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.

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/XXXXXXX.XXXXXX

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, 45, 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].

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
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.

The overall model architecture is shown in Figure 1. The three views in the multiplex network example, with missing data from each individual views, i.e., $x_4$ is missing from view 1, $x_3$ 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.

3 DEEP PARTIAL MULTIPLEX NETWORK EMBEDDING

3.1 Problem Definition

Given a multiplex network $G=(V,E)$, where $V=\{v_i \mid i=1, \ldots, n\}$ denotes the set of nodes. $X=\{X^1, X^2, \ldots, X^d\}$ are the multiplex features associated with the nodes, where $t$ is the total number of views, $X^t=\{x^t_i \mid i=1, 2, \ldots, n\}$ with $x^t_i \in \mathbb{R}^d$, is the feature of the $i$-th node in the $s$-th view. $E=\{E^s\}$ denotes the edge sets in the multiplex network, where $E^s_{ij}=1$ indicates that $v_i$ and $v_j$ are linked in the $s$-th view. In the partial data setting, a partial data $\hat{X}=\{\hat{X}^1, \hat{X}^2, \ldots, \hat{X}^d\}$ instead of $X$ is given, where some of the node features are missing from each individual views, e.g., $x^1_4$ is missing from View 1 and $x^2_3$ is missing from View 2 in Figure 1. The purpose of DPMNE is to learn a low-dimensional embedding representation $\hat{Y}=\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\} \in \mathbb{R}^{n \times d}$ of $G$, 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:

$$h = \sigma(W^{(en)} x + b^{(en)}), \quad \hat{x} = \sigma(W^{(de)} h + b^{(de)})$$

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

$$h_1 = \sigma(W^{(en)} x + b^{(en)}), \quad h_k = \sigma(W^{(en)} h_{k-1} + b^{(en)})$$

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

\[
\min_H \sum_{s=1}^{l} \|X^s - \hat{X}^s\|_F^2
\]  

(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 to learn 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:

\[
\min_{Y,B,H} \sum_{s=1}^{l} \|I^s(H^s - YB^s)\|_F^2 + \lambda R(Y,B,H)
\]  

(4)

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

In the above formulation, the individual basis matrix \(B^s\), which are learned from all available instances from all views, are connected by the common latent representation \(Y\). 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., \(h^s_i = y_iB^s\). By solving the above problem, the deep representation \(H\), the latent space basis \(B\) and the homogeneous feature representation \(Y\) 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 \(Y\) 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:

\[
\min_Y \sum_{i,j} P_{ij}^s \|y_i - y_j\|^2 = \sum_{i,j} P_{ij} \|y_i - y_j\|^2
\]  

(5)

here \(P_{ij} = \sum_s P_{ij}^s\) and \(P^s\) is the proximity matrix in \(s\)-th view, which can be obtained from the edges in \(s\)-th network \(E^s\). A simple way to define \(P^s\) is to directly use the first-order proximity, i.e., \(P^s = E^s\). 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 \(P^s\) as:

\[
P^s = \sum_{l=1}^L w_l E^s + \sum_{l=2}^L w_l (E^s)^2 + \cdots + w_l (E^s)^l
\]  

(6)

where \(l\) is the order, and \(w_1, \ldots, w_l\) are the weights for each term\(^1\). Matrix \((E^s)^l\) 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 \(D\), whose entries are given by \(D_{ii} = \sum_s P_{ii}^s\). Eqn.5 can be rewritten as:

\[
tr\left(Y^T (D - P) Y\right) = tr\left(Y^T L Y\right)
\]  

(7)

where \(L\) 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:

\[
\min_{Y,B,H} \sum_{s=1}^{l} \|I^s(H^s - YB^s)\|_F^2 + \alpha \sum_{s=1}^{l} \|I^s(H^s - YB^s)\|_F^2 + \beta \sum_{i,j} P_{ij} \|y_i - y_j\|^2 + \lambda R(Y,B,H)
\]  

(8)

where \(\alpha, \beta\) and \(\lambda\) 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 respect to \(Y, B\) and \(H\) coupled together. We propose to use coordinate descent scheme by iteratively solving the optimization problem with respect to \(Y, B\) and \(H\) as follows:

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

\[
\min_Y O(Y) = \alpha \sum_{s=1}^{l} \|I^s(H^s - YB^s)\|_F^2
\]  

(9)

\[+ \beta \sum_{i,j} P_{ij} \|y_i - y_j\|^2 + \lambda R(Y) + \text{const}
\]

\(^1\)In our implementation, we set \(l = 5\), \(w_1 = 1\) and \(w_5 = 0.5, w_{5-1}\).
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:

\[
\frac{\partial O(Y)}{\partial Y} = 2\Gamma(YB^\top - H^\top)(B^\top)^\top + 2\beta LY + 4\alpha Y(Y^\top Y - I_d) \tag{10}
\]

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

\[
\min_{B^\top} O(B^\top) = \|Y - (\Gamma^\top X)\|_F^2 + \alpha \|\Gamma^\top X - YB^\top\|_F^2 + \lambda \|B^\top\|_F^2 \tag{11}
\]

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

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

\[
\min_{H^\top} O(H^\top) = \|X^\top - \hat{X}^\top\|_F^2 + \alpha \|\Gamma^\top (Y - YB^\top)\|_F^2 + \lambda R(H^\top) \tag{12}
\]

which can be solved with gradient back-propagation. We then alternate the process of updating \( Y, B \) and \( H \) 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 \( C \) for the nodes in the network by binarizing the learned embedding \( Y \). 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 \( Y_Q \) is also an optimal embedding):

\[
\min_{C, Q} \|C - Y_Q\|_F^2 \tag{13}
\]

s.t. \( C \in (-1, 1)^{n \times d}, \quad Q^\top Q = I_d \)

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 \( O(t ndd_n + tnd^2 + n^2d) \) and \( O(tnd^2 + tndd_n) \) 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 \( O(tnmd) \). Here \( m \) is the number of unique units in the network. Thus, the total time complexity of the learning algorithm is bounded by \( O(tndd_n + ld + tnd^2 + tnmdd_n) \).

### 4 EXPERIMENTS

#### 4.1 Experimental Setting

| 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**
  - 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**
  - 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 constructed 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**
  - 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**
  - Data were 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, 31,647 links).

3https://linqs-data.soe.ucsc.edu/public/lbc/cora.tgz
4https://www.aminer.org/aminernetwork

---

4.1.1 Datasets. The proposed approach is evaluated on four benchmarks: Cora, DBLP, Flickr and Last.fm.
All these multiplex network are suffering from missing data. For view methods, DeepWalk different state-of-the-art baselines, including three single-view statistics of the datasets with average PDR are given in Table 2. Flickr data, e.g., the PDR of the CommentView of Partial Data Ratio (PDR) to represent the fraction of the missing Last.fm of 6,163), where comments are missing from a certain amount of Table 4: Node clustering accuracy results on all datasets.

| methods       | Cora Micro-F1 | Cora Macro-F1 | DBLP Micro-F1 | DBLP Macro-F1 | Flickr Micro-F1 | Flickr Macro-F1 | Last.fm Micro-F1 | Last.fm Macro-F1 |
|---------------|---------------|---------------|---------------|---------------|----------------|----------------|----------------|----------------|
| DeepWalk      | 0.817 ± 0.022 | 0.809 ± 0.019 | 0.709 ± 0.014 | 0.715 ± 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 |
| DFMNE         | 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 4: Node clustering accuracy results on all datasets.

| methods | 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 |
| DFMNE | 0.711 | 0.649 | 0.506 | 0.483 |

128,865 links), TopArtistView (6,436 nodes, 12,473 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. 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\(^3\) and we tune the best hyperparameter with 5-fold cross validation.

For our method, the parameters \(\alpha\), \(\beta\) and \(\lambda\) 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 DFMNE provides the best results among all compared

\(^3\)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 compared 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.
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 directly 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

This work is supported by the National Natural Science Foundation of China (No. 62176270).
[66] C. Yang, A. Pal, A. Zhai, N. Pancha, J. Han, C. Rosenberg, and J. Leskovec. Multisage: Empowering GCN with contextualized multi-embeddings on web-scale multipartite networks. In SIGKDD, pages 2434–2443, 2020.

[67] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. CoRR, abs/2004.00216, 2020.

[68] L. Yang, Q. Wang, Z. Yu, A. Kulkarni, S. Sanghavi, B. Shu, J. Elsas, and B. Kanagal. Mave: A product dataset for multi-source attribute value extraction. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, page 1256–1265, 2022.

[69] D. Zhang, J. Yin, X. Zhu, and C. Zhang. SINE: scalable incomplete network embedding. In ICDM, pages 737–746, 2018.

[70] H. Zhang, L. Qiu, L. Yi, and Y. Song. Scalable multiplex network embedding. In IJCAI, pages 3082–3088, 2018.

[71] X. Zhang, Y. Li, D. Shen, and L. Carin. Diffusion maps for textual network embedding. In NIPS, pages 7598–7608, 2018.

[72] Y. Zhang, G. Song, L. Du, S. Yang, and Y. Jin. DANE: domain adaptive network embedding. In IJCAI, pages 4362–4368, 2019.

[73] J. Zhao, X. Wang, C. Shi, Z. Liu, and Y. Ye. Network schema preserving heterogeneous information network embedding. In C. Bessiere, editor, IJCAI, pages 1366–1372, 2020.

[74] S. Zhu, J. Li, H. Peng, S. Wang, P. S. Yu, and L. He. Adversarial directed graph embedding. In AAAI, pages 4741–4748, 2021.