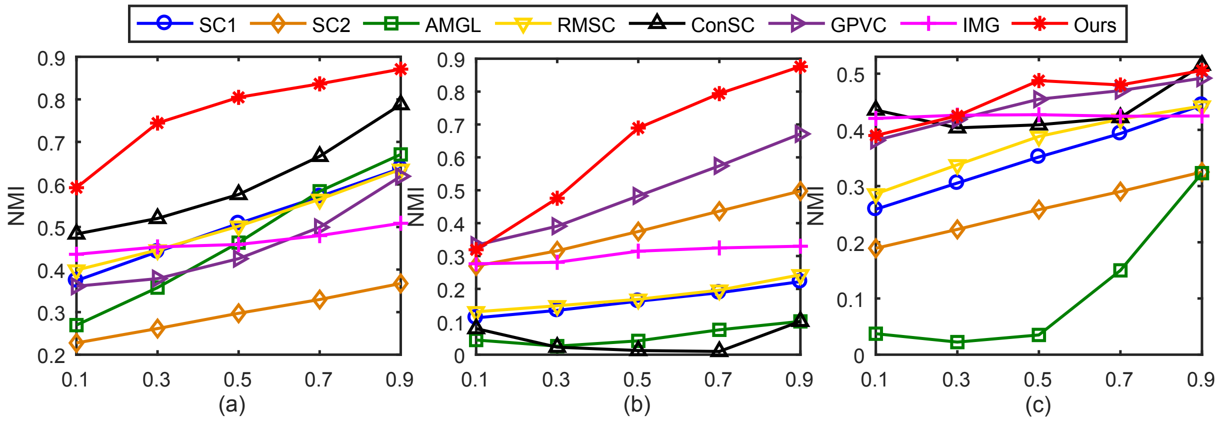

Figure 2: The Average Clustering NMI of all the methods vs. different impartial ratio on the three Database: (a) HW database, (b) BDGP database, (c) MNIST database.

2) *Baseline methods and evaluate metrics:* To simulate the partial view setting, we test all the methods under different impartial ratio (1-partial/incomplete example ratio). Impartial ratio varying from 0.1 to 0.9 with an interval of 0.2. Considering that some multi-view clustering cannot deal with missing instances, we use the average feature vector to fill in the missing instances at first. This process is repeated 10 times in our experiments. We evaluate the clustering performance with three standard clustering evaluation metrics, i.e. Accuracy (ACC) [26], Normalized Mutual Information (NMI) [27]. TABLE I lists the average clustering accuracy of all methods on the HW database, BDGP database, and sampled MNIST database. Fig. 2 shows the average clustering NMI vs. different impartial ratios on the three databases.

3) *Implementation details:* We implement our algorithm with PyTorch and run all the experiments on the platform of Ubuntu Linux 16.04 with NVIDIA Titan V Graphics Processing Units (GPUs) and 32 GB memory size. We use Adam [28] optimizer with default parameter setting to train our model and fix the learning rate as 0.0001. We conduct 20 epoches for each training step. All the other methods are tested on the same environment by Matlab.

B. Partial Multi-view Clustering Performance

As shown by TABLE I, and Fig. 2, our approach generally achieves the best clustering performance on all the cases. Here, we summarize some observations as follows.

1. From TABLE I, and Fig. 2, we observe that partial multi-view methods achieve superior results in most cases, especially when the partial ratio is large. It illustrates that the existence of missing view data will degrade the performance of multi-view clustering methods. We can see that partial multi-view methods are more effective when encountering with partial multi-view data problem. TABLE I also shows the multi-view clustering method AMGL becomes worse than single-view methods when there exist incomplete data. It further indicates that some multi-view methods are easily influenced by data missing and noise data.

2. The results of TABLE I also illustrate that our method outperforms other state-of-the-art methods, It is probably due to the fact that our method learns a consistent clustering structure for each view and uses it to infer the missing data. Then the inferred missing data puts forward to a more effective common subspace, which is crucial for the final clustering task.

C. Ablation study

In order to show the effect of each loss in our objective function, we run ablation studies to isolate the effect of the Auto-encoder, GAN, and clustering layer. Due to our training procedure is step by step, we simply compute the clustering

Table II: The ablation study of our approach under different impartial ratios on the HW dataset, where we show the performance of our method with different loss function by clustering accuracy.

Loss	0.1	0.3	0.5	0.7	0.9
AE	0.5800	0.7192	0.7734	0.7944	0.8216
AE+GAN	0.6034	0.7554	0.7817	0.8590	0.8777
AE+GAN+Clustering	**0.6590**	**0.8232**	**0.8651**	**0.8855**	**0.9070**

performance after step 1, step 2 and step 3 of Algorithm 1 to evaluate the performance of each component. Step 1 only uses auto-encoder network under AE loss. Step 2 is based on the result of step 1, so the result of step 2 is obtained by auto-encoder and GAN network under AE loss and GAN loss; Step 3 uses all models under total objective function. TABLE II respectively shows the clustering accuracy for these three kinds of loss. We can see the clustering accuracy will increase with adding loss step by step. It illustrates each loss in our objective function is significant for the final performance of our method. GAN loss improves the result of AE loss, it prove the contribution that generated missing view data promote to achieve a better clustering result. Clustering loss boosts the result of the first two component. This illustrates that a well common representation help infer more realistic missing view data and they promote mutually.

IV. Conclusion

In this paper, we propose a novel consistent GAN model for partial multi-view clustering task, which simultaneously learns an excellent clustering structure and infers the incomplete views on the common subspace structure via GAN model. In addition, the imputation of incomplete view data can further study a consistent common structure which can improve the clustering performance. Comprehensive experiments validate the clustering performance improvement of the proposed method compared with state-of-the-art methods.

Acknowledgment

This work is supported by National Natural Science Foundation of China under Grant 61773302, and 61728103.

References

[1] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[2] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.

[3] J. Salvador and J. R. Casas, "Multi-view video representation based on fast monte carlo surface reconstruction," *IEEE TIP*, vol. 22, no. 9, pp. 3342–3352, 2013.

[4] Z. Ding, M. Shao, and Y. Fu, "Robust multi-view representation: A unified perspective from multi-view learning to domain adaption." in *IJCAI*, 2018, pp. 5434–5440.

[5] F. Nie, J. Li, and X. Li, "Self-weighted multiview clustering with multiple graphs," in *IJCAI*, 2017, pp. 2564–2570.

[6] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, "From ensemble clustering to multi-view clustering," in *IJCAI*, 2017, pp. 2843–2849.

[7] S.-Y. Zhi and H. Zhou, "Partial multi-view clustering," in *AAAI*, 2014.

[8] W. Shao, X. Shi, and S. Y. Philip, "Clustering on multiple incomplete datasets via collective kernel learning," in *ICDM*, 2013, pp. 1181–1186.

[9] H. Zhao, H. Liu, and Y. Fu, "Incomplete multi-modal visual data grouping." in *IJCAI*, 2016, pp. 2392–2398.

[10] N. Rai, S. Negi, S. Chaudhury, and O. Deshmukh, "Partial multi-view clustering using graph regularized nmf," in *ICPR*, 2016, pp. 2192–2197.

[11] B. Qian, X. Shen, Y. Gu, Z. Tang, and Y. Ding, "Double constrained nmf for partial multi-view clustering," in *DICTA*, 2016, pp. 1–7.

[12] L. Tran, X. Liu, J. Zhou, and R. Jin, "Missing modalities imputation via cascaded residual autoencoder," in *IEEE CVPR*, 2017, pp. 1405–1414.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014, pp. 2672–2680.

[14] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[15] C. Shang, A. Palmer, J. Sun, K.-S. Chen, J. Lu, and J. Bi, "Vigan: Missing view imputation with generative adversarial networks," *arXiv preprint arXiv:1708.06724*, 2017.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[17] B. Zhao, X. Wu, Z.-Q. Cheng, H. Liu, Z. Jie, and J. Feng, "Multi-view image generation from a single-view," *arXiv preprint arXiv:1704.04886*, 2017.

[18] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016, pp. 478–487.

[19] A. Kumar, P. Rai, and H. Daume, "Co-regularized multi-view spectral clustering," in *NIPS*, 2011, pp. 1413–1421.

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

[21] F. Nie, J. Li, X. Li *et al.*, "Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification." in *IJCAI*, 2016, pp. 1881–1887.

[22] M. Van Breukelen, R. P. Duin, D. M. Tax, and J. Den Hartog, "Handwritten digit recognition by combined classifiers," *Kybernetika*, vol. 34, no. 4, pp. 381–386, 1998.

[23] X. Cai, H. Wang, H. Huang, and C. Ding, "Joint stage recognition and anatomical annotation of drosophila gene expression patterns," *Bioinformatics*, vol. 28, no. 12, pp. i16–i24, 2012.

[24] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[25] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NIPS*, 2016, pp. 469–477.

[26] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *IEEE TKDE*, vol. 17, no. 12, pp. 1624–1637, 2005.

[27] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE TNNLS*, vol. 20, no. 2, pp. 189–201, 2009.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.