Hindawi
Complexity
Volume 2021, Article ID 2260488, 10 pages
https://doi.org/10.1155/2021/2260488

Research Article

Layer Information Similarity Concerned Network Embedding

Ruili Lu,[1] **Pengfei Jiao**,[2] **Yinghui Wang**,[3] **Huaming Wu**,[4] **and Xue Chen**[2]

[1]*Tianjin International Engineering Institute, Tianjin University, Tianjin 300072, China*
[2]*Law School of Tianjin University, Tianjin 300072, China*
[3]*College of Intelligence and Computing, Tianjin University, Tianjin 300350, China*
[4]*Center for Applied Mathematics, Tianjin University, Tianjin 300072, China*

Correspondence should be addressed to Xue Chen; xuechen@tju.edu.cn

Received 10 June 2021; Accepted 17 August 2021; Published 26 August 2021

Academic Editor: Fei Xiong

Great achievements have been made in network embedding based on single-layer networks. However, there are a variety of scenarios and systems that can be presented as multiplex networks, which can reveal more interesting patterns hidden in the data compared to single-layer networks. In the field of network embedding, in order to project the multiplex network into the latent space, it is necessary to consider richer structural information among network layers. However, current methods for multiplex network embedding mostly focus on the similarity of nodes in each layer of the network, while ignoring the similarity between different layers. In this paper, for multiplex network embedding, we propose a **L**ayer **I**nformation **S**imilarity **C**oncerned **N**etwork **E**mbedding (LISCNE) model considering the similarities between layers. Firstly, we introduce the common vector for each node shared by all layers and layer vectors for each layer where common vectors obtain the overall structure of the multiplex network and layer vectors learn semantics for each layer. We get the node embeddings in each layer by concatenating the common vectors and layer vectors with the consideration that the node embedding is related not only to the surrounding neighbors but also to the overall semantics. Furthermore, we define an index to formalize the similarity between different layers and the cross-network association. Constrained by layer similarity, the layer vectors with greater similarity are closer to each other and the aligned node embedding in these layers is also closer. To evaluate our proposed model, we conduct node classification and link prediction tasks to verify the effectiveness of our model, and the results show that LISCNE can achieve better or comparable performance compared to existing baseline methods.

1. Introduction

In the past few decades, network embedding has obtained remarkable achievements. The basic idea is converting a node into a low-dimensional space in which the network structure and properties can be preserved effectively. In the early period, traditional models such as MDS [1], Isomap [2], LLE [3], and LE [4] are mainly based on dimensionality reduction technologies. These models are not suitable for large networks due to their computational complexity. As Word2Vec [5] plays a vital role in the field of natural language processing, random walk-based methods that regard nodes in the network as words are proposed, such as DeepWalk [6] and Node2Vec [7]. In recent years, with the continuous development of deep learning, SDNE [8], DNGR [9], and GCNs [10] have developed neural networks into network embedding models.

The methods mentioned above are all designed for single-layer networks. Figure 1(a) shows an example of a single-layer network, through which we can see that there is only one relation in the network. However, there are still many complex scenarios in the real world that cannot be described by single-layer networks. For example, the same set of individuals in social networks may participate in Twitter, Facebook, or Weibo for different purposes. Interactions in different social networks can be represented by a single-layer network. Each layer of the network has a specific relationship and specific semantics. However, these single-layer networks do not operate in isolation and there are always connections between them. Instead, these complex

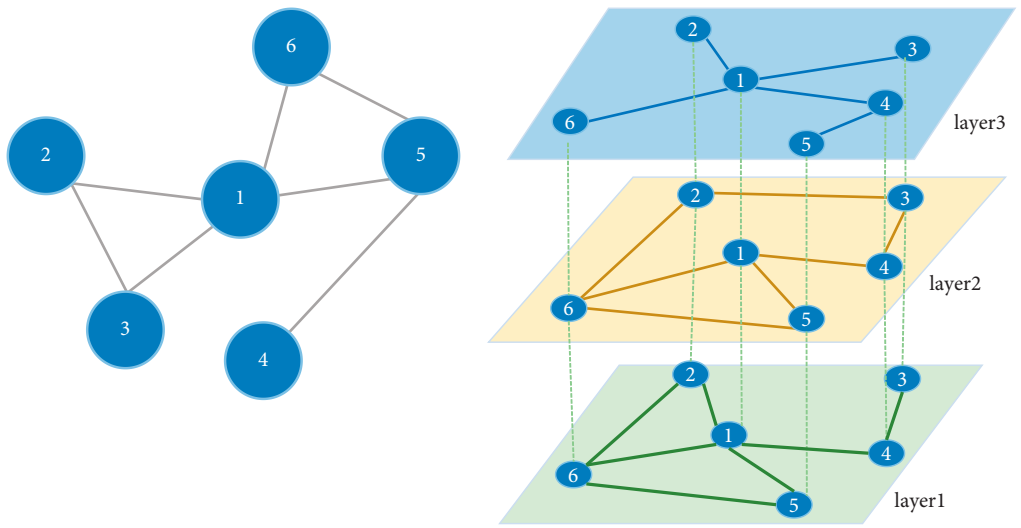

Figure 1: The toy examples of single-layer network and multiplex network. (a) Single-layer network. (b) Multiplex network.

scenes can be represented as a multiplex network, which is also a multilayer network in which layers share the same set of nodes. Each layer in a multiplex network represents a particular relationship of nodes, and the structure of each layer is typically associated. Figure 1(b) illustrates an example of an undirected multiplex network, and it has a unique structure in different layers while there also exist correlations between layers. Unlike the single-layer network, there are three relationships among a set of nodes, each of which describes a unique interaction in the given network structure. Multiplex relationships cannot be captured using single-layer methods. Therefore, it is necessary to conduct in-depth research on multiplex network embedding.

Compared with the single-layer network, one of the challenges for multiplex network embedding is how to aggregate the diverse types of structure in different layer networks without destroying their unique properties. To solve this problem, MCGE [11], MANE [12], and MVNE [13] use the tensor factorization to concurrently capture the main local structure and correlations between different layers. MNE [14] and MGCN [15] define one common vector shared by all layers to capture the shared information in all layer networks and low-dimensional node vectors in each layer to capture the unique properties. In addition to introducing the common vector, CrossMNA [16] also introduces a layer vector to extract the semantic meaning. One2Multi [17] uses one encoder to encode the most informative network from which we can extract the shared information and multiple decoders to reconstruct all layers learning the specific structure in each layer. DMNE [18] and MrMine [19] take advantage of the links between subgraphs or communities to learn the cross-network relationships.

While each layer in the multiplex network is constructed from different semantics and makes the structure of each layer different, the varying relatedness between different semantics leads to diverse structural similarities between different layers. For example, we can observe from Figure 1(b) that layer2 and layer3 have more of the same edges between nodes compared to layer1, that is, the structure of layer2 and layers3 is more similar than that of layer1. Also, the similarity of any two layers is always different, which leads to the divergence in different layers of network analysis. It has been proved that considering interlayer similarity can significantly improve the performance of link prediction [20] and community detection [21] in multiplex networks. Hence, it is an essential feature that should not be ignored in multiplex network embedding. However, the existing methods can obtain embedded representations of a multiplex network, and most of them fail to consider the similarities between different layers which is an important characteristic in the multiplex network.

To incorporate layer similarities when learning node vectors or layer vectors, we propose a novel model, Layer Information Similarity Concerned Network Embedding (LISCNE), and our model takes advantage of the common and local features in multiplex networks and exploits layer similarity at the same time.

Specifically, we firstly obtain node embeddings by concatenating common vector for each node shared by all layers and layer vector for each layer. Common vectors capture characteristics shared by cross layer by merging all the networks into a new single-layer network and training the common vector for each node in the new network. In addition, layer vectors learn the overall semantics for each layer. Then, to model the layer similarity, we define an index to formalize the similarity between different layers. With the constraint of layer similarities, we force the vectors with greater similarity to be closer.

The major contributions are summarized as follows:

(i) After investigating the existing multiplex network embedding methods, we find that the methods consider the node connectivity among layers but ignore the inter-layer similarities.

(ii) We propose a novel **L**ayer **I**nformation **S**imilarity **C**oncerned **N**etwork **E**mbedding (LISCNE) model, which effectively exploits the overall and local structure in multiplex networks and combines the concept of layer vectors with layer similarity at the same time.

(iii) We conduct experiments to evaluate the proposed method using several real-world datasets on link prediction and node classification tasks. Compared with existing benchmark methods, LISCNE can achieve better or comparable performance.

2. Related Work

In this section, we review related work from two main aspects, namely, single-layer network embedding and multiplex network embedding.

2.1. Single-Layer Network Embedding. By assuming that the more similar the structure of nodes is, the closer their representation vectors are, the network embedding can learn latent low-dimensional representations for the nodes or links in a network. Earlier studies [2, 3, 22–24] were mainly based on matrix factorization. Isomap [2] obtained the shortest path d_{ij} between node i and node j by constructing a neighborhood graph with connectivity algorithms and then obtained the vector presentation by minimizing the function of $(d_{ij} - \|u_i - u_j\|)^2$. GraRep [24] defined a node transition probability and preserved k-order proximity. Inspired by Word2Vec [5], new types of methods [6, 7, 25, 26] using skip-gram model [27] have gradually emerged. The goal of the skip-gram model is to maximize the co-occurrence probability based on the context in a sentence:

$$\max \prod_{v_i \in V} \prod_{v_j \in \text{context}(v_i)} \Pr\left(\phi(v_i) | \phi\left(v_j\right)\right). \tag{1}$$

DeepWalk [6] regarded each vertex in the network as a word. It applied the Depth-First Sampling (DFS) strategy to obtain walk sequences when conducting random walks and performed the skip-gram algorithm for training the sequences. Node2Vec [7] employed a biased random walk strategy when getting the walk sequence. It defined two parameters p and q to adjust between BFS and DFS during random walks. Topo2Vec [26] used a greedy goal-based searching strategy to generate the node context and obtain the local and global topologically proximal nodes in a network. While these random walk-based methods cannot model the nonlinear structural information, some methods based on deep neural networks [8–10, 28–30] have been proposed. Both SDNE [8] and DNGR [9] used deep autoencoders, where SDNE used the encoder to preserve the first- and second-order proximity of nodes, while DNGR captured higher-order proximity by using PPMI matrix which is indirectly transformed by the probabilistic co-occurrence matrix created by random surfing. GCNs [10] iteratively aggregated previous node embeddings and their neighbor embeddings to learn the new node embeddings. VGAE [28] was an inference model parameterized by a two-layer GCN. Pedronette and Latecki [31] proposed rank-based self-training to improve the accuracy of GCNs on semisupervised classification tasks. Recently, some novel algorithms [32–34] in the field of Contrastive Self-Supervised Learning have yielded good results. The core is to measure the similarities of sample pairs in a representation space, and the similarity between positive samples is much greater than the negative samples. These models are performed on the single-layer network. More discussion and methods for network embedding can be found in [35–38].

2.2. Multiplex Network Embedding. To better represent the multiplex networks used to describe the real-world data, there also exist various works for multiplex network embedding.

MCGE [11] applied tensor factorization and defined a multiview kernel tensor to obtain common latent factors that capture the global structure information. Random walks have been applied in network embedding [14, 19, 39–42]. MNE [14] learned two vectors for a node at the same time, i.e., a common vector sharing by all layers and a lower-dimensional vector for an individual layer. Then, it introduced a transformation matrix to align these two vectors. PMNE's [39] network aggregation and result aggregation are essentially single-layer approaches. Considering the interactions between layers, the co-analysis method can traverse between layers with a probability r when taking a random walk. GATNE [43] proposed a unified framework to address the problem of embedding learning for attributed multiplex heterogeneous networks, and GATNE-T was a generalization of MNE [14] when training edge embeddings directly. MrMine [19] simultaneously learned the multinetwork representation at three resolutions of network, subgraph, and nodes, and it further constructed cross-resolution including network-subgraph, subgraph-node, node-node context. HMNE [44] defined a heuristic 3D interactive walk and sampled sequences of node cross layers. It preserved cross-layer neighborhood of nodes and learned information of multitype relations into a unified embedding space.

MVE [45] learned the robust representation by promoting the collaboration of different layers and different weights which were assigned to layers during voting. CrossMNA [16] defined a network vector extracting the semantic meaning of the network and an inter-vector reflecting the common features of the anchor nodes in different networks. Then, these two vectors were added to form an intra-vector, which preserved the specific structural feature for a node in its selected network. MGCN [46] extended GCN to multiplex networks, which defined a general vector and dimension-specific vector to capture the common and individual layer information. TCMGC [47] developed a multilayer GCN to

capture the structure and multiview information. DMNE [18] used an encoder for all individual networks and regularized the cross-network embeddings through two types of loss functions to penalize the embedding inconsistency. DMGI [48] was an unsupervised model based on DGI [49]. In an individual layer, it performed the DGI algorithm to get the relation-type specific embedding and then took advantage of the multiplexity of the network by introducing consensus regularization and multiheaded attention mechanisms. MEGAN [50] was a multiplex GAN that designed a multilayer generator to model multilayer connectivity to generate fake samples and a node pair discriminator to enforce the generator to more accurately t the distribution of multilayer network connectivity. One2Multi [17] used the network with the most information as the input of encoder to learn the shared information of all the networks and then used a multidecoder to reconstruct the multiplex network from the shared information.

All the single-layer models mentioned above are effective for single network embedding; however, they do not consider the correlation in the multiplex network. In addition, the GCN-based multiplex network embedding models only consider the local information in the network, while other models ignore the similarities between layers. Our model combines the similarities between the layers and can simultaneously capture the local and global information in the network and the multiplex relationships between layers.

3. Notations and Problem Formulation

We begin with a formal definition of multiplex network, followed by the problem formulation. For the sake of clarity, the main notations are summarized in Table 1.

Definition 1 (multiplex network). A multiplex network consists of a set of N nodes $V = (v_1, v_2, \ldots, v_N)$ and L layer. All layers share the same set of nodes V and the nodes form diverse structures in each layer. The structure layer l can be represented as ε_l. We denote this multiplex network as $G = \{G^1, G^2, \ldots, G^L\} = \{V, \varepsilon\}$, where $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_L\}$.

Given such a multiplex network with L layers, the goal of our work is to learn low-dimensional embeddings $Z_i^l \in R^d$ for each node v_i on each individual network G^l, where d is the dimension of the embedding. The learned representations can be used as features in a variety of applications such as node classification and visualization, relationship mining, and link prediction. In our experiments, we perform both link prediction and node classification tasks to verify the effectiveness of the learned embedding.

4. Layer Information Similarity Concerned Network Embedding

As the nodes in each layer of the multiplex network are same, they shared the common information and the same node may show some similar features among layers. However, the structure among nodes in each layer is formed by different semantics and thus leads to quite diverse local structures of this node in each layer, and the varying relatedness between different semantics also leads to diverse structural similarities between different layers. In this paper, we propose LISCNE which models the common and local features in multiplex networks and exploits layer similarity at the same time.

Table 1: Main symbols and their definitions.

Symbol	Definition
L	The number of layers in the multiplex network
G^l	The network for layer l in the multiplex network
N	The number of nodes
V	The node set of the multiplex network
ε_l	The edge set of l-th network
r^l	The layer vector for l-th network
u_i	The common vector for node v_i
U	The common vector matrix for all nodes
Z_i^l	The embedding vector for node v_i in network G^l
d_1	The dimension of common vector
d_2	The dimension of layer vector
d	The dimension of final node vector
S	The similarity matrix between networks
$S^{\alpha\beta}$	The similarity between networks G^α and G^β

Figure 2 illustrates the framework of LISCNE for a three-layer multiplex network. The architecture contains two components. The first part is modeling the common vector for all nodes that are shared by the counterpart nodes among different layers. The second part is learning the node embedding in each layer by integrating the common layer and layer vector introduced to capture distinct semantic information of different networks. The last part is describing the process of training layer vectors with layer similarities. The embedding for node v_i in layer l is defined as

$$Z_i^l = f\left(u_i + r^l\right), \tag{2}$$

where f is the map function integrating common vector and layer vector to get the final node presentation. In our model, we use concatenation as the map function. LISCNE specifies the relationship of different networks by the layer similarities, i.e., S^{12} indicates the index of structural similarity of network G_1 and network G_2. By adding layer similarities to the layer vector, it can associate within-network and cross-network structure information.

Next, we will describe our model LISCNE in detail and introduce it in three parts: common feature modeling, learning node embedding in each layer, and integrating the similarity between layers.

4.1. Common Feature Modeling. In this part, we learn the common feature shared by the counterpart nodes among different layer networks in the multiplex network. Firstly, we use a network aggregation method to aggregate all layers into a new single-layer network, where multiple edges are not allowed. Specifically, we set the new network as $G^{\text{new}} = \{V, \varepsilon_{\text{new}}\}$, and for the edge in $\varepsilon_l \in \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_L\}$, we add the edge in ε_{new}. The process is shown in Figure 3. Then, over the obtained new network, we learn the common vector

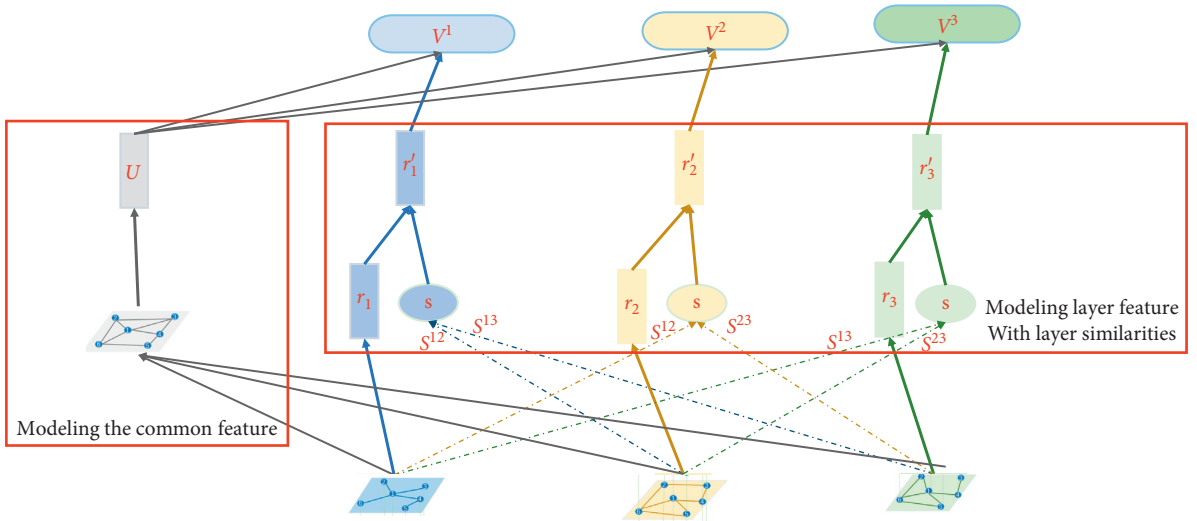

Figure 2: The simple framework of LISCNE.

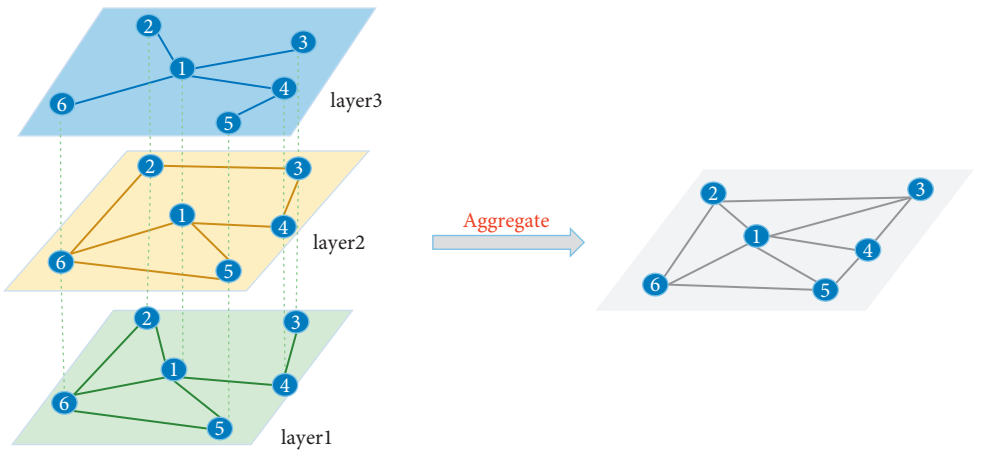

Figure 3: The process of aggregating the multiplex network into a single-layer network.

matrix U for all nodes. We take node v_i as an example; to get the common vector u_i, our goal is to maximize the probability of its neighbors' context in the given walk sequence:

$$\max P\left(v_{i-w}, \ldots, v_{i+w} | v_i; u_i\right), \tag{3}$$

where w is half of the window size and the neighbors of v_i are $v_{i-w}, \ldots v_{i+w}$.

Based on the assumption of conditional independence and using the logarithmic probability, it can be further factorized as

$$L_1 = \sum_{v_i \in V} \sum_{i-c \le j \le i+c, j \ne i} \log P\left(v_j | v_i\right), \tag{4}$$

where $P(v_j|v_t)$ can be defined with a softmax function as

$$P\left(v_j | v_t\right) = \frac{\exp\left(u_j^T u_i\right)}{\sum_{k \in V} \exp\left(u_k^T u_i\right)}, \tag{5}$$

where u_i and u_j are the common vectors for the input node v_i and context v_j, respectively.

4.2. Learning Node Embedding in Each Layer. As discussed before, each layer in a multiplex network has distinct information, and to capture the specific structure for an individual network, we introduce the layer vector that maps single layers into a latent space, i.e., the layer vector r_l for the individual graph G^l. To obtain the overall structure of the multiplex network and layer vectors and learn semantics for each layer simultaneously, we get the node embedding for each layer by concatenating them. For a random node v_i, the embedding in layer G^l can be defined as $V_i^l = u_i \| r^l$.

To preserve local neighborhoods of nodes in each layer, our goal is to maximize the probability of specific neighbors' context in each individual layer:

$$L_2 = \sum_{l} \sum_{v_i \in V} \sum_{v_j \in C^l(v_i)} \log P\left(v_j | v_i; Z_i^l\right), \tag{6}$$

where $C^l(v_i)$ is the context of node v_i in layer G^l and $P(v_j|v_i; V_i^l)$ can be defined as

$$P\left(v_j|v_i;V_i^l\right)=\frac{\exp\left(Z_j^{lT}Z_i^l\right)}{\sum_{k\in V}\exp\left(Z_k^{lT}Z_i^l\right)}, \tag{7}$$

where V_i^l and V_j^l are the node embeddings for the input node v_i and context v_j, respectively.

4.3. Integrating the Similarity between Layer-Networks. The layer vector learned above can capture the distinct structure information within the layer, while in the multiplex network, there is another essential characteristic, which is the similarity between layers varying from layer to layer. Najari et al. [20] testified that incorporating the inter-layer similarities can improve the link prediction performance. Therefore, inspired by their study, we thought of using similarities to enhance embedding capabilities. We added constraints for layer vectors with the similarities between this layer and other layers. Through integrating into the layer similarity, we made the layer vector capture the cross-layer and within-layer information simultaneously.

Firstly, in our model, we used the Global Overlap Rate (GOR) algorithm to measure the similarity among layers in multiple networks. In detail, given two layers α and β in a multiplex network, an overlap edge means that the same node pair simultaneously exist in both networks. The global overlap between layers α and β is denoted by $S^{\alpha\beta}$, which represents the total number of overlapping edges observed in layers α and β. It can be formulated as

$$S^{\alpha\beta}=\frac{\left|\varepsilon_\alpha\cap\varepsilon_\beta\right|}{\left|\varepsilon_\alpha\cup\varepsilon_\beta\right|}, \tag{8}$$

where ε_α is the total number of edges in layer α. The range of $S^{\alpha\beta}$ is in [0,1], and the higher the value, the more the similarity between layers. Particularly, $S^{\alpha\beta}=0$ represents that there are no overlapping edges between layers, indicating that the layers are not related; otherwise, $S^{\alpha\beta}=1$ means that the layers are completely correlated. The similarity $S^{\alpha\beta}$ between layer α and β is the same as similarity $S^{\beta\alpha}$ of layer β and α, and this can also be seen from equation (8).

After illustrating the definition, the next problem we should deal with is how to incorporate it into the model. To address this issue, we assume that if the structures of the two layers are more similar, their representation in the vector space should be closer. We force the following equation to obtain the minimum value:

$$L_3=\sum_{\alpha\beta}\left\|r^\alpha-r^\beta\right\|S^{\alpha\beta}. \tag{9}$$

From equation (9), we can employ stochastic gradient descent to minimize I_3 function as follows:

$$r^l=r^l-\sum_\beta\frac{r^l-r^\beta}{\left\|r^l-r^\beta\right\|_2}S^{l\beta}. \tag{10}$$

4.4. Time Complexity Analysis. Our loss function includes two components. The first part is maximizing the probability-specific neighbors' context in each individual layer to learn the node embedding in each layer, where the main processes of time consumption include getting random walk sequences and skip-gram training, just as the ordinary random walk algorithm. Assuming that the number of nodes is N, the number of edges in each layer is M, the walking length is T, and the number of walking sequences per node is t, the complexity of sampling all sequences is $O(M)+O(N^*T^*t)$. Besides, the complexity of optimization of N^*t sequences with the skip-gram model is $O(N\ \log\ N)$. Therefore, the time complexity of learning the node embedding in each layer is $O(M)+O(N)+O(N\ \log\ N)$. The second part is integrating the similarity between layer-networks. In this part, we exploit the structural similarity between pairs of two layers, and the time complexity is $O(L^*(L-1))$. In real-world network data, the number of layers of L is often very small. The time complexity of this part is relatively insignificant compared to that of the first part of learning node embedding in each layer. So, the overall time complexity of our model is $L^*(O(M)+tOn(N)q+hO_(N\ \log\ N))$.

5. Experiments

In this section, we conduct experiments to validate the proposed LISCNE. To compare our model with some state-of-the-art single-layer embedding methods and multiplex network embedding methods, we perform link prediction and node classification tasks on several datasets with different types of networks.

5.1. Datasets. We employ five real-world multinetwork datasets from three different fields: social, co-authorship, and genetic. The basic statistical information of the datasets is presented in Table 2.

All these datasets are downloaded from the CoMuNe lab's website (https://comunelab.fbk.eu/data.php). The detailed descriptions are as follows:

(i) CKM [51]: by asking the physicians in Illinois, Bloomington, Quincy, and Galesburg three questions, this dataset is classified into three types of relationships. Its ground truth is related to node labels; therefore, we also use this dataset to perform the node classification task.

(ii) PIERRE [52]: this dataset maps layers to different working tasks within the Pierre Auger Collaboration. Based on the keywords and contents of all submissions between 2010 and 2012, the multiplex network is divided into 16 layers.

(iii) ARABIDOPSIS [53, 54]: based on BioGRID, this multiplex network considers genetic interactions of different types of organisms. The multiplex network used in the paper makes use of the following layers: direct interaction, physical association, additive genetic interaction defined by inequality, suppressive genetic interaction defined by inequality, synthetic genetic interaction defined by inequality, association, and colocalization.

Table 2: Statistics of datasets.

Dataset	Network type	Layers	Nodes	Edges	Directed/undirected
CKM	Social	3	246	1,551	Directed
PIERRE	Co-authorship	16	514	7,153	Undirected
ARABIDOPSIS	Genetic	7	6,980	18,654	Directed
MUS	Genetic	7	7,747	19,842	Directed
Arxiv	Co-authorship	13	14,489	59,026	Undirected

(iv) MUS [53, 54]: the dataset is also based on BioGRID. The layers in this dataset are physical association, association, direct interaction, colocalization, additive genetic interaction defined by inequality, synthetic genetic interaction defined by inequality, and suppressive genetic interaction defined by inequality.

(v) Arxiv [52]: choosing papers with "networks" in the title or abstract up to May 2014 in arxiv, the dataset is divided into 13 layers corresponding to different categories with 14,489 nodes.

5.2. Baseline Models. To show the performance of our model, the following six baseline models are implemented for comparison, which can be classified into single-layer network embedding and multiplex network embedding.

(i) DeepWalk [6]: this is a classic single-layer network embedding method, which applies a random walk to get walk sequences and then conducts the skip-gram algorithm on the sequences to train the model.

(ii) Node2Vec [7]: this is also a typical single-layer network embedding model, which utilizes two parameters to take control of the traverse probability in taking the random walk strategies.

(iii) PMNE [39]: this is a multiplex network embedding model that consists of three methods, where network aggregation and result aggregation simply merge all networks or the embedding results of all networks into one, while co-analysis takes the interaction among layers. PMNE_n, PMNE_r, and PMNE_c are used to denote the network aggregation, result aggregation, and co-analysis, respectively.

(iv) MNE [14]: this is a multiplex network embedding model that defines two different dimensional vectors for a node to capture the common information in the whole network and the specific features in a single layer, respectively.

(v) CrossMNA [16]: this is a multiplex network embedding model and also a model for network alignment. It learns simultaneously inter-vector sharing by the anchor nodes in different networks and a network vector for each single layer.

5.3. Experimental Setting. For our model, we set both the common vector dimension and layer vector dimension to 100, and thus after concatenation, the final node embedding vector dimension is 200. For the sake of fairness, we set all the dimensions of final vectors compared with our models as 200. Additionally, for DeepWalk, we set the walk to 20 and the walk length to 80 for each node taking a random walk. For Node2Vec, we empirically set $p = 2$ and $q = 0.5$. For PMME, we follow the default setting in the original paper, which sets α, p, and q to 0.5. For MNE, we set the additional vector dimension to 10 and the common vector dimension to 200. For CrossMNA, according to the original paper, we set the dimension of the inter-layer vector to 200 and the dimension of the network vector to 100.

5.4. Evaluation Metrics. We perform link prediction and node classification tasks to validate the efficiency of our model. For the link prediction task, we execute experiments in each layer and take the average as the final results. Then, we randomly divide datasets into testing sets and training sets. When predicting each positive edge, we also randomly sample unconnected node pairs as a negative edge. We adopt the ROC-AUC evaluation metric to test model performance, that is, the higher the value of AUC is, the better the model performs. For the node classification task, we train all data to get node embeddings of individual layers through our model and baseline models, get the average node embedding of all layers, and then inject the embeddings into a classifier to evaluate the effect. In our experiment, we select a logistic regression classier and choose the $F1$ (weighted) and precision (weighted) as evaluation metrics.

5.5. Performance on Link Prediction. For single-layer methods, we train the node embedding for each layer and use it to predict links in the corresponding layer. For the three methods of PMNE, which take different strategies to aggregate the representations of all layers into one, we take the final node embedding to predict links in all layers. For all models, we average the AUC values of all relation types as final results. In experiments, we take five-fold cross-validation for all datasets.

The results are shown in Table 3, from which we can draw the following observations:

(i) The proposed LISCNE model can stably outperform or achieve comparable performance with all the baseline methods. The results show that merging the layer similarity into models can exactly improve the performance.

(ii) The multiplex network models almost perform better than single-layer models. Meanwhile, these single-layer models in different datasets vary a lot, e.g., in PIERRE dataset, DeepWalk and Node2Vec

Table 3: Results of link prediction on different datasets.

Model	CKM	PIERRE	ARABIDOPSIS	MUS	Arxiv
Node2Vec	0.707	0.572	0.525	0.651	0.753
DeepWalk	0.7	0.589	0.563	0.626	0.759
PMNE_n	0.781	0.8	0.828	0.867	0.872
PMNE_r	0.789	0.65	0.586	0.62	0.782
PMNE_c	0.763	0.516	0.529	0.563	0.599
MNE	0.785	0.791	0.765	0.779	0.834
CrossMNA	0.828	0.733	0.845	0.879	0.922
LISCNE	0.883	0.792	0.825	0.888	0.924

perform poorly. In other words, considering the intersection across layers is essential.

(iii) LISCNE, CrossMNA, and MNE are more effective than PMNE's three methods. PMNE models learn an overall vector for each node by aggregating all layers, while LISCNE, CrossMNA, and MNE all simultaneously define a vector to capture the common information and another vector to capture the distinct information about each specific layer.

5.6. Performance over Common Vector Embedding Dimension. Figure 4 shows the performance of our model as the embedding dimension of the common vector increases. It can be clearly seen from the figure that the larger the dimension, the better the prediction effect. When the dimension reaches 10, the curve tends to stabilize. Here, for the sake of both accuracy and computational complexity, we set the common vector dimension d_1 to 100.

5.7. Performance on Node Classification. In the node classification task, we choose the CKM dataset with reliable node labels to conduct the experiment and take the companies as the classification label. In addition, the ones injected into the classifier are average node vectors for node embeddings in individual layers. For single-layer network methods, we train all nodes in each layer and get the average of node vectors in each layer. For MNE, CrossMNA, and our model, we also get the average of node vectors of intra-vector in individual layers. Then, all the node representations and corresponding node labels in each layer are divided into training and testing datasets to train the classifier. In our experiment, we use a logistic classifier and evaluate the classification performance with the metrics accuracy, precision, and $F1$, respectively, which can be defined as follows:

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ F1 &= \frac{2\text{TP}}{2\text{TP} + \text{FN} + \text{FP}}. \end{aligned} \tag{11}$$

As shown in Figure 5, the results prove the effectiveness of our model, where our model LISCNE can provide the best performance in terms of $F1$ and precision and achieve

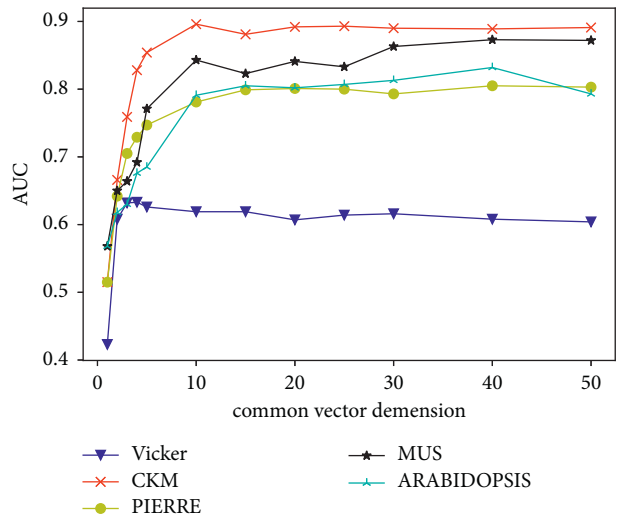

Figure 4: Performance over the dimension d_1 of common vector embedding.

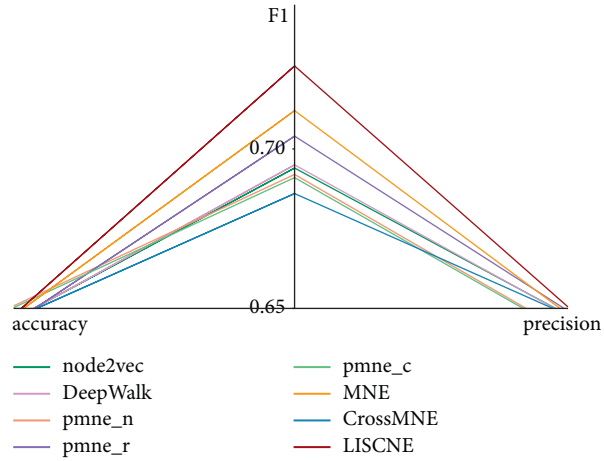

Figure 5: The performance of node classification on CKM.

comparable accuracy with PMNE_n and PMNE_c. However, the effectiveness of multiplex network embedding models like CrossMNA on the link prediction task is not obvious. This may be because every model injected into the classifier is the average of node embedding in all layers. The effect of average is somewhat like aggregation and gets the shared information in all layers.

6. Conclusion and Future Work

In this paper, we propose an effective method called LISCNE for multiplex network embedding. LISCNE defines a common vector for all counterpart nodes in the multiplex network and also introduces a layer vector for each layer. Moreover, when learning layer vectors, it first merges the layer similarities to simultaneously capture intra-layer information and cross-network information. We have performed link prediction and node classification tasks to test LISCNE and conducted extensive experiments

to verify the effectiveness of our proposed model. This model is applicable to aligned networks and certain networks, in which one node in some network is only connected to one node in another network.

Unfortunately, this kind of network cannot cover lots of scenarios in the real world, e.g., the association between a collaboration graph of researchers and a citation graph of papers, where an author can cite papers on multiple topics. In the future, we will extend our model to more manifold networks, for example, one node in some network is connected to several nodes in another network through different weights.

Data Availability

The datasets used to support the results of this study can be available from https://comunelab.fbk.eu/data.php.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (61902278).

References

[1] J. B. Kruskal, *Multidimensional Scaling*, SAGE, no. 11, Thousand Oaks, CA, USA, 1978.

[2] J. B. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[3] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[4] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in Neural Information Processing Systems*, vol. 14, pp. 585–591, 2002.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," http://arxiv.org/abs/1301.3781.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, NY, USA, August 2014.

[7] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, San Francisco, CA, USA, August 2016.

[8] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, San Francisco, CA, USA, August 2016.

[9] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16*, pp. 1145–1152, AAAI Press, Phoenix, AZ, USA, February 2016.

[10] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," http://arxiv.org/abs/1609.02907.

[11] G. Ma, L. He, C. T. Lu, and W. Shao, "Multi-view clustering with graph embedding for connectome analysis," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 127–136, Singapore, November 2017.

[12] J. Li, C. Chen, H. Tong, and H. Liu, "Multi-layered network embedding," in *Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM*, pp. 684–692, San Diego, CA, USA, May 2018.

[13] Y. Sun, N. Bui, T. Y. Hsieh, and V. Honavar, "Multi-view network embedding via graph factorization clustering and co-regularized multi-view agreement," in *Proceedings of the 2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1006–1013, IEEE, Singapore, November 2018.

[14] H. Zhang, L. Qiu, L. Yi, and Y. Song, "Scalable multiplex network embedding," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, vol. 18, pp. 3082–3088, Stockholm, Sweden, July 2018.

[15] M. Ghorbani, M. S. Baghshah, and H. R. Rabiee, "MGCN: semi-supervised classification in multi-layer graphs with graph convolutional networks," in *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 208–211, Vancouver, Canada, August 2019.

[16] X. Chu, X. Fan, D. Yao, Z. Zhu, J. Huang, and J. Bi, "Cross-network embedding for multi-network alignment," in *Proceedings of the World Wide Web Conference*, pp. 273–284, San Francisco, CA, USA, May 2019.

[17] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *Proceedings of the Web Conference 2020*, pp. 3070–3076, Taipei, Taiwan, April 2020.

[18] J. Ni, S. Chang, X. Liu, and W. Cheng, "Co-regularized deep multi-network embedding," in *Proceedings of the 2018 World Wide Web Conference*, pp. 469–478, Lyon, France, April 2018.

[19] B. Du and H. Tong, "Mrmine: multi-resolution multi-network embedding," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 479–488, Beijing, China, November 2019.

[20] S. Najari, M. Salehi, V. Ranjbar, and M. Jalili, "Link prediction in multiplex networks based on interlayer similarity," *Physica A: Statistical Mechanics and its Applications*, vol. 536, Article ID 120978, 2019.

[21] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks," *Science*, vol. 328, no. 5980, pp. 876–878, 2010.

[22] B. Shaw and T. Jebara, "Structure preserving embedding," in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pp. 937–944, Association for Computing Machinery, New York, NY, USA, June 2009.

[23] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in Neural Information Processing Systems*, vol. 14, no. 6, pp. 585–591, 2001.

[24] S. Cao, W. Lu, and Q. Xu, "Grarep: learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and*

Knowledge Management, CIKM 2015, pp. 891–900, ACM, Melbourne, Australia, October 2015.

[25] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, Florence, Italy, May 2015.

[26] K. Mallick, S. Bandyopadhyay, S. Chakraborty, R. Choudhuri, and S. Bose, "Topo2vec: a novel node embedding generation based on network topology for link prediction," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1306–1317, 2019.

[27] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pp. 3111–3119, Lake Tahoe, Nevada, December 2013.

[28] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *Statistics*, vol. 1050, p. 21, 2016.

[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, Long Beach, CA, USA, December 2017.

[30] H. Wang, J. Wang, J. Wang, and M. Zhao, "Graphgan: graph representation learning with generative adversarial nets," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2508–2515, New Orleans, LA, USA, February 2018.

[31] D. C. G. Pedronette and L. J. Latecki, "Rank-based self-training for graph convolutional networks," *Information Processing & Management*, vol. 58, no. 2, Article ID 102443, 2021.

[32] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," http://arxiv.org/abs/1809.10341.

[33] H. Hafidi, M. Ghogho, P. Ciblat, and A. Swami, "Graphcl: contrastive self-supervised learning of graph representations," http://arxiv.org/abs/2007.08025.

[34] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," http://arxiv.org/abs/2006.04131.

[35] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[36] H. Cai, V. W. Zheng, and K. C. C. Chang, "A comprehensive survey of graph embedding: problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[37] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: a survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[38] D. Jin, Z. Yu, P. Jiao, S. Pan, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," http://arxiv.org/abs/2101.01669 CoRR abs/2101.01669.

[39] W. Liu, P. Y. Chen, S. Yeung, T. Suzumura, and L. Chen, "Principled multilayer network embedding," in *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 134–141, IEEE, New Orleans, LA, USA, November 2017.

[40] M. Zitnik and J. Leskovec, "Predicting multicellular function through multi-layer tissue networks," *Bioinformatics*, vol. 33, no. 14, pp. i190–i198, 2017.

[41] A. Bagavathi and S. Krishnan, "Multi-net: a scalable multiplex network embedding framework," in *Proceedings of the International Conference on Complex Networks and their Applications*, pp. 119–131, Springer, Basel, Switzerland, December 2018.

[42] C. Park, C. Yang, Q. Zhu, D. Kim, H. Yu, and J. Han, "Unsupervised differentiable multi-aspect network embedding," http://arxiv.org/abs/2006.04239.

[43] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang, "Representation learning for attributed multiplex heterogeneous network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19*, pp. 1358–1368, Association for Computing Machinery, New York, NY, USA, July 2019.

[44] M. Gong, W. Liu, Y. Xie, Z. Tang, and M. Xu, "Heuristic 3D interactive walk for multilayer network embedding," *IEEE Transactions on Knowledge and Data Engineering*, p. 1, 2020.

[45] Q. Meng, T. Jian, J. Shang, R. Xiang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*, pp. 1767–1776, Singapore, November 2017.

[46] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang, "Multi-dimensional graph convolutional networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 657–665, SIAM, Champaign, IL, USA, May 2019.

[47] R. Wang, L. Li, X. Tao, X. Dong, P. Wang, and P. Liu, "Trio-based collaborative multi-view graph clustering with multiple constraints," *Information Processing & Management*, vol. 58, no. 3, Article ID 102466, 2021.

[48] C. Park, J. Han, and H. Yu, "Deep multiplex graph infomax: attentive multiplex network embedding using global information," *Knowledge-Based Systems*, vol. 197, Article ID 105861, 2020.

[49] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," ICLR, New Orleans, LA, USA, 2019.

[50] Y. Sun, S. Wang, T. Y. Hsieh, X. Tang, and V. Honavar, "Megan: a generative adversarial network for multi-view network embedding," http://arxiv.org/abs/1909.01084.

[51] J. Coleman, E. Katz, and H. Menzel, "The diffusion of an innovation among physicians," *Sociometry*, vol. 20, no. 4, pp. 253–270, 1957.

[52] D. Manlio, L. Andrea, A. Alex, and R. Martin, "Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems," *Physical Review X*, vol. 5, no. 1, p. 11027, 2015.

[53] C. Stark, "Biogrid: a general repository for interaction datasets," *Nucleic Acids Research*, vol. 34, no. 90001, pp. 535–539, 2006.

[54] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, "Structural reducibility of multilayer networks," *Nature Communications*, vol. 6, no. 1, p. 6864, 2015.