Deep Partial Multiplex Network Embedding

Qifan Wang*
Facebook AI
Menlo Park, CA, USA
wqfcr@fb.com

Yi Fang
Santa Clara University
Santa Clara, CA, USA
yfang@scu.edu

Anirudh Ravula
Google Research
Mountain View, CA, USA
ravulaanirudh25@gmail.com

Ruining He
Google Research
Mountain View, CA, USA
ruininghe@google.com

Bin Shen
Google Research
Mountain View, CA, USA
stanshenbin@gmail.com

Jingang Wang
Meituan NLP Center
Beijing, China
wangjingang02@meituan.com

Xiaojun Quan
Sun Yat-sen University
Guangzhou, China
quanxj3@mail.sysu.edu.cn

Dongfang Liu*
Rochester Institute of Technology
Rochester, NY, USA
dongfang.liu@rit.edu

ABSTRACT

Network embedding is an effective technique to learn the low-dimensional representations of nodes in networks. Real-world networks are usually with multiplex or having multi-view representations from different relations. Recently, there has been increasing interest in network embedding on multiplex data. However, most existing multiplex approaches assume that the data is complete in all views. But in real applications, it is often the case that each view suffers from the missing of some data and therefore results in partial multiplex data.

In this paper, we present a novel Deep Partial Multiplex Network Embedding approach to deal with incomplete data. In particular, the network embeddings are learned by simultaneously minimizing the deep reconstruction loss with the autoencoder neural network, enforcing the data consistency across views via common latent subspace learning, and preserving the data topological structure within the same network through graph Laplacian. We further prove the orthogonal invariant property of the learned embeddings and connect our approach with the binary embedding techniques. Experiments on four multiplex benchmarks demonstrate the superior performance of the proposed approach over several state-of-the-art methods on node classification, link prediction and clustering tasks.

CCS CONCEPTS

• **Information systems → Web searching and information discovery**.

*Corresponding Authors. This work was mainly done before joining Meta.

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524717

KEYWORDS

network embedding, multiplex learning, social network representation, graph representation

ACM Reference Format:
Qifan Wang, Yi Fang, Anirudh Ravula, Ruining He, Bin Shen, Jingang Wang, Xiaojun Quan, and Dongfang Liu. 2022. Deep Partial Multiplex Network Embedding. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3487553.3524717

1 INTRODUCTION

Network embedding is designed for learning low-dimensional and typically non-linear representations of nodes in the network, which is able to preserve network information. Network embedding has been shown to be useful in many downstream tasks, such as node classification [55], node clustering [16], link prediction [31] and community detection [59]. A variety of network embedding techniques have been proposed in the literature [4, 12, 15, 26, 33–36, 42, 46, 50, 52, 58, 68, 72]. However, most of these methods focus on single networks, where nodes in the networks are only associated with one type of features or relations.

In many real-world applications, data usually have multiplex representations [70], where nodes are associated with multiple features from different sources. Multiple types of edges/relations are then generated from these disparate features. For example, in document corpus, a document has hyperlink feature that connects to other related documents. It can also have semantic representation such as attribute or tag feature. Documents are linked together in the attribute view if they share at least one attribute. In Flickr [5], users can be represented with their friendship to others, public comments, photos, reviews, tags, etc. Similarly, users are linked in the photo network or tag network if they share same photos or tags. Previous research on multiplex representation learning [32, 56] has demonstrated improved performance by leveraging complementary information from different views. Therefore, there is a growing interest in multiplex network embedding that effectively integrates information from disparate views [14, 71, 73].

Although existing multiplex network embedding methods generate promising results in dealing with multiplex data, most of them assume that all nodes in the network have full information in all views. However, in real-world tasks, it is often the case that a view suffers from some missing information, which results in partial data [29, 51, 54]. For instance, in document corpus, many documents may not contain any hyperlink or tag information. In Flickr, a user might have few or even no friend connections, reviews or tags, resulting in an isolated node in the corresponding relationship network. Moreover, it is also common that users don't share some of their information, such as photos and comments, for privacy consideration. Therefore, it is a practical and important research problem to design effective network embedding methods on partial multiplex data.

There are several ways to apply existing multiplex network embedding methods to partial data. One can either remove the data that suffer from missing information, or preprocess the partial data by first filling in the missing data. The first strategy is clearly not suitable since the purpose is to map all nodes to their corresponding embedding vectors, and our experiments show that the second strategy does not achieve good performance either. In this paper, we propose a novel Deep Partial Multiplex Network Embedding (DPMNE) approach to deal with such incomplete data. More specifically, a deep autoencoder network is introduced to learn the deep representations of node features. A unified learning framework is developed to learn the network embedding, which simultaneously minimizes the reconstruction error from the deep autoencoder, enforces the data consistency across different views via common latent subspace learning, and preserves data proximity within the same network through graph Laplacian. A coordinate descent algorithm is applied as the optimization procedure. We then further connect our approach to binary embedding methods [48] based on the orthogonal invariant property of our formulation. Experiments on four multiplex datasets demonstrate the advantages of the proposed approach over several state-of-the-art single and multiplex network embedding methods.

2 RELATED WORK

2.1 Single Network Embedding

Single network embedding methods [8, 11, 13, 18, 60, 74] learn an information preserving embedding of a single-view network for node classification, node clustering and many other related tasks. A spectral based method [3] has been proposed, which uses the top-k eigenvectors to represent the network nodes. DeepWalk [36] introduces the idea of Skip-gram to learn node representations from random-walk sequences. LINE [44] tries to use the embedding to approximate the first-order and second-order proximities of the network. On top of DeepWalk, Node2Vec [21] adds two parameters to control the random walk process and make it biased random walk. SDNE [47] uses deep neural networks to preserve the neighbors structure proximity in network embedding. Recently, graph neural network (GNN) based methods [1, 23, 66] have been proposed, which generate embedding by sampling and aggregating features from a node local neighborhood in the network. GraphSAGE [23] is a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. DEGNN [28] presents a distance encoding GNN that learns a generator to approximate the node connectivity distribution and a discriminator to differentiate fake nodes and the nodes sampled from the true data distribution. For a comprehensive review on GNN models, please refer to [61].

methods	Deep	Partial	Multiplex	Network
DeepWalk [36]	x	x	x	✓
Node2Vec [21]	x	x	x	✓
LINE [44]	x	x	x	✓
GraphSAGE [23]	✓	x	x	✓
SDNE [47]	✓	x	x	✓
DANE [16]	✓	x	✓	✓
MVE [38]	x	x	✓	✓
MNE [70]	x	x	✓	✓
CFANE [35]	✓	x	✓	✓
MAGCN [10]	✓	x	✓	✓
HWNN [42]	✓	x	✓	✓
PVC [29]	x	✓	✓	x
IMVC [32]	x	✓	✓	x
DPMNE	✓	✓	✓	✓

Table 1: Summary of existing network embedding and partial data representation approaches. Deep: learning deep representations. Partial: handling partial data. Multiplex: incorporating multiplex data. Network: modeling network information.

2.2 Multiplex Network Embedding

Various approaches have been proposed to learn network embedding on multiplex data [6, 7, 9, 30, 37, 53, 62, 65, 67]. Such methods effectively integrate information from individual views while exploiting complementary information supplied by different views. MVE [38] is one of the pioneer work in this category. This work combines the information from multiple sources using a weighted voting scheme. MVN2Vec [41] further studies how varied extent of preservation and collaboration can impact the multiplex embedding learning. DANE [16], DONE [2] and ProGAN [17] treat the attribute/tag information associated with the nodes as additional feature and learn embedding with deep neural networks. More recently, MNE [70] uses a latent space to integrate the information across multiple views. MAGCN [10] proposes a multi-view attribute graph convolution networks model for the clustering task. A quick survey on multiplex network representation is provided in [67]. These multiplex network embedding methods [24, 57, 64] achieve promising results. However, the partial data scenarios are not considered.

2.3 Partial Data Learning

It is also worth mentioning that there are few single network embedding approaches that deal with incomplete data [59, 69]. Obviously, these methods are not suitable in multiplex network setting. There are also some multi-view approaches [22, 29, 32, 39, 63] proposed to deal with incomplete data in clustering task. For example, PVC [29] develops a two-view clustering method that learns a unique representation between views to handle incomplete data in each view. Most recently, IMVC [32] uses multiple kernel k-means with incomplete kernels to jointly conduct

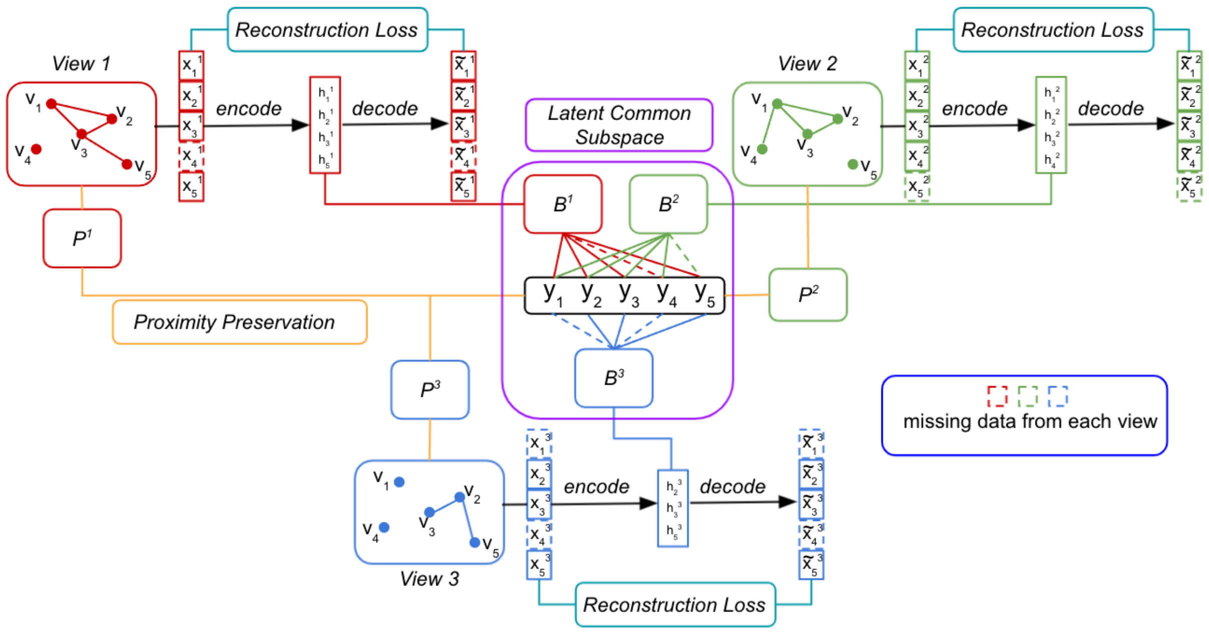

Figure 1: The architecture of our proposed DPMNE. There are three views in the multiplex network example, with missing data from each individual views, i.e., x_4 is missing from view 1, x_5 is missing from view 2, x_1 and x_4 are missing from view 3. The three key components of DPMNE are: 1) Deep reconstruction loss from all views. 2) Data consistency among views via latent common subspace learning. 3) Proximity preservation from each view.

clustering and kernel matrix imputation. Although these methods obtain reasonable performance, they are not directly applicable to network embedding as network information can not be modeled by these methods. Moreover, none of these methods learn the deep representations for the multiplex network. We compare and summarize these approaches in Table 1.

3 DEEP PARTIAL MULTIPLEX NETWORK EMBEDDING

3.1 Problem Definition

Given a multiplex network $\boldsymbol{G}$=($\boldsymbol{V}, \boldsymbol{E}$), where $\boldsymbol{V}$=$\{v_i \mid i = 1, \ldots, n\}$ denotes the set of nodes. $\boldsymbol{X}$=$\{\boldsymbol{X}^1,\boldsymbol{X}^2, \ldots ,\boldsymbol{X}^t\}$ are the multiplex features associated with the nodes, where t is the total number of views, $\boldsymbol{X}^s$=$\{x_i^s \mid i = 1, 2, \ldots , n\}$ with $x_i^s \in \mathbb{R}^{d_s}$ is the feature of the i-th node in the s-th view. $\boldsymbol{E}$=$\{\boldsymbol{E}^s\}$ denotes the edge sets in the multiplex network, where $\boldsymbol{E}^s_{i,j}$=1 indicates that v_i and v_j are linked in the s-th view. In the partial data setting, a partial data $\bar{\boldsymbol{X}}$=$\{\bar{\boldsymbol{X}}^1,\bar{\boldsymbol{X}}^2, \ldots ,\bar{\boldsymbol{X}}^t\}$ instead of $\boldsymbol{X}$ is given, where some of the node features are missing from each individual views, e.g., x_4^1 is missing from View 1 and x_5^2 is missing from View 2 in Figure 1. The purpose of DPMNE is to learn a low-dimensional embedding representation $\boldsymbol{Y}$=$\{y_1, y_2, \ldots , y_n\} \in \mathbb{R}^{n\times d}$ of $\boldsymbol{G}$, where $d \ll d_s$ is the latent embedding dimension.

The overall model architecture is shown in Figure 1. The objective function of DPMNE is composed of three components: (1) Deep reconstruction loss within each views, where we learn a deep representation of the node feature using autoencoder model. (2) Data consistency among views, where latent common subspace learning is utilized to ensure that the node embeddings generated from different views are consistent. (3) Proximity preservation within view, where graph Laplacian is applied to enforce that linked nodes within each network should have close embeddings.

3.2 Deep Representation

To capture the sparsity and highly non-linear structure in the feature space, we adopt a deep autoencoder to map the input data to the representation space. Autoencoder is a powerful unsupervised deep model for feature learning. It has been widely used for various machine learning applications [27, 47]. The basic autoencoder contains three layers, they are the input layer, the hidden layer, and the output layer, which are defined as follows:

$$h = \sigma(\boldsymbol{W}^{(en)}x + b^{(en)}), \quad \tilde{x} = \sigma(\boldsymbol{W}^{(de)}h + b^{(de)}) \tag{1}$$

where h is the deep representation from the encoder, $\tilde{x}$ is the reconstructed feature from the decoder. $\boldsymbol{H} = \{\boldsymbol{W}^{(en)}, b^{(en)}, \boldsymbol{W}^{(de)}, b^{(de)}\}$ are autoencoder parameters. $\sigma(.)$ denotes the non-linear activation function. In our implementation, we employ K layers in the encoder:

$$\begin{aligned} h_1 &= \sigma(\boldsymbol{W}_1^{(en)}x + b_1^{(en)}) \\ h_k &= \sigma(\boldsymbol{W}_k^{(en)}h_{k-1} + b_k^{(en)}) \end{aligned} \tag{2}$$

Similarly, there will be K layers in the decoder. The goal of the autoencoder is to minimize the reconstruction loss of the reconstructed features $\tilde{x}$ and the input features x, in each individual views (note that there is one autoencoder per view):

$$\min_{\boldsymbol{H}} \sum_{s=1}^{t} \|\boldsymbol{X}^s - \tilde{\boldsymbol{X}}^s\|_F^2 \tag{3}$$

Although minimizing the reconstruction loss does not explicitly preserve the similarity between samples, as shown in [40], the reconstruction criterion can effectively capture the data manifolds and thus preserve the similarity between data examples.

3.3 Data Consistency among Views

In the partial multiplex network setting, the nodes are represented by heterogeneous features of different dimensions, which makes it difficult for learning their embeddings. By investigating the problem from view perspective, in each individual view, the nodes are sharing the same feature space, and different views are coupled/bridged by the shared common nodes. If we can learn a common latent subspace across views, where embeddings belonging to the same node among different views are consistent, while at the same time for each view, the representations for linked nodes are close in the latent subspace. Then the embeddings can be directly learned from this subspace, and we do not need to fill in or complete the partial data. Following the above idea, the deep latent subspace learning can be formulated as:

$$\min_{\boldsymbol{Y},\boldsymbol{B},\boldsymbol{H}} \sum_{s=1}^{t} \|\boldsymbol{I}^s(\boldsymbol{H}^s - \boldsymbol{Y}\boldsymbol{B}^s)\|_F^2 + \lambda\, R(\boldsymbol{Y},\boldsymbol{B},\boldsymbol{H}) \tag{4}$$

where $\boldsymbol{I}^s \in \mathbb{R}^{n\times n}$ is a diagonal matrix with element $\boldsymbol{I}^s_{ii} = 0$ or 1 indicating whether x^s_i is missing or not. $\boldsymbol{H}^s \in \mathbb{R}^{n\times d^h_s}$ are the deep representations of the data in $s\text{-}th$ view as described in Section 3.2. Note that the $i\text{-}th$ row of $\boldsymbol{H}^s$ will be 0 if x^s_i is missing. $\boldsymbol{B}^s \in \mathbb{R}^{d\times d^h_s}$ is the basis matrix for $s\text{-}th$ view's latent space. The same latent space dimension d is shared across views. $\boldsymbol{Y} \in \mathbb{R}^{n\times d}$ is the common representation/embedding of nodes in the latent space. $R(\cdot) = \|\cdot\|_F^2$ is the regularization term and λ is the trade-off parameter.

In the above formulation, the individual basis matrix $\boldsymbol{B}^s$, which are learned from all available instances from all views, are connected by the common latent representation $\boldsymbol{Y}$. Moreover, no interpolation is needed for the missing data beforehand. In fact, the deep representation of the missing data can even be reconstructed with the learned basis and the common latent embedding, i.e., $h^s_i = y_i B^s$. By solving the above problem, the deep representation $\boldsymbol{H}$, the latent space basis $\boldsymbol{B}$ and the homogeneous feature representation $\boldsymbol{Y}$ are simultaneously learned to minimize the latent representation error.

3.4 Proximity Preservation within Views

One of the key problems in network embedding algorithms is proximity preserving, which indicates that linked nodes should be mapped to similar embedding within a close distance. Therefore, besides the data consistency across different views, we also preserve the data proximity within each individual network. In other words, we want the learned embedding $\boldsymbol{Y}$ to preserve the proximity structure in each network. In this work, we use the L_2 distance to measure the proximity between y_i and y_j as $\|y_i - y_j\|^2$ as in most network embedding work. Then one natural way to preserve the proximity in each network is to minimize the weighted average distance as follows:

$$\sum_{s=1}^{t}\sum_{i,j} \boldsymbol{P}^s_{ij}\|y_i - y_j\|^2 = \sum_{i,j} \boldsymbol{P}_{ij}\|y_i - y_j\|^2 \tag{5}$$

here $\boldsymbol{P}_{ij} = \sum_{s=1}^{t} \boldsymbol{P}^s_{ij}$ and $\boldsymbol{P}^s$ is the proximity matrix in $s\text{-}th$ view, which can be obtained from the edges in $s\text{-}th$ network $\boldsymbol{E}^s$. A simple way to define $\boldsymbol{P}^s$ is to directly use the first-order proximity, i.e., $\boldsymbol{P}^s = \boldsymbol{E}^s$. However, the first-order proximity is usually very sparse and insufficient to fully model the relationships between nodes in most cases, especially under the partial data setting. In order to characterize the connections between nodes better, we adopt high-order proximity [16, 31] and define $\boldsymbol{P}^s$ as:

$$\boldsymbol{P}^s = w_1\boldsymbol{E}^s + w_2(\boldsymbol{E}^s)^2 + \cdots + w_l(\boldsymbol{E}^s)^l \tag{6}$$

where l is the order, and $w_1,\ldots,w_l$ are the weights for each term[1]. Matrix $(\boldsymbol{E}^s)^l$ denotes the l-order proximity matrix. To meet the proximity preservation criterion, we seek to minimize the quantity in Eqn.5 in the network since it incurs a heavy penalty if two connected nodes have very different embedding representations. By introducing a diagonal matrix $\boldsymbol{D}$, whose entries are given by $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{P}_{ij}$. Eqn.5 can be rewritten as:

$$tr\left(\boldsymbol{Y}^T(\boldsymbol{D}-\boldsymbol{P})\boldsymbol{Y}\right) = tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) \tag{7}$$

where $\boldsymbol{L}$ is called graph *Laplacian* [3] and $tr(\cdot)$ is the matrix trace function. By minimizing the above objective in all networks, the proximity between different nodes can be preserved in the learned embedding.

3.5 Overall Objective and Optimization

The entire objective function consists of three components: the deep reconstruction loss in Eqn.3, the data consistency among views in Eqn.4 and proximity preservation within views given in Eqn.7 as follows:

$$\begin{aligned}\min_{\boldsymbol{Y},\boldsymbol{B},\boldsymbol{H}} O = \sum_{s=1}^{t} \|\boldsymbol{X}^s - \tilde{\boldsymbol{X}}^s\|_F^2 + \alpha \sum_{s=1}^{t} \|\boldsymbol{I}^s(\boldsymbol{H}^s - \boldsymbol{Y}\boldsymbol{B}^s)\|_F^2 \\ +\beta\, tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) + \lambda\, R(\boldsymbol{Y},\boldsymbol{B},\boldsymbol{H})\end{aligned} \tag{8}$$

where α, β and λ are trade-off parameters to balance the weights among the terms. Directly minimizing the objective function in Eqn.8 is intractable since it is a non-convex optimization problem with $\boldsymbol{Y}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ coupled together. We propose to use coordinate descent scheme by iteratively solving the optimization problem with respect to $\boldsymbol{Y}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ as follows:

(1) Update Y **by fixing** B **and** H. Given the basis matrix $\boldsymbol{B}^s$ and encoders $\boldsymbol{H}^s$ for all views, we seek to solve the following sub-problem:

$$\begin{aligned}\min_{\boldsymbol{Y}}\ O(\boldsymbol{Y}) \ = \alpha \sum_{s=1}^{t} \|\boldsymbol{I}^s(\boldsymbol{H}^s - \boldsymbol{Y}\boldsymbol{B}^s)\|_F^2 \\ +\beta\, tr\left(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}\right) + \lambda\, R(\boldsymbol{Y}) + const\end{aligned} \tag{9}$$

[1]In our implementation, we set l to 5, w_1 to 1 and $w_i = 0.5w_{i-1}$.

where *const* is the constant value independent with the parameter that to be optimized with. Although Eqn.9 is still non-convex, it is smooth and differentiable which enables gradient descent methods for efficient optimization with the calculated gradient:

$$\partial \frac{O(\boldsymbol{Y})}{\boldsymbol{Y}} = 2\boldsymbol{I}^s(\boldsymbol{Y}\boldsymbol{B}^s - \boldsymbol{H}^s)(\boldsymbol{B}^s)^T + 2\beta \boldsymbol{L}\boldsymbol{Y} + 4\lambda \boldsymbol{Y}(\boldsymbol{Y}^T\boldsymbol{Y} - \boldsymbol{I}_d) \quad (10)$$

(2) Update B^s **by fixing** Y **and** H. It is equivalent to solve the following least square problems:

$$\min_{\boldsymbol{B}^s} O(\boldsymbol{B}^s) = \alpha \|\boldsymbol{I}^s(\boldsymbol{H}^s - \boldsymbol{Y}\boldsymbol{B}^s)\|_F^2 + \lambda \|\boldsymbol{B}^s\|_F^2 \quad (11)$$

which has a closed form solution and can be simply derived.

(3) Update H^s **by fixing** Y **and** B. It is a standard autoencoder with an additional regression loss:

$$\min_{\boldsymbol{H}^s} O(\boldsymbol{H}^s) = \|\boldsymbol{X}^s - \tilde{\boldsymbol{X}}^s\|_F^2 + \alpha \|\boldsymbol{I}^s(\boldsymbol{H}^s - \boldsymbol{Y}\boldsymbol{B}^s)\|_F^2 + \lambda\, R(\boldsymbol{H}^s) \quad (12)$$

which can be solved with gradient back-propagation. We then alternate the process of updating $\boldsymbol{Y}$, $\boldsymbol{B}$ and $\boldsymbol{H}$ for several iterations to find a locally optimal solution.

3.6 Binary Embedding

This section connects our work to the quantization-based binary embedding techniques [20, 48, 49], which learn compact binary representations of the data examples for efficient similarity search tasks. Quantization-based binary embedding methods directly binarize the low-dimensional representation to achieve the binary codes. In this work, we can easily obtain the binary codes $\boldsymbol{C}$ for the nodes in the network by binarizing the learned embedding $\boldsymbol{Y}$. However, the quantization error can be further reduced based on the orthogonal invariant property, by minimizing the quantization error between the binary codes and the orthogonal rotation of the embeddings (since $\boldsymbol{Y}\boldsymbol{Q}$ is also an optimal embedding):

$$\begin{aligned} &\min_{\boldsymbol{C},\boldsymbol{Q}} \|\boldsymbol{C} - \boldsymbol{Y}\boldsymbol{Q}\|_F^2 \\ &s.t. \quad \boldsymbol{C} \in \{-1,1\}^{n\times d}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}_d \end{aligned} \quad (13)$$

The above quantization problem is well studied in the literature [20].

3.7 Theoretical Analysis

This section provides some complexity analysis on the training cost of the learning algorithm. The optimization algorithm of DPMNE consists of three steps in each iteration to update Y, B and H. The time complexities for solving Y and B are bounded by $O(tndd_s + tnd^2 + n^2d)$ and $O(tnd^2 + tndd_s)$ respectively. In practice, L is usually a sparse matrix, and the cost can be reduced from $O(n^2d)$ to $O(ld)$ with sparse matrix multiplication, where l is the number of non-zero elements in L. The cost of updating H depends on the number of hidden layers and units in the autoencoder network, which is roughly $O(tnmd_s)$. Here m is the number of unique units in the network. Thus, the total time complexity of the learning algorithm is bounded by $O(tndd_s + ld + tnd^2 + tnmd_s)$.

4 EXPERIMENTS

4.1 Experimental Setting

4.1.1 Datasets. The proposed approach is evaluated on four benchmarks: **Cora**, **DBLP**, **Flickr** and **Last.fm**.

Dataset	#nodes	#total edges	#views	#labels	avg. PDR
Cora	2,708	12,887	2	7	0.02
DBLP	69,110	1,884,236	3	8	0.39
Flickr	6,163	378,547	5	10	0.46
Last.fm	10,197	1,325,367	12	11	0.52

Table 2: A summary of the statistics on all datasets.

- **Cora**[2] is a widely used document corpus from paper citation networks. It contains 2,708 scientific publications classified into one of 7 classes. Citation links and attributes are used as the multiplex, where the features are represented by a 0/1-valued vector indicating the absence/presence of the corresponding link and attribute respectively. The citation network consists of 5,429 edges. For the attribute network, there is an edge between two papers if they share at least one attribute, resulting in 7,458 edges.
- **DBLP**[3] is an author network from the DBLP dataset [45]. It contains 69,110 nodes. Three views are identified including the co-authorship, author-citation and text views. For co-authorship and author-citation views, 0/1-valued feature vectors are used indicating co-authorship and citation between two authors, resulting in 430,117 and 763,029 edges in the corresponding views. For the text view, TF-IDF features are extracted from author's title and abstract. The network is construct based on the text similarity, i.e., there is a link between two authors if their text similarity is high, resulting in 691,090 edges. We select eight diverse research fields as labels including "machine learning", "computational linguistics", "programming language", "data mining", "database", "system technology", "hardware" and "theory". For each field, several representative conferences are selected, and only papers published in these conferences are kept to construct the three views.
- **Flickr** [5] data were collected from the Flickr photo sharing service. It consists of 6,163 users with 10 unique labels. There are 5 views associated with this data: Comment, Favorite, Photo, Tag and User. Here, the views correspond to different aspects of Flickr and edges denote shared interests between users. For example, in the comment view, there is a link between 2 users if they have both commented on the same set of 5 or more photos. All features are 0/1-valued vectors. The resulting five views are: CommentView (2,358 nodes, 13,789 links), FavoriteView (2,724 nodes, 30,757 links), PhotoView (4,061 nodes, 91,329 links), TagView (1,341 nodes, 154,620 links), and UserView (6,163 nodes, 88,052 links).
- **Last.fm** [5] dataset was collected from the music network, with the nodes representing the users and the edges corresponding to different relationships between Last.fm users and other entities. In each view, two users are connected by an edge if they share similar interests, yielding 12 views: ArtistView (2,118 nodes, 149,495 links), EventView (7,240 nodes, 177,000 links), NeighborView (5,320 nodes, 8,387 links), ShoutView (7,488 nodes, 14,486 links), ReleaseView (4,132 nodes, 129,167 links), TagView (1,024 nodes, 118,770 links), TopAlbumView (4,122 nodes,

[2]https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz
[3]https://www.aminer.org/aminernetwork

	Cora		DBLP		Flickr		Last.fm	
methods	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
DeepWalk	0.817 ± 0.022	0.809 ± 0.019	0.709 ± 0.014	0.713 ± 0.010	0.472 ± 0.015	0.451 ± 0.013	0.482 ± 0.011	0.453 ± 0.014
GraphSAGE	0.826 ± 0.015	0.814 ± 0.017	0.718 ± 0.019	0.720 ± 0.021	0.479 ± 0.015	0.462 ± 0.014	0.503 ± 0.011	0.465 ± 0.012
SDNE	0.821 ± 0.021	0.816 ± 0.016	0.706 ± 0.014	0.709 ± 0.018	0.482 ± 0.015	0.454 ± 0.009	0.508 ± 0.011	0.479 ± 0.019
DANE	0.841 ± 0.015	0.832 ± 0.012	0.727 ± 0.015	0.721 ± 0.014	0.485 ± 0.021	0.467 ± 0.013	0.503 ± 0.014	0.481 ± 0.012
IMVC	0.828 ± 0.017	0.820 ± 0.016	0.741 ± 0.013	0.734 ± 0.016	0.478 ± 0.020	0.464 ± 0.012	0.496 ± 0.017	0.478 ± 0.013
MAGCN	0.856 ± 0.022	**0.847 ± 0.021**	0.750 ± 0.025	0.736 ± 0.019	0.508 ± 0.018	0.480 ± 0.015	0.530 ± 0.011	0.508 ± 0.012
HWNN	0.851 ± 0.017	0.843 ± 0.016	0.752 ± 0.023	0.741 ± 0.014	0.502 ± 0.013	0.474 ± 0.021	0.532 ± 0.013	0.511 ± 0.016
DPMNE	**0.859 ± 0.016**	0.844 ± 0.015	**0.784 ± 0.016**	**0.769 ± 0.009**	**0.526 ± 0.011**	**0.512 ± 0.014**	**0.558 ± 0.013**	**0.534 ± 0.015**

Table 3: Node classification results on all datasets. Results are statistically significant with p-value < 0.001.

	Cora	DBLP	Flickr	Last.fm
DeepWalk	0.676	0.573	0.429	0.435
GraphSAGE	0.683	0.589	0.422	0.441
SDNE	0.677	0.594	0.397	0.430
DANE	0.695	0.613	0.454	0.444
IMVC	0.692	0.635	0.457	0.447
MAGCN	0.707	0.628	0.480	0.461
HWNN	0.702	0.631	0.482	0.465
DPMNE	**0.711**	**0.649**	**0.506**	**0.483**

Table 4: Node clustering accuracy results on all datasets.

128,865 links), TopArtistView (6,436 nodes, 12,4731 links), TopTagView (1,296 nodes, 136,104 links), TopTrackView (6,164 nodes, 87,491 links), TrackView (2,680 nodes, 93,358 links), and UserView (10,197 nodes, 38,743 links).

All these multiplex network are suffering from missing data. For example, the CommentView of **Flickr** only has 2,358 users (out of 6,163), where comments are missing from a certain amount of users. In the TagView of **Last.fm**, only 1,024 out of 10,197 users have associated with tags, resulting partial multiplex data. We use Partial Data Ratio (PDR) to represent the fraction of the missing data, e.g., the PDR of the CommentView of **Flickr** is 0.62[4]. The statistics of the datasets with average PDR are given in Table 2.

4.1.2 Baselines. The proposed approach is compared with seven different state-of-the-art baselines, including three single-view methods, **DeepWalk**, **GraphSAGE** and **SDNE** with four multi-view methods, **DANE**, **IMVC**, **MAGCN** and **HWNN**.

- **DeepWalk** [36] learns node representations from random-walk on the networks.
- **GraphSAGE** [23] aggregates the neighbor information to generate node embeddings with graph neural network.
- **SDNE** [47] uses deep neural networks to preserve the structure proximity in network embedding.
- **DANE** [16] uses attribute/tag information associated with the nodes as additional feature and learn embedding with deep neural networks.
- **IMVC** [32] is a multi-view clustering method that explicitly handles incomplete data.
- **MAGCN** [10] learns network embeddings with a multi-view graph convolution networks (GCN).
- **HWNN** [42] is a GNN-based representation learning for heterogeneous hypergraphs, which models multiple non-pairwise relations.

4.1.3 Implementation Details. To apply single-view method, we generate a single network from the multiplex network by placing an edge between a pair of nodes if they are linked by an edge in any view. For DeepWalk, we set the window size as 10, the walk length as 80, and the number of walks as 10 which are the optimal parameters tuned in our experiments. The number of units in the hidden layer is set to 200 in the deep neural network for GraphSAGE, SDNE and DANE. For IMVC, the code is public available[5] and we tune the best hyperparameter with 5-fold cross validation.

For our method, the parameters α, β and λ are tuned by 5-fold cross validation on the training set. To get a fair comparison with deep models, we adopt the same architecture of the neural network, with 200 units in the hidden layer. We set the maximum number of iterations to 60. The number of embedding dimension is set to 128 for all methods (for DANE, each view has dimension of 64). We repeat each experiment 10 times and report the result based on the average over these runs. 50% of the data with random split is used as training.

4.2 Results and Discussion

4.2.1 **Evaluation of Different Methods**. We first evaluate the performance of different methods. To apply the compared deep neural network methods to the partial data, a simple way is to fill in the missing features with 0. However, this may result in large fitting errors across views for the multi-view methods, since the embedding for the missing instance will be 0. Therefore, to achieve stronger baseline results, we replace the missing feature using the linear combination of its 5-nearest neighbor examples, weighted by the similarities, which appear across views. Then the baseline deep methods can be directly applied on these extended data.

We conduct both node classification and clustering on the learned node embedding. Specifically, for classification, we employ L2-regularized logistic regression as the classifier, with Micro-F1 and Macro-F1 as metrics. For clustering, we employ K-means as the clustering method and use clustering accuracy as the metric. The classification results with standard deviations on all datasets are reported in Table 3. From these comparison results, we can see that DPMNE provides the best results among all compared

[4](6,163-2,358)/6,163 = 0.62

[5]https://github.com/xinwangliu/TPAMI_EEIMVC

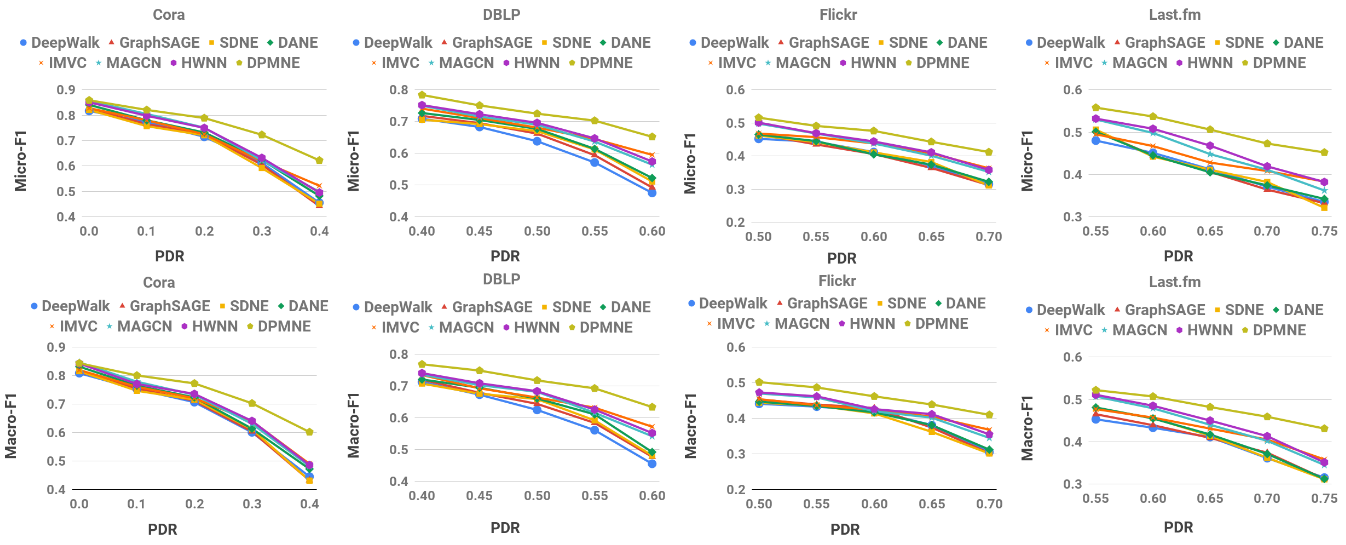

Figure 2: Performance of node classification under different PDRs on all datasets.

Micro-F1	Cora	DBLP	Flickr	Last.fm
w/o deep autoencoder	0.835	0.747	0.479	0.524
w/o partial multiplex	0.852	0.739	0.488	0.517
w/o proximity preservation	0.830	0.721	0.477	0.535
DPMNE	**0.859**	**0.784**	**0.526**	**0.558**

Table 5: Performance of node classification with different model ablations.

methods on all datasets. For example, the Micro-F1 of DPMNE increases over 4.3% and 7.8% compared with both HWNN and DANE on **DBLP**. The reason is that DPMNE can effectively handle the partial data by common latent subspace learning across views and proximity preservation within individual networks, while the compared methods fail to accurately extract a common space from the partial nodes. We observe that DPMNE outperforms IMVC by 8.8%. Although IMVC tries to deal with incomplete data, the network information is not fully explored. We also observe that DPMNE achieves comparable or slightly better results compare with other baselines on **Cora**, whose PDR is very small, i.e., 0.02 from Table 2. This further validates that DPMNE is equally effective on multiplex network without missing data. It can be seen that multi-view methods outperform the single-view methods on all four datasets. The reason is that multi-view methods construct embedding that incorporates complementary information from all views. The clustering result is summarized in Table 4. From this table, we can find that our approach achieves much better clustering performance than the others for most cases, which further verifies the effectiveness of DPMNE.

4.2.2 **Effect of Partial Data Ratio**. To evaluate the effectiveness of the proposed DPMNE under different PDRs, we progressively increase the PDR by randomly removing features from the multiplex network, and compare our method with the other baselines. The node classification results are shown in Figure 2. It can be seen from the figure that when the partial data ratio is 0 (on **Cora**), the data actually becomes the traditional multiplex setting without missing data. As aforementioned, DPMNE is also comparable with other baselines. However, as the PDR increases, our DPMNE approach significantly outperforms other baselines on all datasets. In other words, the performance of DPMNE drops much slower compared to the baseline methods. For example, the Micro-F1 of DPMNE increases over 20% compared with the state-of-the-art GNN-based models, HWNN and MAGCN, on **Cora** with 0.4 PDR. Our hypothesis is that, although the missing data are recovered from the common nodes across views, the baseline deep methods seem less effective in the view missing case. The missing data may not be accurately recovered when the data are missing blockwise for the partial data setting. In other words, the missing nodes can be dissimilar to all the nodes appear across views.

4.2.3 **Ablation Study**. We conduct a series of ablation studies of our model. We first analyze the behavior of each component in DPMNE. There are three key components in our DPMNE, the deep autoencoder, the common latent subspace learning and the network proximity preservation. We train three additional models by removing each of these components separately. Specifically, we remove the deep autoencoder by fixing all its parameters to 1, i.e., both encoder and decoder will map the input to itself. For the other two components, we simply remove the loss terms in the objective. The comparison results are shown in Table 5. It can be seen that all these components are indispensable in DPMNE. Without the common latent subspace learning part, our model degrades to HWNN or MAGCN which is not able to model the partial multiplex data effectively. On the other hand, we observe that deep autoencoder clearly improve the model quality. The reason is that it captures the sparsity and non-linearity in the original feature

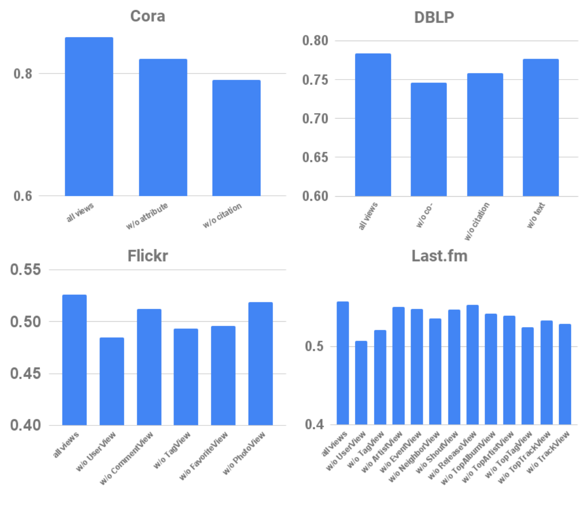

Figure 3: View importance chart on all datasets.

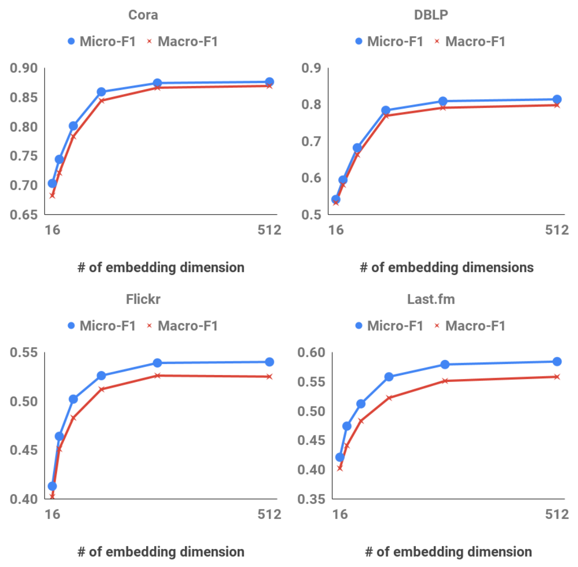

Figure 4: Performance of node classification with different numbers of embedding dimension on all datasets.

Micro-F1	Cora	DBLP	Flickr	Last.fm
CCA-ITQ	0.748	0.657	0.438	0.451
DCMH	0.762	0.683	0.449	0.474
bi-DPMNE	0.788	0.709	0.463	0.482
bi-DPMNE-ITQ	0.803	0.727	0.472	0.495

Table 6: Node classification comparison of binary embedding with 128 bits on all datasets.

space. Lastly, it is obvious that network proximity preservation is crucial in learning network embeddings.

To understand which views are important for learning the network embeddings, we conduct another set of experiments on view importance analysis. Specifically, we build and evaluate the model performance by removing one view from the multiplex network at a time. The node classification results are shown in Figure 3. It can be seen that co-authorship and author-citation views are more important than the text view in **DBLP** dataset, which is consistent with our expectation, as text similarity might not truly reflect the relationships among authors. We also observe that the UserView is the most important view in both **Flickr** and **Last.fm** datasets. The reason is that the UserView directly reveals the connections among the users. Some other useful views are TagView and FavoriteView.

We evaluate the performance of DPMNE with different embedding dimensions by varying the dimension d from {16, 32, 64, 128, 256, 512}. The node classification results on all datasets are shown in Fig.4. It can be seen from the figure that the values of both Micro-F1 and Macro-F1 of DPMNE consistently increase with the increasing of the embedding dimension, from 16 to 256, on all datasets. Our approach achieves similar results between dimension 256 and 512. This observation indicates that very large embedding size is not needed for node representation, which is consistent with the observation in MEGAN [43].

4.2.4 **Effect of Binary Embedding**. We further evaluate our approach in learning binary embeddings. The binary embeddings can be direct achieved by binarizing the learned embeddings, and this method is referred to bi-DPMNE. To obtain more effective binary embeddings, we further conduct iterative quantization based on the orthogonal invariant property of the learned embeddings from Eqn.13. This method is referred to bi-DPMNE-ITQ. We then compare these two methods with two state-of-the-art multiplex binary embedding methods, CCA-ITQ [19, 20] and DCMH [25]. The Micro-F1 results with 128 bits are reported in Table 6. It can be seen from the table that bi-DPMNE consistently performs better than the two baselines on the incomplete data. On the other hand, the bi-DPMNE-ITQ achieves even better results compared to bi-DPMNE, which is consistent with our expectation as it further minimizes the quantization errors.

5 CONCLUSIONS

In this paper, we propose a novel deep network embedding approach to deal with partial multiplex data. We formulate a unified learning framework by simultaneously minimizing the deep reconstruction loss with the autoencoder neural network, ensuring data consistency among different views via common latent subspace learning, and preserving data proximity within the same view through graph Laplacian. Extensive experiments on four benchmarks have demonstrated the effectiveness of the proposed approach. In future, we plan to adopt distributed optimization to speed up the learning process. We also plan to further extend the subspace partial view learning to nonlinear cases.

ACKNOWLEDGMENTS

Xiaojun Quan's contributions were funded in part by the National Natural Science Foundation of China (No. 62176270).

REFERENCES

[1] Mohammadreza Armandpour, Patrick Ding, Jianhua Huang, and Xia Hu. 2019. Robust Negative Sampling for Network Embedding. In *AAAI*. 3191–3198.

[2] Sambaran Bandyopadhyay, Lokesh N, Saley Vishal Vivek, and M. Narasimha Murty. 2020. Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding. In *WSDM*. 25–33.

[3] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*. 585–591.

[4] Ayan Kumar Bhowmick, Koushik Meneni, Maximilien Danisch, Jean-Loup Guillaume, and Bivas Mitra. 2020. LouvainNE: Hierarchical Louvain Method for High Quality and Scalable Network Embedding. In *WSDM*. 43–51.

[5] Ngot Bui, Thanh Le, and Vasant G. Honavar. 2016. Labeling actors in multi-view social networks by integrating information from within and across multiple views. In *BigData*. 616–625.

[6] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *SIGKDD*. 1358–1368.

[7] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. 2015. Heterogeneous Network Embedding via Deep Architectures. In *SIGKDD*. 119–128.

[8] Weijian Chen, Fuli Feng, Qifan Wang, Xiangnan He, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. CatGCN: Graph Convolutional Networks with Categorical Node Features. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[9] Xia Chen, Guoxian Yu, Jun Wang, Carlotta Domeniconi, Zhao Li, and Xiangliang Zhang. 2019. ActiveHNE: Active Heterogeneous Network Embedding. In *IJCAI*. 2123–2129.

[10] Jiafeng Cheng, Qianqian Wang, Zhiqiang Tao, De-Yan Xie, and Quanxue Gao. 2020. Multi-View Attribute Graph Convolution Networks for Clustering. In *IJCAI*. 2973–2979.

[11] Calin Cruceru, Gary Bécigneul, and Octavian-Eugen Ganea. 2021. Computationally Tractable Riemannian Manifolds for Graph Embeddings. In *AAAI*.

[12] Lun Du, Zhicong Lu, Yun Wang, Guojie Song, Yiming Wang, and Wei Chen. 2018. Galaxy Network Embedding: A Hierarchical Community Structure Preserving Approach. In *IJCAI*. 2079–2085.

[13] Fuli Feng, Weiran Huang, Xiangnan He, Xin Xin, Qifan Wang, and Tat-Seng Chua. 2021. Should graph convolution trust neighbors? a simple causal inference method. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1208–1218.

[14] Ming-Han Feng, Chin-Chi Hsu, Cheng-Te Li, Mi-Yen Yeh, and Shou-De Lin. 2019. MARINE: Multi-relational Network Embeddings with Relational Proximity and Node Attributes. In *WWW*. 470–479.

[15] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*. 2331–2341.

[16] Hongchang Gao and Heng Huang. 2018. Deep Attributed Network Embedding. In *IJCAI*. 3364–3370.

[17] Hongchang Gao, Jian Pei, and Heng Huang. 2019. ProGAN: Network Embedding via Proximity Generative Adversarial Network. In *SIGKDD*. 1308–1316.

[18] Lin Gong, Lu Lin, Weihao Song, and Hongning Wang. 2020. JNET: Learning User Representations via Joint Network Embedding and Topic Embedding. In *WSDM*. 205–213.

[19] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. 2014. A Multi-View Embedding Space for Modeling Internet Images, Tags, and Their Semantics. *Int. J. Comput. Vis.* 106, 2 (2014), 210–233.

[20] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *TPAMI* 35, 12 (2013), 2916–2929.

[21] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*. 855–864.

[22] Jun Guo and Jiahui Ye. 2019. Anchors Bring Ease: An Embarrassingly Simple Approach to Partial Multi-View Clustering. In *AAAI*. 118–125.

[23] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[24] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. 2704–2710.

[25] Qing-Yuan Jiang and Wu-Jun Li. 2017. Deep Cross-Modal Hashing. In *CVPR*. 3270–3278.

[26] Song Jiang, Bernard Koch, and Yizhou Sun. 2021. HINTS: Citation Time Series Prediction for New Publications via Dynamic Heterogeneous Information Network Embedding. In *WWW*. 3158–3167.

[27] Wenhao Jiang, Hongchang Gao, Fu-Lai Chung, and Heng Huang. 2016. The l2, 1-Norm Stacked Robust Autoencoders for Domain Adaptation. In *AAAI*. 1723–1729.

[28] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In *NeurIPS*. 4465–4478.

[29] Shao-Yuan Li, Yuan Jiang, and Zhi-Hua Zhou. 2014. Partial Multi-View Clustering. In *AAAI*. 1968–1974.

[30] Xiaohe Li, Lijie Wen, Chen Qian, and Jianmin Wang. 2020. GAHNE: Graph-Aggregated Heterogeneous Network Embedding. *CoRR* abs/2012.12517 (2020).

[31] Yu Li, Ying Wang, Tingting Zhang, Jiawei Zhang, and Yi Chang. 2019. Learning Network Embedding with Community Structural Information. In *IJCAI*. 2937–2943.

[32] Xinwang Liu, Miaomiao Li, Chang Tang, Jingyuan Xia, Jian Xiong, Li Liu, Marius Kloft, and En Zhu. 2020. Efficient and Effective Regularized Incomplete Multi-view Clustering. In *TPAMI*.

[33] Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-Preserving Dynamic Attributed Network Embedding. In *WWW*. 1629–1638.

[34] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *WSDM*. 393–401.

[35] Guosheng Pan, Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu. 2021. Unsupervised Attributed Network Embedding via Cross Fusion. In *WSDM*. 797–805.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD*. 701–710.

[37] Léo Pio-Lopez, Alberto Valdeolivas, Laurent Tichit, Elisabeth Remy, and Anaïs Baudot. 2020. MultiVERSE: a multiplex and multiplex-heterogeneous network embedding approach. *CoRR* abs/2008.10085 (2020).

[38] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An Attention-based Collaboration Framework for Multi-View Network Representation Learning. In *CIKM*. 1767–1776.

[39] Nishant Rai, Sumit Negi, Santanu Chaudhury, and Om Deshmukh. 2016. Partial Multi-View Clustering using Graph Regularized NMF. In *ICPR*. 2192–2197.

[40] Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reason.* 50, 7 (2009), 969–978.

[41] Yu Shi, Fangqiu Han, Xinran He, Carl Yang, Jie Luo, and Jiawei Han. 2018. mvn2vec: Preservation and Collaboration in Multi-View Network Embedding. *CoRR* abs/1801.06597 (2018).

[42] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. 2021. Heterogeneous Hypergraph Embedding for Graph Classification. In *WSDM*. 725–733.

[43] Yiwei Sun, Suhang Wang, Tsung-Yu Hsieh, Xianfeng Tang, and Vasant G. Honavar. 2019. MEGAN: A Generative Adversarial Network for Multi-View Network Embedding. In *IJCAI*. 3527–3533.

[44] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. 1067–1077.

[45] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *SIGKDD*. 990–998.

[46] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. 2018. Deep Recursive Network Embedding with Regular Equivalence. In *SIGKDD*. 2357–2366.

[47] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *SIGKDD*. 1225–1234.

[48] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2018. A Survey on Learning to Hash. *TPAMI* 40, 4 (2018), 769–790.

[49] Qifan Wang. 2015. *Learning compact hashing codes with complex objectives from multiple sources for large scale similarity search*. Ph.D. Dissertation. Purdue University.

[50] Qifan Wang, Gal Chechik, Chen Sun, and Bin Shen. 2017. Instance-level label propagation with multi-instance learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2943–2949.

[51] Qianqian Wang, Zhengming Ding, Zhiqiang Tao, Quanxue Gao, and Yun Fu. 2020. Generative Partial Multi-View Clustering. *CoRR* abs/2003.13088 (2020).

[52] Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. WebFormer: The Web-page Transformer for Structure Information Extraction. In *WWW*.

[53] Qifan Wang and Ruining He. 2021. Deep Multi-View Network Embedding on Incomplete Data. US Patent App. 17/154,123.

[54] Qifan Wang, Luo Si, and Bin Shen. 2015. Learning to Hash on Partial Multi-Modal Data. In *IJCAI*. 3904–3910.

[55] Suhang Wang, Jiliang Tang, Charu C. Aggarwal, and Huan Liu. 2016. Linked Document Embedding for Classification. In *CIKM*. 115–124.

[56] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A. Bilmes. 2015. On Deep Multi-View Representation Learning. In *ICML*. 1083–1092.

[57] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[58] Zekai Wang, Hongzhi Liu, Yingpeng Du, Zhonghai Wu, and Xing Zhang. 2019. Unified Embedding Model over Heterogeneous Information Network for Personalized Recommendation. In *IJCAI*. 3813–3819.

[59] Xiaokai Wei, Bokai Cao, Weixiang Shao, Chun-Ta Lu, and Philip S. Yu. 2016. Community detection with partially observable links and node attributes. In *BigData*.

[60] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. 2022. Graph convolution machine for context-aware recommender

system. *Frontiers of Computer Science* 16, 6 (2022), 1–12.
[61] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24.
[62] Yuanzhen Xie, Zijing Ou, Liang Chen, Yang Liu, Kun Xu, Carl Yang, and Zibin Zheng. 2021. Learning and Updating Node Embedding on Dynamic Heterogeneous Information Network. In *WSDM*. 184–192.
[63] Cai Xu, Ziyu Guan, Wei Zhao, Hongchang Wu, Yunfei Niu, and Beilei Ling. 2019. Adversarial Incomplete Multi-view Clustering. In *IJCAI*. 3933–3939.
[64] Hansheng Xue, Luwei Yang, Vaibhav Rajan, Wen Jiang, Yi Wei, and Yu Lin. 2021. Multiplex Bipartite Network Embedding using Dual Hypergraph Convolutional Networks. In *WWW*. 1649–1660.
[65] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment. In *WWW*. 3907–3917.
[66] Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure Leskovec. 2020. MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In *SIGKDD*. 2434–2443.
[67] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *CoRR* abs/2004.00216 (2020).
[68] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. MAVE: A Product Dataset for Multi-Source Attribute Value Extraction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1256–1265.
[69] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. SINE: Scalable Incomplete Network Embedding. In *ICDM*. 737–746.
[70] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *IJCAI*. 3082–3088.
[71] Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion Maps for Textual Network Embedding. In *NIPS*. 7598–7608.
[72] Yizhou Zhang, Guojie Song, Lun Du, Shuwen Yang, and Yilun Jin. 2019. DANE: Domain Adaptive Network Embedding. In *IJCAI*. 4362–4368.
[73] Jianan Zhao, Xiao Wang, Chuan Shi, Zekuan Liu, and Yanfang Ye. 2020. Network Schema Preserving Heterogeneous Information Network Embedding. In *IJCAI*, Christian Bessiere (Ed.). 1366–1372.
[74] Shijie Zhu, Jianxin Li, Hao Peng, Senzhang Wang, Philip S. Yu, and Lifang He. 2021. Adversarial Directed Graph Embedding. In *AAAI*. 4741–4748.