Expert Systems With Applications 230 (2023) 120552

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A graph-enhanced attention model for community detection in multiplex networks☆

Bang Wang [a,b], Xiang Cai [b], Minghua Xu [c], Wei Xiang [b,*]

[a] Hubei Key Laboratory of Smart Internet Technology, Huazhong University of Science and Technology (HUST), Wuhan, China
[b] School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China
[c] School of Journalism and Information Communication, Huazhong University of Science and Technology (HUST), Wuhan, China

ARTICLE INFO

Keywords:
Community detection
Multiplex network
Graph contrastive learning
Self-attention
Modularity

ABSTRACT

Community detection is to divide a graph or a network into several components with high intra-edge density in a single-layer network. However, a multiplex network, consisting of multiple layers which share the same nodes but each owning different types of edges, brings new challenges for community detection. Although many classic algorithms and several multiplex embedding-based models try to address this problem, i.e. community detection in multiplex networks, they mostly focus on the local topological structures but ignoring the global information in each network layer, which is important for encoding intra-layer structural attributes and merging inter-layer semantic relations. In this paper, we propose a *graph-enhanced attention model* (GEAM) for community detection in multiplex networks by utilizing the above global information. In particular, the GEAM first constructs a layer contrastive learning module to encode node and graph embedding from the local and global graph view in each network layer. We also propose a self-attention adaptive fusion mechanism to learn a comprehensive version of node representation by fusing multiple layers. Finally, we propose an edge density-driven community detection module to train our GEAM in an end-to-end manner and output community divisions with strong modular structures. Experiments on both synthetic and real-world datasets validate the superiority of our GEAM over the state-of-the-art algorithms in terms of better community detection performance.

1. Introduction

1.1. Background

Community detection is to divide a graph into several components each with high intra-edge density, which is one of the most important tasks in many real-world applications such as crime organization mining in social network analysis (Magalingam et al., 2015). For example, community detection can mine latent criminal organizations through their different types of communications, like facebook/twitter relations, phone calls and offline geolocations (Magalingam et al., 2015).

In recent years, lots of community detection algorithms have been proposed, which can be generally divided into two categories, i.e., the traditional heuristic methods and the deep learning-based methods. Those heuristic methods (Li, Chen et al., 2022; Li, Lei et al., 2022) are usually based on objective optimization. For example, Li, Chen et al. (2022) proposes a stable community detection algorithm by using peak clustering and label propagation technique. Those deep learning-based methods (Al-Andoli et al., 2022a, 2022b) learn the node representation by training a neural network model. For example, Al-Andoli et al. (2022a) design a parallel deep learning model to solve the local minima and slow convergence problem during community detection.

However, most of the research on community detection is applied in single-layer networks, which consist of the node set and the unique type of edges. When there is more than one kind of relationship between nodes, it is no longer suitable to use single-layer networks for problem modeling. Instead, a multiplex network is a suitable modeling technique, where nodes are connected with different types of edges, each of which describes a particular relation. The task of community detection in multiplex networks can be defined as follows.

Assume that a multiplex network consists of L network layers, represented as (V, E_l), where $l = 1, 2, \ldots, L$. All layers share the same node set, but different layers are with different edges. Community detection

☆ This work is supported by Major Project of the National Social Science Foundation in China (AI and Precise International Communication, Grant No: 22&ZD317).

* Corresponding author.
E-mail addresses: wangbang@hust.edu.cn (B. Wang), cxiang@hust.edu.cn (X. Cai), xuminghua@hust.edu.cn (M. Xu), xiangwei@hust.edu.cn (W. Xiang).

https://doi.org/10.1016/j.eswa.2023.120552
Received 20 September 2022; Received in revised form 24 May 2023; Accepted 25 May 2023
Available online 1 June 2023
0957-4174/© 2023 Elsevier Ltd. All rights reserved.

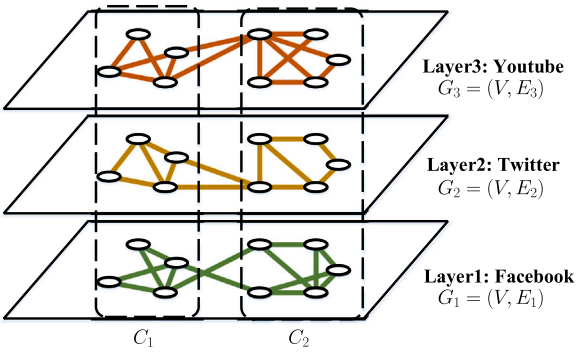


Fig. 1. Illustration of a three-layer toy multiplex network, where each layer corresponds to users' interactions on Facebook, Twitter, and YouTube respectively. It can be divided into two communities according to the nodes' relations.

in multiplex networks is to partition the node set V ($|V| = N$) into M communities, i.e. $\{C_1, C_2, \ldots, C_M\}$. Fig. 1 illustrates a three-layer multiplex network with two communities.

1.2. Motivation

Community detection in multiplex networks is more challenging, and two main challenges are as follows. (1) Most traditional algorithms are the simple extension of those single-layer methods. Since multilayer modularity (Mucha et al., 2010) was proposed, some algorithms have designed for community detection in multiplex networks (Berlingerio et al., 2011; Boutemine & Bouguessa, 2017; Pramanik et al., 2017). A common idea of these algorithms is to extend traditional single-layer algorithms to multiplex networks. However, the extensions are too native to achieve good performance. For example, Berlingerio et al. (2011) first flatten a multiplex network into a single-layer one by judging whether the edges exist in at least one layer and then detect communities from this flattened network. This method ignores layer difference and may lead to the loss of intra-layer structural information. (2) Some node embedding algorithms lack the specific research on community detection or focus only on the local structures. In recent three years, some multiplex embedding algorithms have tried to produce a generalized node representation for multiple downstream tasks, one of which is community detection. However, these algorithms have some weakness. For example, some contrastive learning models, i.e. DMGI (Park et al., 2020) and HDMI (Jing et al., 2021), lack the community-specific characteristics and need extra node attributes as input. Other node embedding algorithms such as M-DeepWalk (Song & Thiagarajan, 2019) only focus on the local topology but ignores the latent global information when treating all network layers from an integrated systematic viewpoint.

We consider that the global information is an important factor for the task of community detection in multiplex networks. Firstly, the global information can facilitate the interaction between any two nodes, even if they are topologically far apart. In one single-layer, we hope that a node can communicate with the other nodes which may be topologically far away. These distant nodes are considered as "latent neighbors" and the node with lots of "latent neighbors" is regarded as an active node at the current layer. Secondly, the global information can provide guidance for deciding the importance of each layer during multilayer fusion. We think that the active nodes are more important compared to the same node in other layers which is inactive and have more impacts on merging multiple layers in a multiplex network.

We further argue that the global information can not only encode node embedding with structural characteristics in each single-layer, but also help to determine the importance of multiple layers during the layer merging process. We focus on two ways to mine the latent global information, i.e., graph diffusion and graph pooling. The former uses densely weighted edges to reconstruct the whole graph and the diffusion graph can be seen as a global view for contrastive learning in each network layer. The latter generates a global perspective of one network layer for comparison with the other layers. Motivated from the aforementioned analysis, we propose a *Graph-Enhanced Attention Model* (GEAM) for community detection in multiplex networks.

1.3. Contribution

The GEAM is proposed by encoding intra-layer topological information and merging inter-layer latent semantic information. There are three key components in GEAM. (1) How to learn local and global intra-layer topological information from each layer. In GEAM, we design a *layer contrastive learning* module to encode the origin graph-based local topology and the diffusion graph-based global topology, and generate the node encoding and graph encoding for each network layer. (2) How to better fuse the inter-layer latent semantics for community detection. In GEAM, we develop a new self-attention adaptive fusion mechanism to merge node encoding from multiple layers and learn a holistic version of node representation. (3) How to detect communities with high modular structures. In GEAM, we use an edge density-driven community detection module to train our GEAM and output community divisions with cohesive structures.

The main contributions of our work are summarized as follows:

- Propose a graph-enhanced attention model for community detection in multiplex networks by encoding intra-layer topologies and merging inter-layer semantics.
- Design a layer contrastive learning module to encode local and global information and generate node and graph encoding for each network layer.
- Develop a self-attention adaptive fusion mechanism to acquire a comprehensive node representation by fusing multiple layers' encodings.
- Introduce an edge density-driven community detection module for detecting communities with strong modular structures.
- Experiments on both synthetic and real-world datasets validate the effectiveness of our proposed model.

The rest of the paper is organized as follows. We provide a brief review of the most relevant works in Section 2. Then we introduce the proposed GEAM in Section 3. Experiments are presented in Sections 4 and 5. Section 6 concludes this paper.

2. Related work

In this section, we briefly review the related works, including multiplex network community detection, graph contrastive learning, as well as attention mechanisms.

2.1. Multiplex network community detection

In general, algorithms for community detection in multiplex networks can be categorized into three types depending on the way of handling multiplex networks, i.e. flattening, aggregation, and direct methods (Huang et al., 2021).

Flattening methods usually turn a multiplex network into a single-layer one and then apply community detection algorithms on this flattened network (Berlingerio et al., 2011; Gao et al., 2019). There are two ways of flattening, i.e., reduction-based flattening and extension-based flattening. For example, Berlingerio et al. (2011) is a reduction-based flattening method and the flattened graph keeps the same size as the origin network layer. While the extension-based flattening methods try to extend a multiplex network to a supra-graph, which contains the raw intra-layer edges and the generated inter-layer edges. For example, Gao et al. (2019) construct a extended adjacency matrix and use particle competition model to detect communities.

Aggregation methods first extract community structural features from each layer and then refine the consensus information for community detection (Ali et al., 2019; Li et al., 2023; Tagarelli et al., 2017). For example, Tagarelli et al. (2017) build a co-association matrix to find connected components in each network layer, and then extract consensus community structures from them.

Direct methods detect communities directly from multiplex networks by optimizing some quality-assessment metrics without flattening or aggregation (Amini et al., 2022; Pramanik et al., 2017; Shao et al., 2022). For example, Pramanik et al. (2017) modify multilayer modularity (Mucha et al., 2010) and use the improved GN (Newman & Girvan, 2004) and Louvain (Blondel et al., 2008) algorithm for community detection.

2.2. Graph contrastive learning

Graph contrastive learning is one of the most popular ways in graph representation learning, which aims to learns a discriminator for node representation or graph representation by contrasting positive and negative samples. According to different contrastive objectives, we divide graph contrastive learning algorithms into two main categories, i.e. local–local and local–global contrasting methods.

Local–local contrasting methods choose positive and negative samples from node or edges (Xia et al., 2022; Zhu et al., 2021). For example, SimGRACE (Xia et al., 2022) designs a simple framework for graph contrastive learning, which takes the origin graph as input and GNN model with its variants as two encoders for local–local contrast. Another example is GCA (Zhu et al., 2021) which contrasts node representation from two graph augmentation views and takes a adaptive strategy to choose topological nodes and attributes as samples.

Local–global contrasting methods choose samples from nodes and graphs (Hassani & Khasahmadi, 2020; Velickovic et al., 2019; Zhou et al., 2022). For example, DGI (Velickovic et al., 2019) uses a injective read function to generate a graph-level representation and maximizes the mutual information between the node representation in a patch graph and the corresponding graph representation. MVGRL (Hassani & Khasahmadi, 2020) provides multiple graph views for contrastive learning. The MVDGI (Zhou et al., 2022) maximizes the mutual information between the node representation and the unified graph representation for contrast learning.

2.3. Attention mechanisms

Attention mechanisms can help a neural network to merge the multiple inputs by unequally treating each input component according to its importance, which have reached pretty good performance in many NLP tasks such as event extraction (Dai et al., 2022) and text classification (Yang et al., 2022). For example, the AutoDefect (Yang et al., 2022) uses a channel attention mechanism to extract robust feature for defecting text classification system.

As one kind of the attention mechanisms, self-attention shows powerful capabilities on encoding semantics and learning dependencies between each input (Du et al., 2023; Wang et al., 2022). For example, Wang et al. (2022) propose a novel multi-view self-attention mechanism for speaker recognition. Du et al. (2023) propose a novel method call SAITS, which utilizes the self-attention technique to impute the missing value in multivariate time series. We compare a node in one network layer to a word in NLP and use attention mechanisms to merge node encoding from different layers.

3. Graph-enhanced attention model

In this section, we introduce the proposed GEAM. Fig. 2 presents the architecture of GEAM, which contains the following modules: layer contrastive learning, self-attention adaptive fusion and edge density-driven community detection.

3.1. Layer contrastive learning

Layer contrastive learning is designed to learn a generalized local–global node encoding in each network layer. From the perspective of topology, we apply a graph augmentation mechanism by calculating a diffusion graph matrix. We denote the diffusion graph as a global view and the origin graph as a local view. From the perspective of feature, we generate the node initial embedding by applying DeepWalk (Perozzi et al., 2014) and corrupt it to acquire negative samples. Besides, we use a view-specific encoder to encode neighbor-aware topological structures and a graph pooling function to aggregate the graph-level information. At last, we use a discriminator to contrast the node encoding and graph encoding from the two views in an interwoven fashion and maximize the local–global mutual information in each single network layer.

3.1.1. Graph augmentation

In view of the importance of global information, we hope that a node can learn its encoding from as many nodes as possible, which may be topologically far apart. In graph neural networks, nodes exchange features with their low-order neighbors. Though we can extract the distant features by increasing the depth of GNNs, it may cause the over-smoothing problem. Besides, the input graph uses a binary value to indicate whether an edge exists, which cannot represent the extent of importance between nodes. Given all of that, we try to build another view of the origin graph by applying the graph diffusion technique to help message pass through the high-order neighbors in a denoising manner, that is, to balance the influence of the global information and the local information. The graph diffusion matrix $\mathbf{S}$ is defined by

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k T^k \in \mathbb{R}^{N\times N}, \tag{1}$$

where θ_k is the weighting coefficient of the kth transition to control the ratio of local–global information. $\mathbf{T} \in \mathbb{R}^{N\times N}$ is the generalized transition matrix, where N is the number of nodes. Specifically, there are two main graph diffusion functions, i.e. Personalized PageRank (PPR) and Heat Kernel (HK). The generalized transition matrix of the two functions is the same as $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$, where $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix and $\mathbf{D} \in \mathbb{R}^{N\times N}$ is the degree matrix. The difference is that $\theta_k^{PPR} = \alpha(1-\alpha)^k$ and $\theta_k^{HK} = e^{-t}t^k/k!$, where α is teleport probability in a random walk and t is diffusion time. The stationary solutions of PPR and HK are as follows, respectively.

$$\mathbf{S}_l^{PPR} = \alpha(\mathbf{I}_N - (1-\alpha)\mathbf{D}_l^{-1/2}\mathbf{A}_l\mathbf{D}_l^{-1/2}), \tag{2}$$

$$\mathbf{S}_l^{HK} = e^{\mathbf{A}_l\mathbf{D}_l^{-1}-t}. \tag{3}$$

3.1.2. Feature generation and corruption

The multiplex network only consists of nodes and edges in our problem setting. Therefore, some graph embedding techniques are applied to obtain node initial embedding in each network layer. More specifically, we choose DeepWalk (Perozzi et al., 2014) to capture path-aware topological features by its characteristics of random walk and sampling node sequence. Given a multiplex network, we denote $\mathbf{X}_l \in \mathbb{R}^{N\times d_0}$ as the node initial embedding of the lth network layer, where d_0 is the dimension of initial embedding. Note that other graph embedding methods are also used and we will compare them in our experiments. Although the node initial embedding can be also directly used for community detection to some extent, we note that it is not a proper way. Firstly, the initial embedding only contains local topology structures and each node is difficult to communicate with its distant nodes. Secondly, we cannot acquire useful knowledge from node initial embedding to judge which network layer is more important. Therefore, we input the initial embedding into the subsequent module for the following encoding and training.

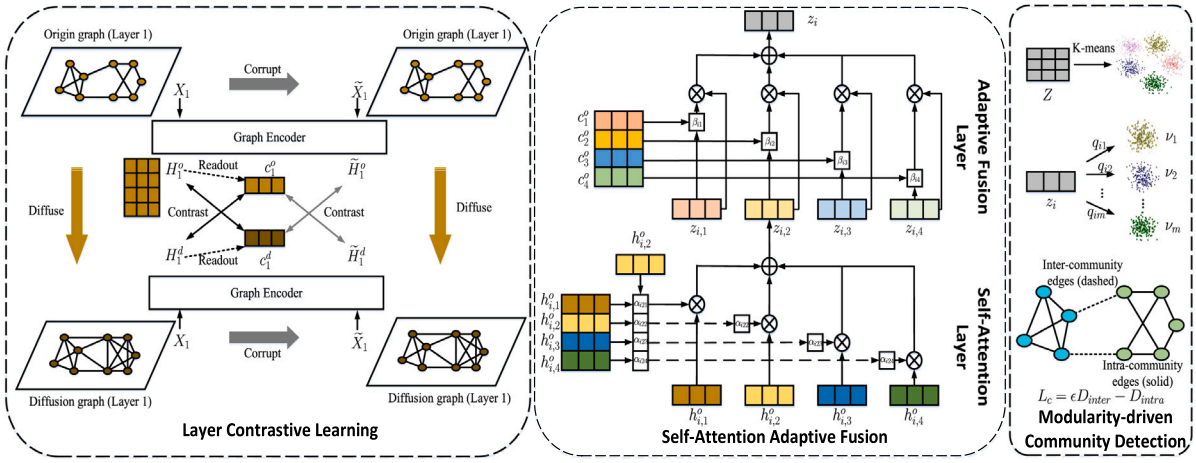


Fig. 2. The overall architecture of the GEAM. The left is layer contrastive learning module for each network layer; The middle is self-attention adaptive fusion mechanism for merging node encoding; The right is edge density-driven community detection module for training our model.

Although graph neural network models (Bo et al., 2020) usually use the attributes of nodes as the initial features, some classic algorithms for community detection are almost applied in non-attribute multiplex networks. We note that using such attributes could improve the performance of community detection. However, it may introduce extra information for unfair comparisons. In order to ensure fair and valid comparisons, we choose the multiplex network without node attributes and generate the initial features from the topological structures of the multiplex network.

In order to acquire negative samples, we attempt to change the distribution of node initial embedding, i.e. X_l, to mislead the feature expression by using a corruption function $C(\cdot)$. Generally speaking, corruption functions are usually designed from either the perspective of feature or topology. In view that the graph diffusion has modified the topological structures, we consider a random feature permutation for the initial embedding of each layer, i.e., exchanging the node embedding of any two nodes.

$$\widetilde{\mathbf{X}}_l = C(\mathbf{X}_l) \in \mathbb{R}^{N \times d_0}. \tag{4}$$

3.1.3. Encoders

Adjacency matrix and diffusion matrix can be seen as a original graph view (denoted as o) and a diffusion graph view (denoted as d), which contain the local and global topological structures separately. In order to extract these topological features effectively, we prefer to use a basic graph encoder to encode node embedding. More concretely, We apply an independent two-layer Graph Convolutional Network (GCN) for each view, which is the so-called view-specific encoder. The encoding process takes the following form:

$$\mathbf{H}_l^o = ReLU(\widehat{\mathbf{A}}_l ReLU(\widehat{\mathbf{A}}_l \mathbf{X}_l \mathbf{W}_{l,1}^o)\mathbf{W}_{l,2}^o), \tag{5}$$

$$\mathbf{H}_l^d = ReLU(\mathbf{S}_l ReLU(\mathbf{S}_l \mathbf{X}_l \mathbf{W}_{l,1}^d)\mathbf{W}_{l,2}^d), \tag{6}$$

where $\widehat{\mathbf{A}}_l = \widetilde{\mathbf{D}}_l^{-1/2}\widetilde{\mathbf{A}}_l\widetilde{\mathbf{D}}_l^{-1/2}$ is symmetrically normalized adjacency matrix, $\widetilde{\mathbf{D}}_l \in \mathbb{R}^{N \times N}$ is the degree matrix of $\widetilde{\mathbf{A}}_l = \mathbf{A}_l + \mathbf{I}_N$. $\mathbf{S}_l$ is diffusion matrix. $\mathbf{W}_{l,1}^o$, $\mathbf{W}_{l,2}^o$, $\mathbf{W}_{l,1}^d$, and $\mathbf{W}_{l,2}^d$ are learnable weight matrices. For corruption node embedding, we input it to into the same graph encoder of each view to generate node corruption encoding $\widetilde{\mathbf{H}}_l^o$ and $\widetilde{\mathbf{H}}_l^d$ for contrastive learning.

3.1.4. Contrastive learning

In order to make sure that the node encoding H_l^o can learn latent global topological information from view d, we use the deep InfoMax (Hjelm et al., 2018) method to maximize the mutual information between view o and view d by contrasting node encoding in one view and graph encoding in the other view. Firstly, we input the node encoding into a graph pooling module, i.e. a readout function, to generate the graph encoding. The pooling process is calculated as:

$$\mathbf{c}_l^o = Readout(\mathbf{H}_l^o) = PReLU(\frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_{i,l}^o), \tag{7}$$

where $\mathbf{c}_l^o$ is the graph encoding of the origin graph. $\mathbf{h}_{i,l}^o$ is the encoding of node i of view o. $\mathbf{c}_l^d$ is calculated as the same way. Note that not only the graph encoding can be used to build the positive and negative samples, but also it can provide the importance of current layer in multiplex network for the following fusion module.

Then we choose the positive samples from node encoding $\mathbf{h}_{i,l}^o$ and graph encoding $\mathbf{c}_l^d$, and vice versa, node encoding $\mathbf{h}_{i,l}^d$ and graph encoding $\mathbf{c}_l^o$. The negative samples are chosen from the corruption node encoding and graph encoding. In total, we optimize the objective function as:

$$max\frac{1}{2N}\sum_{i=1}^{N}(MI(\mathbf{h}_{i,l}^o, \mathbf{c}_l^d) + MI(\mathbf{h}_{i,l}^d, \mathbf{c}_l^o)), \tag{8}$$

where $MI(\mathbf{h}_{i,l}^o, \mathbf{c}_l^d)$ is the mutual information between node encoding $\mathbf{h}_{i,l}^o$ and graph encoding $\mathbf{c}_l^d$, which is implemented by a discriminator to achieve 1 for the positive samples and 0 for the negative. Therefore, we hope to maximize the agreement of the positive samples and simultaneously minimize the agreement of the negative samples. More specifically, we compute the cross entropy loss for the positive and negative samples in the lth layer:

$$\begin{aligned} L_{r,l} = &\frac{1}{2N}\sum_{i=1}^{N}[log\mathcal{D}(\mathbf{h}_{i,l}^o, \mathbf{c}_l^d) + log(1 - \mathcal{D}(\widetilde{\mathbf{h}}_{i,l}^o, \mathbf{c}_l^d))] + \\ &\frac{1}{2N}\sum_{i=1}^{N}[log\mathcal{D}(\mathbf{h}_{i,l}^d, \mathbf{c}_l^o) + log(1 - \mathcal{D}(\widetilde{\mathbf{h}}_{i,l}^d, \mathbf{c}_l^o))], \end{aligned} \tag{9}$$

where $\mathcal{D}(\mathbf{h}_{i,l}^o, \mathbf{c}_l^d)$ is a discriminator that measures the agreement between node encoding $\mathbf{h}_{i,l}^o$ of view o and graph encoding $\mathbf{c}_l^d$ of view d, which is computed by:

$$\mathcal{D}(\mathbf{h}_{i,l}^o, \mathbf{c}_l^d) = Sigmod((\mathbf{h}_{i,l}^o)^T P_l \mathbf{c}_l^d), \tag{10}$$

where P_l is learnable parameter. For multiple network layers, we optimize the sum of contrastive learning loss:

$$L_r = \sum_{l=1}^{L} L_{r,l}. \tag{11}$$

3.2. Self-attention adaptive fusion

Layer contrastive learning module provides two kinds of useful information, i.e. the node encoding H_l^o which learns from the global diffusion graph in the lth network layer and the graph encoding $\mathbf{c}_l^o$ which measures the importance of the current layer in a multiplex network. In order to determine the contribution of each layer for a node and merge them to obtain a comprehensive node representation, we consider an self-attention adaptive fusion mechanism, which contains a self-attention layer and an adaptive fusion layer.

3.2.1. Self-attention layer

We apply a self-attention layer to learn the layer-aware representation for a node from other network layers. We first calculate the alignment scores for each node pair as follows:

$$f^{sa}(\mathbf{h}_{i,s}, \mathbf{h}_{i,r}) = tanh(\mathbf{W}^{sa}\mathbf{h}_{i,s} + \mathbf{U}^{sa}\mathbf{h}_{i,r} + \mathbf{b}^{sa}), \tag{12}$$

where $\mathbf{h}_{i,s}$ and $\mathbf{h}_{i,r}$ correspond to node i in layer s and layer l, separately. $\mathbf{W}^{sa}$, $\mathbf{U}^{sa}$, and $\mathbf{b}^{sa}$ are learnable parameters. We normalize it by a softmax function:

$$\boldsymbol{\alpha}_{isr} = \frac{exp(f^{sa}(\mathbf{h}_{i,s}, \mathbf{h}_{i,r}))}{\sum_{r'=1}^{L} exp(f^{sa}(\mathbf{h}_{i,s}, \mathbf{h}_{i,r'}))}, \tag{13}$$

where $\boldsymbol{\alpha}_{isr}$ is the self-attention wight for node i between layer s and between r. Afterwards, for the lth layer, the layer-aware node representation $z_{i,l}$ is calculated as follows:

$$\mathbf{z}_{i,l} = \sum_{r=1}^{L} \boldsymbol{\alpha}_{ilr}\mathbf{h}_{i,l}, \tag{14}$$

3.2.2. Adaptive fusion layer

We next design a fusion mechanism to compress the node representation in all network layers to obtain a comprehensive node representation $\mathbf{Z}$ under the guidance of the graph encoding. Specifically, for the lth layer, we feed node representation $\mathbf{z}_{i,l}$ and graph encoding $\mathbf{c}_l$ into a MLP-based attention layer to calculate an adaptive alignment score:

$$f^{af}(\mathbf{z}_{i,l}, \mathbf{c}_l) = \mathbf{c}_l^T tanh(\mathbf{W}^{af}\mathbf{z}_{i,l} + \mathbf{b}^{af}), \tag{15}$$

where $\mathbf{W}^{af}$ and $\mathbf{b}_{af}$ are learnable parameters. We note that a larger $f^{af}(\mathbf{z}_{i,l}, \mathbf{c}_l)$ means that the node representation $\mathbf{z}_{i,l}$ are more important in layer l and corresponds to a higher weight in the following multilayer fusion, computed by:

$$\boldsymbol{\beta}_{il} = \frac{exp(f^{af}(\mathbf{z}_{i,l}, \mathbf{c}_l))}{\sum_{l'=1}^{L} exp(f^{af}(\mathbf{z}_{i,l}, \mathbf{c}_l))}, \tag{16}$$

$$\mathbf{z}_i = \sum_{l=1}^{L} \boldsymbol{\beta}_{il}\mathbf{z}_{i,l}, \tag{17}$$

where $\mathbf{z}_i$ is the ith row of $\mathbf{Z}$ and $\boldsymbol{\beta}_{il}$ is the adaptive wight of node i in layer l.

3.3. Edge density-driven community detection

To detect communities with strong modular structures, we design an edge density-driven community detection module for training our model in an end-to-end manner. Note that the modularity mentioned refers to multilayer modularity (Mucha et al., 2010) without additional explanation. It is an unsupervised metric which usually measures the structural strength of communities in a multiplex network. Modularity is computed as follows:

$$\mathbf{Q} = \frac{1}{2\eta}\sum_{ijsr}[(\mathbf{A}_{ijs} - \gamma_s\frac{d_{is}d_{jr}}{2m_s})\delta(s,r)\delta\left(g_{is}, g_{jr}\right) + \delta(i,j)\mathscr{C}_{jsr}\delta\left(g_{is}, g_{jr}\right)], \tag{18}$$

where i and j index nodes, s and r index layers. γ_s is a resolution parameter in the sth layer. $\mathscr{C}_{jsr}$ denotes the inter-layer edge strength, which is called the coupling parameter. η is the total multilayer edge strength, defined as $\eta = \left(\sum_{ijs}\mathbf{A}_{ijs} + \sum_{jsr}\mathscr{C}_{jsr}\right)/2$. m_s is the total intra-layer edge strength of the sth layer, defined as $m_s = \left(\sum_{ij}\mathbf{A}_{ijs}\right)/2$. d_{is} is the degree of nodes i of the sth layer. g_{is} is the community label of node i in the sth layer. We assigns a node with the same community label in all layers, i.e. $\delta\left(g_{is}, g_{ir}\right) = 1$. Therefore, the second term of Eq. (18) is a constant in our problem setting and the first term mainly measures the intra-community edge density. We hope that a community consists of dense intra-community edges and loose inter-community edges. Therefore, we modify Dense Community Aggregation (Li, Jing et al., 2022) for multiplex networks.

For node representation $\mathbf{z}_i$, we use the Student's t-distribution (Van der Maaten & Hinton, 2008) as a kernel to measure the similarity between each node and each community center and obtain the soft community assignment:

$$\mathbf{q}_{im} = \frac{(1 + \|\mathbf{z}_i - \boldsymbol{\nu}_m\|^2)^{-1}}{\sum_{m'}(1 + \|\mathbf{z}_i - \boldsymbol{\nu}_{m'}\|^2)^{-1}}, \tag{19}$$

where $\mathbf{q}_{im}$ measures the probability of node i assigned to community m and $\sum_m \mathbf{q}_{im} = 1$. $\boldsymbol{\nu}_m$ is the representation of the mth community center, which is initialized by using the K-means algorithm.

Then we compute two kinds of expectation, i.e. intra-community and inter-community edge density, calculated as:

$$D_{intra} = \frac{1}{LN}\sum_{l}\sum_{ij}\sum_{m}[\mathbf{A}_{ijl} - d_l(m)]\mathbf{q}_{im}\mathbf{q}_{jm}, \tag{20}$$

$$D_{inter} = \frac{1}{LN(N-1)}\sum_{l}\sum_{ij}\sum_{m_1 \neq m_2}\mathbf{A}_{ijl}\mathbf{q}_{im_1}\mathbf{q}_{jm_2}, \tag{21}$$

where $d_l(m)$ is the ratio of real edges for community m in layer l. Given community m with $|E_m|$ edges and $|C_m|$ nodes, $d(m)$ is computed as $d(m) = \frac{|E_m|}{|C_m|(|C_m|-1)}$.

Our objective function is to maximize the intra-community density and minimize inter-community density simultaneously. We calculate the community detection loss as:

$$L_c = \epsilon D_{inter} - D_{intra}, \tag{22}$$

where ϵ is the coefficient to balance two kinds of density.

The GEAM is training in an end-to-end manner. We define the total loss of GEAM as:

$$L = L_r + \lambda L_c, \tag{23}$$

where λ is the coefficient to balance two losses of contrastive learning and community detection.

Finally, the community label of each node is obtained from the optimized distribution $\mathbf{q}$:

$$g_i = \arg\max_m(\mathbf{q}_{im}). \tag{24}$$

3.4. Complexity analysis

For a multiplex network with N nodes and L layers, the complexity of DeepWalk is $\mathcal{O}(L\rho Nt\omega(d_0 + d_0 logN))$, where ρ is the number of random walks, t is the walk length, ω is the window size, and d_0 is initial embedding size. The complexity of layer contrastive learning module is $\mathcal{O}(\sum_{l=1}^{L}|E_l|d_0d_1d_2)$, where $|E_l|, l = 1, 2, \ldots, L$ is the number of edges in each layer, d_1 and d_2 are node encoding dimension. The complexity of self-attention adaptive fusion module is $\mathcal{O}(N^2d_2)$ for self-attention layer and $\mathcal{O}(Nd_2)$ for adaptive fusion layer. For edge density-driven community detection module, the complexity is $\mathcal{O}(NM + NlogN)$, where M is the number of detected communities. Therefore, the total time complexity of GEAM is $\mathcal{O}(L\rho Nt\omega(d_0 + d_0 logN) + \sum_{l=1}^{L}|E_l|d_0d_1d_2 + N^2d_2 + Nd_2 + NM + NlogN)$.

4. Experiment setup

4.1. Datasets

Experiments are conducted on both synthetic datasets and the real-world datasets. Synthetic datasets mainly refer to **mLFR** (Bródka, 2016) which is a benchmark tool for building multiplex networks. For the real-world datasets, we select several common multiplex networks, including **AUCs** (Magnani et al., 2013), **RM** (Eagle & Pentland, 2006), **C.elegans** (Chen et al., 2006), **London** (De Domenico et al., 2014), **Vickers** (Zhang et al., 2017), **FFTWYT** (Dickison et al., 2016), **CKM** (Coleman et al., 1957), **Plasmodium** (Stark et al., 2006), **Lazega** (Grossetti & Lazega, 2003), and **Pierre** (Domenico et al., 2014) to conduct experiments. These real-world datasets are all collected manually and without ground truth labels.

• **mLFR** (Bródka, 2016): There are one main parameters we will vary in the mLFR datasets, i.e. the *mixing parameter* μ which is the probability of edges connecting to the nodes of the other communities. A larger μ means less obvious community structures. Table 1 presents some parameter settings for the mLFR datasets in our experiment.

For the real-world multiplex networks, we choose them from the society, transportation, biology and other fields. Table 2 presents their basic statistics.

• **AUCs** (Magnani et al., 2013): It consists of five kinds of online and offline relationships, i.e. Facebook, leisure, work, co-authorship, and lunch between the employees in the Aarhus University.

• **RM** (Eagle & Pentland, 2006): It records the three social relationships including friendship, average proximity at work and average proximity outside lab between students in the MIT Media Laboratory and incoming students in the MIT Sloan business school.

• **C.elegans** (Chen et al., 2006): It is the "Caenorhabditis elegans" connectome with three layers corresponding to different synaptic junctions, i.e. electric connection, chemical monadic connection and chemical polyadic connection.

• **London** (De Domenico et al., 2014): It describes the existing routes between the train stations in London in 2013, including the underground lines, overground lines, and DLR.

• **Vickers** (Zhang et al., 2017): It is collected by Vickers from 29 seventh grade students in a school in Victoria by the questionnaire about their social relations, including affinity in the class, best friends and working together.

• **FFTWYT** (Dickison et al., 2016): It is an anonymized dataset acquired from three online social platforms: Friendfeed, Twitter, and YouTube, which records the relationship between the users registering their accounts on these platform.

• **CKM** (Coleman et al., 1957): It is a multiplex network collected from physicians in four towns in Illinois, Peoria, Bloomington, Quincy and Galesburg by the questionnaire about their opinions on medical innovation.

• **Plasmodium** (Stark et al., 2006): It considers different types of genetic interactions for organisms in the Biological General Repository for Interaction Datasets, including direct interaction, physical association and association.

• **Lazega** (Grossetti & Lazega, 2003): It is a multiplex social network which consists of three kinds of relationship between partners and associates of a corporate law partnership, including co-work, friendship, and advice.

• **Pierre** (Domenico et al., 2014): It is a multiplex dataset which consists of 16 layers corresponding to different working tasks within the Pierre Auger Collaboration between 2010 to 2012, i.e, neutrinos, detector, enhancements, anisotropy, point source, mass-composition, horizontal, hybrid reconstruction, spectrum, photons, atmospheric, sd-reconstruction, hadronic interactions, exotics, magnetic, and astrophysical scenarios.

• **FriendFeed** (Magnani & Rossi, 2011): It is an online social dataset, consisting of 21,006 nodes and 573,600 edges. It contains three layers which correspond to the interactions among users in Friendfeed, including commenting, liking, and following.

Table 1
The basic parameter setting of mLFR dataset.

mLFR dataset		
Parameter	Description	Value
N	Number of nodes	500
L	Number of layers	4
d	Average degree	16
$maxd$	Maximal degree	32
κ	Degree power-law	2
ψ	Community power-law	1
μ	Mixing parameter	[0.2, 0.25, 0.3, 0.35, 0.4]

Table 2
The statistics of real-world datasets.

Datasets	Description	Nodes	Edges	Layers
AUCs	Social	61	620	5
RM	Social	94	1385	3
C.elegans	Neuronal	279	5863	3
London	Transportation	369	441	3
Vickers	Social	29	740	3
FFTWYT	Social	6407	74 862	3
CKM	Social	246	1551	3
Plasmodium	Biological	1203	2521	3
Lazega	Social	71	2223	3
Pierre	Physical	514	7153	16
FriendFeed	Social	21 006	573 600	3

4.2. Parameters

For the node initial embedding, we generate an 128-dimensional vector in each network layer. For the Deepwalk, we set the window size as 10, the walks per node as 20 and the walk length as 40. For graph diffusion, we use the PPR by default. We set the teleport probability of PPR as 0.2 and the diffusion time of HK as 0.5. The dimension of all the hidden layers in GCN encoders is set to 512. We adopt the Adam optimizer with the initial learning rate 0.001. The loss coefficient λ is set to 0.1. For density coefficient ϵ, we will vary it in our ablation experiment. We release the code at: https://codeocean.com/capsule/6798104/tree/v1.

4.3. Competitors

For the competitors, we select the state-of-the-art algorithms, which can be divided into the following categories. The first group consists of those traditional community detection algorithms:

• PMM (Tang et al., 2009): It is a modularity optimization and matrix decomposition based algorithm called Principal Modularity Maximization. PMM has two main stages, i.e. structural feature extraction and cross-layer integration.

• MDLPA (Boutemine & Bouguessa, 2017): It extends label propagation algorithm (LPA) to multiplex networks by a flattening way. MDLPA is parameter-free and can dinnertime the number of communities automatically.

• CSNMF (Gligorijević et al., 2019): It is called Collective Symmetric Non-negative Matrix Factorization which optimizes reconstruction objective function to obtain the consensus, non-negative low dimensional matrix for community detection.

• DH-Louvain (Shao et al., 2022): It is a multi-greedy algorithm for community detection by optimizing a weighted modularity density.

The second group is three multiplex embedding algorithms, including one auto-encoder based method and two contrastive learning based methods.

• M-DeepWalk (Song & Thiagarajan, 2019) It is an auto-encoder based model which applies DeepWalk on the built supra graph from a multiplex network to capture multiplex structural features and then inputs the node embedding into an auto-encoder to detect communities.

Table 3
Performance on mLFR synthetic datasets in terms of NMI (%), ARI (%), Purity (%), and modularity.

Dataset	Metrics	PMM	MDLPA	CSNMF	DH-Louvain	M-DeepWalk	DMGI	HDMI	GEAM
$\mu = 0.2$	NMI	83.38 ± 0.35	79.82 ± 0.10	94.11 ± 0.33	95.24 ± 0.27	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**
	ARI	86.60 ± 0.32	78.45 ± 0.07	96.31 ± 0.31	99.17 ± 0.11	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**
	Purity	95.45 ± 0.33	92.26 ± 0.05	98.84 ± 0.24	99.72 ± 0.13	**100 ± 0**	**100 ± 0**	**100 ± 0**	**100 ± 0**
	Modularity	0.510 ± $5e^{-3}$	0.468 ± $2e^{-3}$	0.536 ± $7e^{-3}$	0.540 ± $1e^{-3}$	**0.546 ± 0**	**0.546 ± 0**	**0.546 ± 0**	**0.546 ± 0**
$\mu = 0.25$	NMI	80.90 ± 0.34	73.64 ± 0.04	89.44 ± 0.42	91.44 ± 0.31	90.71 ± 0.10	81.23 ± 0.14	85.33 ± 0.21	**94.73 ± 0.12**
	ARI	82.77 ± 0.25	57.96 ± 0.09	92.00 ± 0.37	94.37 ± 0.15	91.98 ± 0.07	82.43 ± 0.17	85.37 ± 0.20	**95.77 ± 0.15**
	Purity	94.24 ± 0.17	72.46 ± 0.06	97.41 ± 0.36	98.46 ± 0.21	97.45 ± 0.12	96.40 ± 0.10	95.25 ± 0.13	**98.25 ± 0.12**
	Modularity	0.449 ± $3e^{-3}$	0.387 ± $5e^{-4}$	0.477 ± $1e^{-2}$	0.471 ± $5e^{-3}$	0.478 ± $1e^{-3}$	0.452 ± $1e^{-3}$	0.475 ± $1e^{-3}$	**0.488 ± $2e^{-3}$**
$\mu = 0.3$	NMI	68.20 ± 0.22	72.84 ± 0.09	70.59 ± 0.32	74.48 ± 0.14	77.12 ± 0.22	75.38 ± 0.19	75.82 ± 0.27	**84.31 ± 0.21**
	ARI	72.14 ± 0.17	56.04 ± 0.13	75.15 ± 0.27	73.60 ± 0.20	77.67 ± 0.24	78.94 ± 0.20	76.52 ± 0.24	**85.32 ± 0.19**
	Purity	90.00 ± 0.21	69.65 ± 0.12	91.27 ± 0.24	91.52 ± 0.23	92.45 ± 0.19	92.20 ± 0.22	90.42 ± 0.15	**93.42 ± 0.20**
	Modularity	0.399 ± $2e^{-3}$	0.360 ± $1e^{-3}$	0.409 ± $5e^{-3}$	0.412 ± $2e^{-3}$	0.402 ± $1e^{-3}$	0.385 ± $6e^{-3}$	0.389 ± $3e^{-3}$	**0.427 ± $2e^{-3}$**
$\mu = 0.35$	NMI	51.87 ± 0.32	61.00 ± 0.14	35.96 ± 0.52	62.96 ± 0.33	61.91 ± 0.24	58.43 ± 0.34	52.52 ± 0.31	**70.92 ± 0.19**
	ARI	53.62 ± 0.36	45.12 ± 0.16	28.70 ± 0.39	49.63 ± 0.41	45.96 ± 0.26	55.27 ± 0.42	53.12 ± 0.27	**61.07 ± 0.28**
	Purity	80.81 ± 0.23	67.20 ± 0.11	61.65 ± 0.47	70.91 ± 0.43	67.42 ± 0.21	64.64 ± 0.24	67.87 ± 0.30	**83.12 ± 0.20**
	Modularity	0.303 ± $8e^{-3}$	0.311 ± $3e^{-3}$	0.318 ± $1e^{-2}$	0.325 ± $5e^{-3}$	0.316 ± $2e^{-3}$	0.307 ± $3e^{-3}$	0.312 ± $2e^{-3}$	**0.349 ± $3e^{-3}$**
$\mu = 0.4$	NMI	32.44 ± 0.24	38.37 ± 0.13	21.25 ± 0.79	45.14 ± 0.17	47.18 ± 0.31	35.12 ± 0.28	27.13 ± 0.35	**48.36 ± 0.31**
	ARI	28.56 ± 0.21	34.57 ± 0.15	18.26 ± 0.51	35.35 ± 0.37	38.84 ± 0.29	32.79 ± 0.16	21.62 ± 0.32	**39.63 ± 0.25**
	Purity	**65.04 ± 0.25**	60.00 ± 0.09	54.05 ± 0.54	61.27 ± 0.31	59.42 ± 0.21	56.83 ± 0.17	55.24 ± 0.25	64.00 ± 0.21
	Modularity	0.236 ± $2e^{-3}$	0.285 ± $1e^{-3}$	0.267 ± $2e^{-2}$	0.277 ± $1e^{-2}$	0.290 ± $4e^{-3}$	0.260 ± $3e^{-3}$	0.255 ± $3e^{-3}$	**0.310 ± $3e^{-3}$**

M-DeepWalk assigns a community label for a node in each layer and we use the majority voting to determine its final community label.

• DMGI (Park et al., 2020): It is a contrastive learning based algorithm which extends Deep Graph Infomax (Velickovic et al., 2019) to multiplex networks and obtains multiplex embeddings for several downstream tasks, one of which is community detection. We follow the origin paper and use K-means to obtain communities from the trained node embedding.

• HDMI (Jing et al., 2021): It proposes a High-order Deep Multiplex Infomax model to learn node embedding by optimizing high-order mutual information including both extrinsic and intrinsic mutual information. We test it in the same way as DMGI.

4.4. Metrics

We select some commonly used evaluation metrics of community detection, including NMI (Normalized Mutual Information), ARI (Adjusted Rand Index), Purity, multilayer modularity (Mucha et al., 2010). Among them, modularity is an unsupervised metric which usually measures the structural strength of communities in a multiplex network. It has two parameters, the resolution parameter γ_s and the coupling parameter $\mathscr{C}_{jsr}$, which are both set to 1. The multilayer modularity evaluation metric is computed by Eq. (18). It ranges from 0 to 1 and a higher modularity means a better result with strong community structures.

5. Results and analysis

5.1. Performance on synthetic datasets

Table 3 presents the modularity result of mLFR multiplex networks by varying the mixing parameter μ from 0.2 to 0.4. We notice that our GEAM outperforms the state-of-the-art algorithms in most synthetic datasets except a second best Purity with a small gap in the case of $\mu = 0.4$. We also obverse that the multiplex embedding algorithms usually perform better than those traditional ones. We conclude that those embedding methods take the advantage of the powerful representation learning ability. Though some of them are not directly related to the community detection task, i.e. DMGI and HDMI, they can train a general node embedding with the community-representation attributes.

For the traditional algorithms, the CSNMF has a good performance when $\mu \leq 0.25$ but performs the worst on NMI, ARI, and Purity when $\mu = 0.4$. A possible reason is that the inter-layer difference becomes larger when μ increases and will affect consensus matrix generation. The PMM has a sharp drop on modularity with the increase of μ, which means that it is hard for PMM to extract such modular features. While MDLPA keeps a relatively consistently high result among those traditional algorithms when $\mu \geq 0.3$. The MDLPA takes a flattening way to reduce multiple layers to single one and then apply the label propagation algorithm, which is not so sensitive to μ as CSNMF. The DH-Louvain performs better than other traditional algorithms especially when μ is small.

For the multiplex embedding algorithms, the M-DeepWalk performs better than DMGI and HDMI. The M-DeepWalk can generate the multiplex initial embedding by randow walk on the extended supra-graph; While DMGI and HDMI are proposed with built-in node attributes and such path-aware initial embedding generated by DeepWalk may not fit well. Besides, the HDMI is more dependent on the initial features because it considers the high-order mutual information between the initial features, the node encoding, and the graph encoding, which may cause that it performs not as good as DMGI when $\mu \geq 0.35$.

The superiority of our GEAM over the state-of-the-art algorithms suggests that the local and global information in each network layer are both significant for the task of community detection in multiplex networks. These improvements can be attributed to the global information encoded by layer contrastive learning, the multilayer semantic-aware information learned from self-attention adaptive fusion mechanism, and the compact community structures detected from the edge density-driven training.

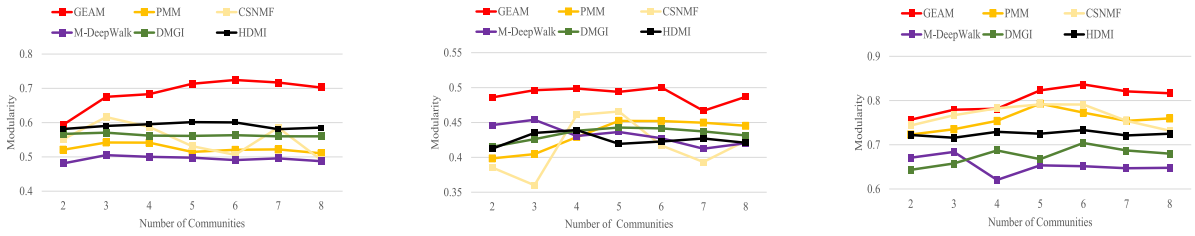
5.2. Performance on real-world datasets

Table 4 presents the modularity result on the real-world datasets. For all the algorithms except MDLPA and DH-Louvain, we test them with different numbers of communities and select the best result (corresponding community number shown in the brackets). Due to the fact that MDLPA and DH-Louvain can determine the community number automatically without extra parameters, we report its modularity directly.

A interesting observation is that some traditional algorithms, i.e. MDLPA and CSNMF, have a better performance than the multiplex embedding algorithms, i.e. M-DeepWalk, DMGI, and HDMI, on most real-world datasets, which is the opposite of the case on synthetic datasets. We guess that there are significant difference on topological structures between synthetic and the real-world datasets. The latter is usually collected from statistics data; While the former is obtained by sampling from the given community distribution. Therefore, those

Table 4
Modularity on real-world datasets.

Dataset	PMM	MDLPA	CSNMF	DH-Louvain	M-DeepWalk	DMGI	HDMI	GEAM
AUCs	0.5421 (3)	0.5512 (6)	0.6160 (3)	0.6057 (3)	0.5049 (3)	0.5721 (4)	0.6013 (5)	**0.7244** (6)
RM	0.5044 (2)	**0.5399** (4)	0.4718 (3)	0.5218 (4)	0.4969 (2)	0.4331 (4)	0.5235 (3)	0.5056 (5)
C.elegans	0.4523 (5)	0.4451 (8)	0.4658 (4)	0.4403 (4)	0.4538 (3)	0.4427 (5)	0.4391 (4)	**0.5005** (6)
London	0.7930 (6)	**0.9185** (47)	0.7915 (5)	0.8051 (9)	0.6838 (3)	0.7042 (4)	0.7331 (6)	0.8365 (6)
Vickers	0.2417 (3)	0.2255 (2)	0.2909 (2)	0.2251 (3)	0.2102 (3)	0.2379 (4)	0.2368 (5)	**0.2979** (3)
FFTWYT	0.2781 (10)	0.2849 (65)	0.2833 (6)	0.2704 (10)	0.1866 (3)	0.2477 (9)	0.2784(8)	**0.2903** (6)
CKM	0.4981 (7)	0.7653 (15)	0.6515 (5)	0.6135 (7)	0.4509 (6)	0.6408 (4)	**0.7743** (6)	0.7704 (4)
Plasmodium	0.6395 (9)	0.6079 (32)	0.7029 (35)	0.6511 (14)	0.5496 (21)	0.6271 (12)	0.5888 (14)	**0.7907** (42)
Lazega	0.2759 (3)	0.1036 (2)	0.3123 (3)	0.2673 (3)	0.0765 (3)	0.2907 (4)	0.3104 (3)	**0.3475** (3)
Pierre	0.8997 (10)	**0.9484** (8)	0.8961 (15)	0.8362 (6)	0.8409 (10)	0.8754 (12)	0.8903 (8)	0.9175 (10)
FriendFeed	0.2514 (12)	0.2870 (9)	0.3166 (12)	0.3461 (13)	0.2936 (10)	0.3147 (8)	0.3266 (5)	**0.3527** (12)



(a) AUCs (b) C.elegans (c) London

Fig. 3. Modularity with different pre-specified numbers of communities on three real-world datasets.

multiplex embedding algorithms can hardly extract the valid community structural information with high modularity from the real-world datasets. However, our GEAM applies the edge density-driven module for detecting communities with stronger modular structures and thus achieves the best modularity on seven datasets and the second best modularity on three datasets.

Some other analysis are as follows. We notice that MDLPA has a relatively good performance especially on London dataset, which is a sparse multiplex network and contains lots of isolated nodes in each network layer. Therefore, MDLPA can easily assign a community label to each isolated node to obtain a higher modularity. The CSNMF performs better than PMM in most cases. These two algorithms are both based on matrix decomposition while CSNMF introduces a collective constraint frame to extract the common feature representation for detecting communities. The DH-Louvain has an average performance for most real-world datasets. Among those multiplex embedding algorithms, the HDMI performs best and M-DeepWalk performs worst. We infer that the M-DeepWalk loses lots of intra-layer structural information when generating the supra-graph from a multiplex network.

Fig. 3 presents the modularity result with the number of communities varying. We obverse that GEAM have a consistently good performance on the three datasets, especially AUCs and C.elegans, which verifies the robustness of our algorithm. Another observation is that the modularity of CSNMF increases or drops sharply when community number changes. A possible reason is that the eigenvectors of matrix decomposition will change a lot because of the division or merger of communities, which may lead to a huge change of modularity.

5.3. Ablation study

5.3.1. Graph diffusion analysis

We conduct ablation experiments to investigate the effect of graph diffusion by comparing different kinds of graph diffusion methods, including PPR and HK mentioned in Section 3.1.1. Besides, we test another variant, called GEAM w/o gd, which removes the graph diffusion and replaces the layer contrastive learning module with DGI (Velickovic et al., 2019) model. Fig. 4 presents the modularity result on three real-world multiplex networks. We notice that GEAM w/o gd has a big

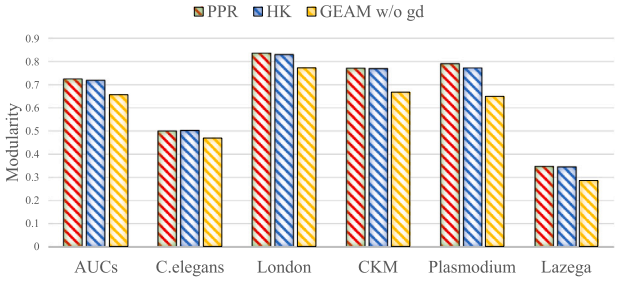


Fig. 4. Modularity with different graph diffusion.

gap with PPR and HK, which suggests the validity of graph diffusion module. Besides, we also obverse that PPR usually achieves a slightly higher modularity over HK, which indicates that PPR provides more appropriate global information for community detection.

5.3.2. Node initial embedding analysis

We then compare four kinds of node initial embedding to examine the effect of path-aware local topology. Except DeepWalk, we choose node2vec (Grover & Leskovec, 2016), one-hot, and centrality method to generate the initial embedding. For node2vec, we set the return parameter as 1 and the in-out parameter as 0.5. For centrality based embedding, we select four common centrality metrics, including degree centrality, betweenness centrality, closeness centrality, and load centrality. Table 5 presents the modularity result with different node initial embedding. It is no wonder that the one-hot performs the worst because it will disable the corruption strategy. Besides, the centrality method has a poor performance. We guess that it cannot well represent the local characteristics of each layer due to the lack of expert experience. The DeepWalk and node2vec both use the path-aware topology and achieve a better performance.

5.3.3. Multiplex fusion analysis

In order to analyze the effect of different fusion strategies, we consider two variants, i.e. GEAM w/o sa and GEAM sum. The first one removes the self-attention layer and inputs the node encoding into the

Table 5
Modularity with different node initial embedding.

Datasets	DeepWalk	node2vec	one-hot	centrality
AUCs	**0.7244**	0.6930	0.4231	0.6102
C.elegans	**0.5005**	0.4975	0.2786	0.4293
London	**0.8365**	0.8207	0.5714	0.6697
CKM	0.7704	**0.7713**	0.5605	0.6074
Plasmodium	**0.7907**	0.7634	0.4332	0.6107
Lazega	**0.3475**	0.3398	0.1504	0.2412

Table 6
Modularity with different fusion strategies.

Datasets	GEAM	GEAM w/o sa	GEAM sum
AUCs	**0.7244**	0.7163	0.6805
C.elegans	**0.5005**	0.4904	0.4887
London	**0.8365**	0.8135	0.8095
CKM	**0.7704**	0.7657	0.7611
Plasmodium	**0.7907**	0.7651	0.7307
Lazega	**0.3475**	0.3431	0.3407

adaptive fusion layer directly. For the second one, we replace the whole fusion module with a simple sum operation of node encoding in each network layer. Table 6 presents the modularity result with different multiplex fusion strategies. We obverse that the modularity drops when the self-attention layer is removed, which illustrates the effectiveness of layer-aware semantic information learned by the self-attention layer. Another observation is that in some datasets such as CKM and Lazega, the fusion module has little improvement compared to the sum operation. A possible reason is that the layer difference is small in these datasets and our fusion module treat the layers approximately equally.

5.3.4. Parameter analysis

To investigate the effect of different density coefficient ϵ, we test the modularity by varying ϵ from 0 to 1 with a step of 0.1 per increment. Fig. 5 presents the modularity result with varying ϵ in three real-world datasets, i.e. AUCs, C.elegans and Plasmodium. We can obverse that our GEAM is not sensitive to ϵ on AUCs and C.elegans dataset. A possible reason is that both high intra-community density and low inter-community density can contribute to the quality of detected communities. While in Plasmodium, a smaller ϵ corresponds to a higher modularity. We deduce that such cohesive topological structure, i.e. high intra-community density, is more important due to the sparsity of Plasmodium dataset.

5.3.5. Embedding visualization

To examine the effectiveness of the proposed GEAM, we present a case study about the visualization of learned nodes' representations **Z** in four real-world datasets, i.e., AUCs, C.elegans, CKM, and Pierre. Fig. 6 visualizes the nodes' belonging communities in different colors. We can see that the nodes in one communities are close in their representations in the four datasets.

6. Conclusion

In this paper, we have proposed a novel GEAM algorithm for community detection in multiplex networks. Our model contains a layer contrastive learning for encoding local and global information in each network layer, a self-attention adaptive fusion mechanism for extracting the importance of inter-layer semantics and acquiring a comprehensive node representation, and an edge density-driven community detection module for training our model in an end-to-end manner. Experiments on both synthetic and real-world datasets have validated the effectiveness of our GEAM over the state-of-art algorithms. An excellent feature of our model is to point out the importance of global information in each network layer for detecting communities in multiplex networks. In our future work, we shall investigate how to learn such kind of the global information more efficiently.

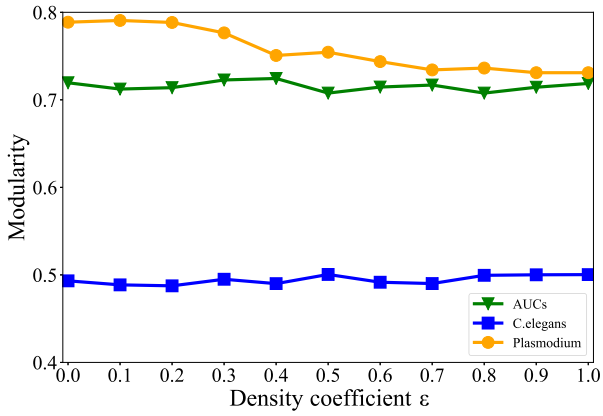


Fig. 5. Modularity with varying ϵ on three real-world datasets.

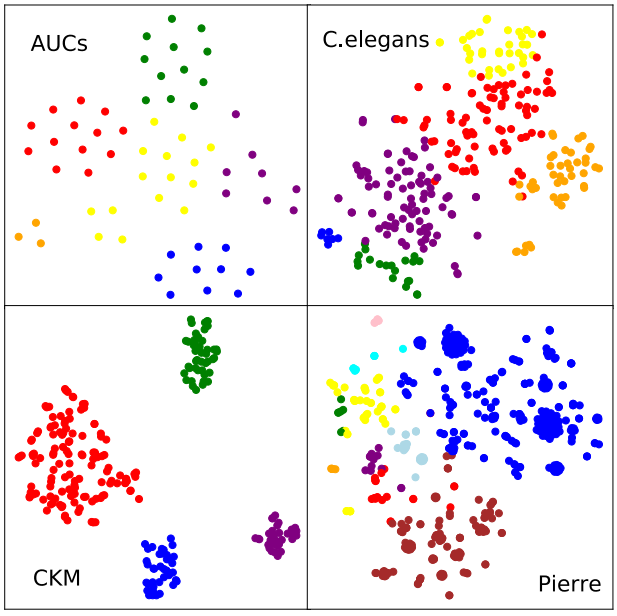


Fig. 6. Visualization of nodes' representations **Z** in the four real-world datasets. Nodes with the same color belong to the same community.

CRediT authorship contribution statement

Bang Wang: Methodology, Experiment, Analysis, Manuscript editing. **Xiang Cai:** Methodology, Experiment, Analysis, Manuscript editing. **Minghua Xu:** Methodology, Analysis. **Wei Xiang:** Methodology, Analysis, Manuscript editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Bang Wang reports financial support was provided by National Natural Science Foundation of China.

Data availability

Data will be made available on request.

References

Al-Andoli, M., Tan, S. C., & Cheah, W. P. (2022a). Parallel stacked autoencoder with particle swarm optimization for community detection in complex networks. *Applied Intelligence*, 1–21.

Al-Andoli, M. N., Tan, S. C., & Cheah, W. P. (2022b). Distributed parallel deep learning with a hybrid backpropagation-particle swarm optimization for community detection in large complex networks. *Information Sciences*, *600*, 94–117.

Ali, H. T., Liu, S., Yilmaz, Y., Couillet, R., Rajapakse, I., & Hero, A. (2019). Latent heterogeneous multilayer community detection. In *Proceedings of the international conference on acoustics, speech and signal processing* (pp. 8142–8146). IEEE.

Amini, A., Paez, M., & Lin, L. (2022). Hierarchical stochastic block model for community detection in multiplex networks. *Bayesian Analysis*, *1*(1), 1–27.

Berlingerio, M., Coscia, M., & Giannotti, F. (2011). Finding and characterizing communities in multidimensional networks. In *Proceedings of international conference on advances in social networks analysis and mining* (pp. 490–494). IEEE.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., & Cui, P. (2020). Structural deep clustering network. In *Proceedings of the world wide web conference* (pp. 1400–1410).

Boutemine, O., & Bouguessa, M. (2017). Mining community structures in multidimensional networks. *ACM Transactions on Knowledge Discovery from Data*, *11*(4), 1–36.

Bródka, P. (2016). A method for group extraction and analysis in multilayer social networks. arXiv preprint arXiv:1612.02377.

Chen, B. L., Hall, D. H., & Chklovskii, D. B. (2006). Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences*, *103*(12), 4723–4728.

Coleman, J., Katz, E., & Menzel, H. (1957). The diffusion of an innovation among physicians. *Sociometry*, *20*(4), 253–270.

Dai, L., Wang, B., Xiang, W., & Mo, Y. (2022). Bi-directional iterative prompt-tuning for event argument extraction. In *Proceedings of the 2022 conference on empirical methods in natural language processing*. Association for Computational Linguistics.

De Domenico, M., Solé-Ribalta, A., Gómez, S., & Arenas, A. (2014). Navigability of interconnected networks under random failures. *Proceedings of the National Academy of Sciences*, *111*(23), 8351–8356.

Dickison, M. E., Magnani, M., & Rossi, L. (2016). *Multilayer Social Networks*. Cambridge University Press.

Domenico, M. D., Lancichinetti, A., Arenas, A., & Rosvall, M. (2014). Identifying modular flows on multilayer networks reveals highly overlapping organization in social systems. *Computer Science*, *5*(1).

Du, W., Côté, D., & Liu, Y. (2023). Saits: Self-attention-based imputation for time series. *Expert Systems with Applications*, *219*, Article 119619.

Eagle, N., & Pentland, A. S. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, *10*(4), 255–268.

Gao, X., Zheng, Q., Verri, F. A., Rodrigues, R. D., & Zhao, L. (2019). Particle competition for multilayer network community detection. In *Proceedings of the 11th international conference on machine learning and computing* (pp. 75–80).

Gligorijević, V., Panagakis, Y., & Zafeiriou, S. (2019). Non-negative matrix factorizations for multiplex network analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(4), 928–940.

Grossetti, M., & Lazega, E. (2003). The collegial phenomenon: The social mechanisms of cooperation among peers in a corporate law partnership. *Revue Franaise de Sociologie*, *44*(1), 186.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).

Hassani, K., & Khasahmadi, A. H. (2020). Contrastive multi-view representation learning on graphs. In *International conference on machine learning* (pp. 4116–4126). PMLR.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., & Bengio, Y. (2018). Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670.

Huang, X., Chen, D., Ren, T., & Wang, D. (2021). A survey of community detection methods in multilayer networks. *Data Mining and Knowledge Discovery*, *35*(1), 1–45.

Jing, B., Park, C., & Tong, H. (2021). Hdmi: High-order deep multiplex infomax. In *Proceedings of the web conference 2021* (pp. 2414–2424).

Li, C., Chen, H., Li, T., & Yang, X. (2022). A stable community detection approach for complex network based on density peak clustering and label propagation. *Applied Intelligence*, *52*(2), 1188–1208.

Li, C., Guo, X., Lin, W., Tang, Z., Cao, J., & Zhang, Y. (2023). Multiplex network community detection algorithm based on motif awareness. *Knowledge-Based Systems*, *260*, Article 110136.

Li, B., Jing, B., & Tong, H. (2022). Graph communal contrastive learning. In *Proceedings of the ACM web conference 2022* (pp. 1203–1213).

Li, T., Lei, L., Bhattacharyya, S., Van den Berge, K., Sarkar, P., Bickel, P. J., & Levina, E. (2022). Hierarchical community detection by recursive partitioning. *Journal of the American Statistical Association*, *117*(538), 951–968.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11), 2579–2605.

Magalingam, P., Davis, S., & Rao, A. (2015). Using shortest path to discover criminal community. *Digital Investigation*, *15*, 1–17.

Magnani, M., Micenkova, B., & Rossi, L. (2013). Combinatorial analysis of multiple networks. arXiv preprint arXiv:1303.4986.

Magnani, M., & Rossi, L. (2011). The ML-model for multi-layer social networks. In *ASONAM* (pp. 5–12). IEEE Computer Society.

Mucha, P. J., Richardson, T., Macon, K., Porter, M. A., & Onnela, J.-P. (2010). Community structure in time-dependent, multiscale, and multiplex networks. *Science*, *328*(5980), 876–878.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*(2), Article 026113.

Park, C., Kim, D., Han, J., & Yu, H. (2020). Unsupervised attributed multiplex network embedding. *34*, In *Proceedings of the AAAI conference on artificial intelligence* (04), (pp. 5371–5378).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Pramanik, S., Tackx, R., Navelkar, A., Guillaume, J.-L., & Mitra, B. (2017). Discovering community structure in multilayer networks. In *2017 IEEE international conference on data science and advanced analytics* (pp. 611–620). IEEE.

Shao, Z., Ma, L., Lin, Q., Li, J., Gong, M., & Nandi, A. K. (2022). PMCDM: Privacy-preserving multiresolution community detection in multiplex networks. *Knowledge-Based Systems*, *244*, Article 108542.

Song, H., & Thiagarajan, J. J. (2019). Improved deep embeddings for inferencing with multi-layered graphs. In *Proceedings of the international conference on big data* (pp. 5394–5400). IEEE.

Stark, C., Breitkreutz, B.-J., Reguly, T., Boucher, L., Breitkreutz, A., & Tyers, M. (2006). Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, *34*(suppl_1), D535–D539.

Tagarelli, A., Amelio, A., & Gullo, F. (2017). Ensemble-based community detection in multilayer networks. *Data Mining and Knowledge Discovery*, *31*(5), 1506–1543.

Tang, L., Wang, X., & Liu, H. (2009). Uncoverning groups via heterogeneous interaction analysis. In *Proceedings of the 9th international conference on data mining* (pp. 503–512). IEEE.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., & Hjelm, R. D. (2019). Deep graph infomax. *ICLR (Poster)*, *2*(3), 4.

Wang, R., Ao, J., Zhou, L., Liu, S., Wei, Z., Ko, T., Li, Q., & Zhang, Y. (2022). Multi-view self-attention based transformer for speaker recognition. In *ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing* (pp. 6732–6736). IEEE.

Xia, J., Wu, L., Chen, J., Hu, B., & Li, S. Z. (2022). Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM web conference 2022* (pp. 1070–1079).

Yang, D. U., Kim, B., Lee, S. H., Ahn, Y. H., & Kim, H. Y. (2022). AutoDefect: defect text classification in residential buildings using a multi-task channel attention network. *Sustainable Cities and Society*, Article 103803.

Zhang, H., Wang, C.-D., Lai, J.-H., & Philip, S. Y. (2017). Modularity in complex multilayer networks with multiple aspects: a static perspective. In *Proceedings of the applied informatics* (pp. 1–29). SpringerOpen.

Zhou, Z., Hu, Y., Zhang, Y., Chen, J., & Cai, H. (2022). Multiview deep graph infomax to achieve unsupervised graph embedding. *IEEE Transactions on Cybernetics*.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021* (pp. 2069–2080).