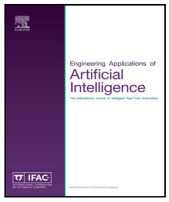
Engineering Applications of Artificial Intelligence 126 (2023) 106717

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



MVMA-GCN: Multi-view multi-layer attention graph convolutional networks

Pengyu Zhang, Yong Zhang *, Jingcheng Wang, Baocai Yin

Faculty of Information Technology, Beijing University of Technology, Beijing, 100124, China

ARTICLE INFO

Keywords:
Graph analysis
Graph convolutional networks
Semi-supervised classification
Graph neural networks

ABSTRACT

The accuracy of graph representation learning is highly dependent on the precise characterization of node relationships. However, representing the complex and diverse networks in the real world using a single type of node or link is challenging, often resulting in incomplete information. Moreover, different types of nodes and links convey rich information, which makes it difficult to design a graph network that can integrate diverse links. This paper introduces a novel multi-view and multi-layer attention model designed to optimize node embeddings for semi-supervised node classification. The proposed model exploits various types of inter-node links and employs the Hilbert–Schmidt independence criterion to maximize the dissimilarity between distinct node relationships. Furthermore, the multi-layer attention mechanism is used to discern the impact of different neighboring nodes and relationships between various node relationships. The performance of the proposed model, MVMA-GCN, was assessed on numerous real-world multi-view datasets. It was observed that MVMA-GCN consistently outperformed existing models, demonstrating superior accuracy in semi-supervised classification tasks. We have made our code publicly available at here to ensure the reproducibility of our results.

1. Introduction

Networks are ubiquitous in the real world, such as social networks and academic paper collaboration networks. Convolutional Neural Networks (CNNs) have been the preferred choice due to their remarkable performance compared to classic handcrafted feature engineering (Pramanik et al., 2022a). Originally proposed for document recognition and later extended to multiple fields such as image, video, speech, and audio processing, researchers typically perform deep feature extraction using CNNs (Pramanik et al., 2022c). However, the use of CNNs does not address every challenge.

Graph structure is naturally suited for representing network structure, and Graph Neural Networks (GNNs) provide a practical framework for graph representation learning (Xu et al., 2018). GNNs have demonstrated impressive performance in many scientific and engineering fields and can uncover underlying relationships among data (Scarselli et al., 2008). For instance, the Graph Convolutional Network (GCN), a neural network specifically designed for non-Euclidean data, has shown superior performance in areas such as recommendation systems for shopping websites (Ying et al., 2018) and social networks (Defferrard et al., 2016; Aburahmah et al., 2016; Li et al., 2022), protein structure analysis (Zhang et al., 2018; Gao and Ji, 2019), drug discovery (Cheung and Moura, 2020), and pandemic wave prediction (Xue et al., 2022; Pramanik et al., 2022b).

Graph data has a wide range of applications in predicting epidemics, preventing avoidable deaths, and improving quality of life (Syed et al., 2019). However, graph data faces challenges in accurately summarizing complex relationships in the real world. Graph data based on associative relationships in the real world is often noisy, inaccurate, and incomplete (Tong et al., 2014). Although the success of single-view data (data with only one type of node relationship) is partially due to its ease of understanding and model design (Bo et al., 2020), it also results in information loss in the network (Wang et al., 2021) and weak performance in message passing effectiveness (Hoang and Maehara, 2019; Wu et al., 2019; Gao et al., 2019). In contrast, multi-view data (data with multiple types of node relationships) can more accurately capture the connections between nodes (Fan et al., 2020; Yin et al., 2020).

For instance, as shown in Fig. 1, when modeling academic paper collaboration networks, there are three types of relationships between nodes: co-author, co-conference, and co-keyword. Single-view modeling only considers one relationship between nodes, whereas multi-view modeling needs to fuse different node relationships with accurate weights.

Due to the complexity of multi-view node relationships, traditional GNN methods cannot accurately describe multi-view networks. When designing GNNs with multiple types of node relationships, we need to address several issues. First, we need modules that can fully capture multiple relationships between nodes to avoid insufficient node information captured by the model. Second, few models can adaptively process multiple node relationships. Third, despite the great success of

* Corresponding author.
E-mail addresses: p.zhang@uva.nl (P. Zhang), zhangyong2010@bjut.edu.cn (Y. Zhang), wang.jc@emails.bjut.edu.cn (J. Wang), ybc@bjut.edu.cn (B. Yin).

https://doi.org/10.1016/j.engappai.2023.106717
Received 30 January 2022; Received in revised form 7 May 2023; Accepted 23 June 2023
Available online xxxx
0952-1976/

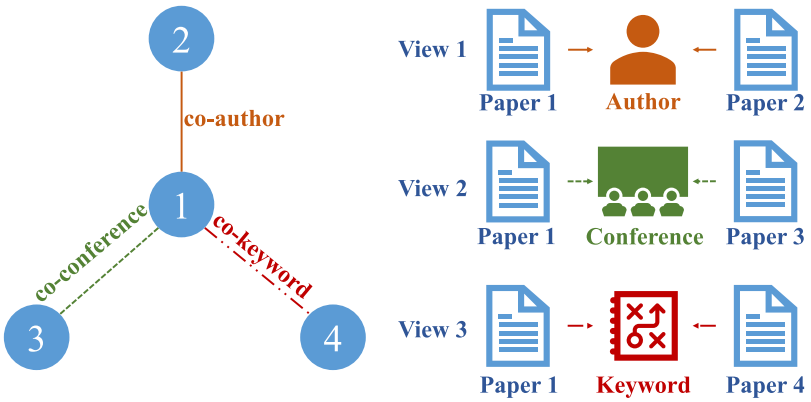


Fig. 1. An illustrative example of a multi-view graph. Figure shows four academic papers consists three types of relationships: co-author, co-conference, and co-keyword.

GNNs in graph analysis, existing works require a way to effectively combine the strengths of autoencoders and GCNs and incorporate structural information captured by k-NN into GCNs.

In response to these challenges, we introduce MVMA-GCN, a novel approach of multi-view, multi-layer attention graph convolutional network. The fundamental idea of our model is to use multiple type of links to encapsulate the complex interrelationships between nodes. A multi-layer attention module that assign different weights to different nodes, and learn the accurate representation of nodes. Additionally, we propose an autoencoder module to extract useful representations from the data and improve classification results. Our work can be summarized as follows.

- We propose a graph convolution network that can process graph data containing multiple relational. Our model fully captures the differences in the multiple relationships between nodes using the Hilbert–Schmidt independence criterion on various node relationship graphs.
- We propose a multi-layer attention mechanism that can adaptively learn the importance of different neighbor nodes and views. Specifically, the single-view attention layer aims to learn the importance between a node and its neighbors, while the multi-view attention layer can learn the importance of different views. And then naturally combined with the node's feature by the autoencoder module.
- Our extensive experiments on several benchmark datasets show the superiority of the proposed model, and MVMA-GCN outperforms the baseline models. More importantly, by analyzing the different modules, our model demonstrates its effectiveness on multiple relationship graphs.

The rest of this article is organized as follows: In Section 2 we reviewed and analyzed the related work. In Section 3 we formally defined the research problem. In Section 4 we develop MVMA-GCN. The experimental setup and results of the proposed method are represented in Section 5. Finally, the conclusions and limitations are discussed in Section 6.

2. Related work

Our work involved graph embedding, attention mechanism, and semi-supervised classification. This section reviews the previous research of these three parts and analyzes the advantages and limitations of the previous research.

2.1. Graph embedding

Graphs are a common way of representing real-world scenarios. To enable faster computation of graph data, the network structure in the graph must be transformed into vectors, known as graph embedding. Graph embedding can embed network information into low-dimensional space while preserving the relationships between nodes. Additionally, graph embedding can apply the learned embedding to subsequent tasks, thereby improving computational efficiency.

Various methods have been proposed for generating graph embeddings. For example, Deepwalk (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) are based on random walk, while the SDNE model (Wang et al., 2016) uses a deep neural network to perform graph embedding and an autoencoder to maintain the proximity of the first-order and second-order networks. Ref. Wang et al. (2017) is based on a matrix decomposition approach, while the Line model (Tang et al., 2015) ensures that the information network retains its first- and second-order similarity when embedded in a low-dimensional space.

In recent years, deep learning-based graph embedding methods have gained widespread attention. For example, DNGR (Cao et al., 2016) integrates random walk and deep autoencoders to generate probabilistic co-occurrence matrices using a random walk model on the input graph, which is fed into a superposition denoising autoencoder to obtain the embedding. However, this method is computationally expensive for sparse graphs. GCN solves this problem by defining a convolution operator on the graph. The model iteratively aggregates the neighborhood embeddings of nodes and uses the embedding obtained in the previous iteration and its embedding function to obtain a new embedding. Many studies based on GCN have been conducted in recent years (Ma et al., 2019; Qu et al., 2019). For example, GAT (Veličković et al., 2017) uses an attention mechanism, which uses learned weights to aggregate node features. GraphSAGE (Hamilton et al., 2017) uses the mean/max/LSTM pool to sample and aggregate features from the local neighborhood of nodes.

Previous works have focused on better utilizing graph structure so that node features can be more efficiently propagated. However, there is still much potential to be explored in capturing different neighbor nodes and relationship graphs.

2.2. Attention mechanism

The attention mechanism in neural networks mimics the human attention mechanism. When humans view a picture, they quickly locate the area that requires focus and devote more attention to it while ignoring other information. In an attention network, the output at a specific moment is determined by the attention it distributes among multiple inputs, i.e., the weights assigned to each input. A significant weight indicates that this input contributes more to the output. The attention mechanism has been successfully used in seq2seq models (Cho et al., 2014), and its ability to weight different inputs has been shown to improve model performance.

Various types of attention mechanisms have been proposed, such as hard attention (Xu et al., 2015), global attention, and local attention

(Luong et al., 2015). Multi-head attention has also been introduced to map the node representation into multiple node representations through a linear mapping, calculate the scaled dot-product attention separately, and combine the calculation results (Vaswani et al., 2017). These attention mechanisms have demonstrated significant potential in various fields and have improved model performance.

Therefore, multi-layer attention is well-suited for fusing different neighbor nodes and views to describe the predicted node. It can effectively learn the importance of different neighbor nodes and views and improve the model's ability to capture relationships between nodes.

2.3. Semi-supervised classification

Semi-supervised learning combines supervised and unsupervised learning by using labeled and unlabeled data in the training process. In many applications, a large amount of unlabeled data can be easily collected, making semi-supervised learning a practical and important approach. However, labeling data can be labor- and material-intensive. The key idea behind semi-supervised learning is that better classification results can be achieved by leveraging the local features of labeled data and the larger amount of unlabeled data, as long as the data distribution is not completely random.

Previous works on semi-supervised learning on graph-structured data have used both single-view and multi-view inputs. However, using only a single view may not capture the complex relationships between nodes as comprehensively as multi-view inputs. Therefore, multi-view inputs have been shown to produce more accurate classification results. However, only a few studies have considered the impact of different neighbor nodes and views on the classification results.

In an effort to address this issue, adaptive multi-channel graph convolutional networks have been proposed for semi-supervised classification (Wang et al., 2020). These networks extract specific and common embeddings from node features and learn adaptive weights using the attention mechanism. Another approach involves creating new graph features based on cosine similarity and combining them with node attributes and graph topologies for graph convolutional networks (Tang et al., 2021). By considering the impact of different neighbor nodes and views, these methods have demonstrated improved classification results in semi-supervised learning on graph-structured data.

3. Preliminaries

This section introduces some concepts and symbols we used in this paper.

Definition 1 (*Networks, Graphs, Nodes, and Edges*)**.** In real-world scenarios, there are many entities that are associated with each other in various ways, forming networks. To represent these relationships mathematically, we can use graphs. In a graph, entities are represented as nodes, and the associations are represented as edges connecting the nodes. The graph itself is denoted as G, and the set of nodes is denoted as V, while the set of edges that connect the nodes is denoted as E.

Definition 2 (*Single Type of Link and Multiple Types of Links*)**.** In the study of networks, a single view only captures one type of relationship between nodes, which can lead to loss of important information. Multi-view, on the other hand, captures various relationships between nodes, providing a more comprehensive understanding of the network structure. Mathematically, a multi-view graph is denoted as $G = \left(V, E^{(1)}, E^{(2)}, \ldots, E^{(m)}, X\right)$, where $V = \left\{v_i\right\}_{i=1}^{n}$ represents the set of nodes in the graph, $E^{(m)}$ represents the set of edges between nodes in the mth view, and $X = \left\{x_i\right\}_{i=1}^{n}$ represents the features of the nodes. The connections between nodes in G can be represented by multiple adjacency matrices $\left\{A^{(m)}\right\}_{m=1}^{M}$, where $a_{i,j}^{(m)} = 1$ if there is a connection between node i and node j in the mth view, and $a_{i,j}^{(m)} = 0$ otherwise. However, in our work, we do not consider connections between a node and itself, so $a_{i,j}^{(m)} = 0$ if $i = j$.

The relationships between papers in the ACM dataset are represented using a multi-view model, as shown in Fig. 1. The nodes represent the papers, and there are three different types of relationships, each represented as a view: co-author (paper–author–paper), co-conference (paper–conference–paper), and co-keywords (paper–keyword–paper). The nodes can be connected based on any of these relationships, and each relationship is considered as a separate view. By incorporating multiple views, this model can more accurately capture the relationships between papers and extract more useful information for subsequent analysis.

Definition 3 (*Semi-Supervised Node Classification Based on Multi-View*)**.** In our work, we aimed to use the multiple relationships between nodes to divide the nodes into K predefined different clusters $\left\{C_1, C_2, \ldots, C_K\right\}$, to ensure that nodes with similar attributes are close to each other and nodes with different attributes are far from each other. A small number of labeled nodes was provided for semi-supervised classification, which is a common setting in node classification (Wang et al., 2020).

4. MVMA-GCN: The proposed model

The intricate interconnections among entities present a significant challenge when attempting to encapsulate node structure information solely via a single view representation of node relationships. Take Fig. 1 for example: a simplistic focus on the co-author relationship between papers for classification purposes neglects the potential existence of other relationships, such as co-keyword or co-conference affiliations. When a multitude of node relationships is employed as input, the task of assigning weights to these various inputs becomes complex. The integration process must assure that the model effectively filters out extraneous noise and selectively extracts the most pertinent node information from the multiple views. Additionally, it is crucial to circumvent the risk of model overfitting.

To address these challenges, we introduce the MVMA-GCN model. The structure of this model is illustrated in Fig. 2. At its core, the model employs a single-view attention layer to ascertain the significance of the relationship between a node and its neighboring nodes, while a multi-view attention layer is utilized to discern the importance of different views. This enables the model to precisely learn node representation. Subsequently, an autoencoder module is utilized to amalgamate the multiple data structures with the multiple representations. Detailed explanations of the model's components are provided in the four subsequent sections.

(1) ***Graph Convolutional Network for Multi-View Information Extraction.*** The proposed model takes as input two parts: node features and a multi-view consisting of node relationship graphs. The node features are extracted from the original data and a k-NN graph is constructed based on them. The multi-view ensures that the model can capture the complex relationships between nodes more comprehensively and accurately.

(2) ***Autoencoder for Data Representation.*** The autoencoder module are connected to the GCN. Each layer in the autoencoder module integrates the information between the nodes learned in the autoencoder into the GCN to ensure accurate data representation learning.

(3) ***Multi-Layer Attention for Multi-View Input.*** The multi-layer attention module is designed to discern and assign different weights across varying nodes and views, which enables the model to adaptively learn the importance of different neighbor nodes and views. The learned node representations are then predicted by a multilayer perceptron.

(4) ***Objective Function.*** To learn the differences between multiple types of links better, the proposed model uses the Hilbert–Schmidt independence criterion as the loss function to learn the representation between different types of links. We use similarity matrix to learn the similarity representation between different views. Finally, an autoencoder is added to form the final loss function.

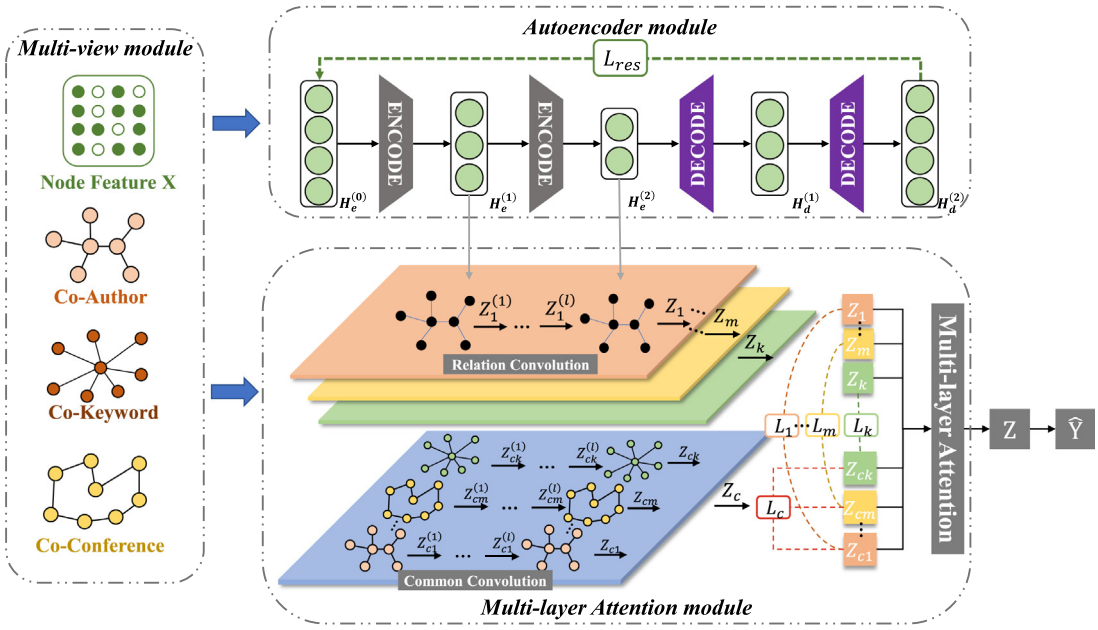


Fig. 2. The framework of MVMA-GCN model. Node feature X is to construct a feature graph. Multi-layer attention module consists of three relation convolution, one common convolution and the attention mechanism.

4.1. GCN for multi-view information extraction

A single view can only partially represent the node relationships, leading to node classification deviations. Some previous works used multi-view but only learned each view individually, so they could not learn the shared representation between views. Our approach involves adding multi-view convolution to learn the shared parameters of all views while using separate single-view convolutions for each view. The independent output of each view and the shared output of multiple views are collectively used as the output of this module.

Feature Matrix X ***and Feature Graphs*** A_k. Initially, we extract salient information that best represents the node from the original datasets to serve as the node's features. For instance, in a paper dataset, we might use the paper's keywords, while in a movie dataset, the movie synopsis could be utilized. Subsequently, these node features are utilized to construct a node feature matrix, denoted as X. Here, $x_i \in X$ symbolizes the feature vector of the ith node. This feature matrix X is employed as the initial input to the model.

Following this, the node feature matrix X is leveraged to compute the nodes' similarity matrix, represented as S. The calculation methodology harnesses the cosine of the angle between two vectors as a measure of similarity. Consequently, S_{ij} denotes the similarity matrix of node i and node j, where x_i and x_j respectively represent the feature vectors of nodes i and j.

$$S_{ij} = \frac{x_i \cdot x_j}{|x_i| \cdot |x_j|}. \tag{1}$$

Lastly, we determine the existence of connections between nodes by calculating the similarity between them. Based on the node features, we construct a k-nearest neighbor graph and represent it as the feature graph A_k. This graph is used as the second input to the model, allowing it to capture the structural information of the relationships between the nodes.

Node Relationship Graphs $\{A_m\}_{m=1}^{M}$. The proposed model takes into account the multi-view relationship graphs constructed based on nodes' relationships, in addition to the feature matrix and feature graph based on node features. The multi-view relationship graphs are represented as $\{A_m\}_{m=1}^{M}$, where M is the number of views. For instance, in the DBLP dataset, there are three relationships between nodes, so $M = 3$. If nodes i and j are connected in a particular view, then the corresponding value in the adjacency matrix is 1, and 0 otherwise. Self-connections are not considered. These relationship graphs are used as the third input to the model, enabling it to capture the complex relationships between nodes from different perspectives.

Single-View Convolution Z_k ***and*** Z_m. We use (A_k, X) and (A_m, X) as inputs to single-view convolution. Each of the inputs is passed through a separate convolution module, which produces two outputs, Z_k and Z_m. The lth layer of the convolution can be represented as follows:

$$Z_k^{(l)} = ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}_k\tilde{D}^{-\frac{1}{2}}\tilde{Z}_k^{(l-1)}W^{(l)}\right), \tag{2}$$

$$Z_m^{(l)} = ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}_m\tilde{D}^{-\frac{1}{2}}\tilde{Z}_m^{(l-1)}W^{(l)}\right), \tag{3}$$

in this context, $W^{(l)}$ represents the weight matrix associated with the lth layer of the Graph Convolutional Network (GCN). The preliminary Z_m is defined as $Z_m^{(0)} = X_{m(att)}$, where $X_{m(att)}$ refers to the node embedding that is learned via the single-view attention network, specifically within view m, the details are described in Section 4.3. The initial Z_k is denoted as $Z_k^{(0)} = X$. The augmented adjacency matrices, $\tilde{A}_k$ and $\tilde{A}_m$, are defined as A_k+I and A_m+I, respectively. Additionally, $\tilde{D}$ represents the diagonal matrix corresponding to $\tilde{A}$.

Multi-View Convolution Z_c. Learning each view in isolation often proves insufficient for identifying commonalities between different views. Thus, we incorporate multi-view convolution to extract the shared information across various views. All previous inputs are utilized as inputs for the multi-view convolution. The resultant output from the multi-view convolution module is denoted as Z_c, and the output from the lth layer of the convolution can be formulated as:

$$Z_c^{(l)} = ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}_c\tilde{D}^{-\frac{1}{2}}Z^{(l-1)}W^{(l)}\right), \tag{4}$$

here, $W^{(l)}$ signifies the weight matrix of the lth layer of the GCN. The initial Z is defined as $Z^{(0)} = X$. The augmented adjacency matrix $\tilde{A}$ is computed as $A_c + I$, where A_c is the sum of A_k and the set $\{A_m\}_{m=1}^{M}$. Furthermore, $\tilde{D}$ represents the diagonal matrix corresponding to $\tilde{A}$.

4.2. Autoencoder for data representation

Effective data representation learning is critical for classification tasks. As such, we implement an autoencoder to comprehend the raw

data representations. This approach allows for the accommodation of diverse data characteristics and the subsequent transmission of the acquired knowledge to the pertinent GCN layer.

Use Autoencoder to Extract Node Representation H. Firstly, if we assume that the autoencoder comprises L layers, then the expression acquired in the lth layer within the autoencoder is denoted as $H_e^{(l)}$:

$$H_e^{(l)} = ReLU\left(W_e^{(l)} H_e^{(l-1)} + b_e^{(l)}\right), \tag{5}$$

in this equation, $ReLU$ serves as the activation function for the fully connected layer, while $W_e^{(l)}$ and $b_e^{(l)}$ respectively represent the weight matrix and bias of the lth layer in the autoencoder. Furthermore, $H_e^{(0)}$ is defined as the feature matrix X.

The encoder part's outcome is $H_e^{(l)}$. Then, we use the decoder part to restore the node representation $H_d^{(l)}$; the representation learned at layer l in the autodecoder is:

$$H_d^{(l)} = ReLU\left(W_d^{(l)} H_d^{(l-1)} + b_d^{(l)}\right). \tag{6}$$

Passing the Node Representation into the GCN Module. Initially, the node representations obtained from the autoencoder, such as $H_e^{(1)}$, $H_e^{(2)}$, …, and $H_e^{(L)}$, are fed into the GCN module. This allows the GCN to encapsulate two distinct kinds of information: the data itself and the structure of the data. For instance, the output from the lth layer, as learned within a single view, can be represented as $Z_k^{(l)}$.

Subsequently, given that the representation $H_e^{(l)}$, as learned by the autoencoder, can reconstruct the data and contains various valuable information, merging the two representations facilitates a more comprehensive representation:

$$\tilde{Z}_k^{(l-1)} = (1-\epsilon) Z_k^{(l-1)} + \epsilon H_e^{(l-1)}, \tag{7}$$

here, ϵ is a hyperparameter with an initial value set to 0.5 (ranging from 0 to 1). It should be noted that we employ $Z_k^{(l)}$ in conjunction with the encoder feature $H_e^{(l-1)}$, as opposed to the decoder feature $H_d^{(l-1)}$. This is because our goal is to maximize the similarity between H_e and H_d through the reconstruction process, ensuring that H_e can more accurately represent the structural information of the original data. As such, the autoencoder and GCN can be interconnected on a layer-by-layer basis.

4.3. Multi-layer attention for multi-view input

The multi-layer attention module operates in three distinct stages: Initially, the module employs the single-view attention layer to ascertain the impact of various neighboring nodes. Subsequently, the multi-view attention layer is harnessed to understand the influence of different views. Finally, these two components are amalgamated to derive the anticipated representation of the nodes.

Single-View Attention Layer. In contrast to prior methods, it is crucial to acknowledge that each node's neighbors exert distinct influences, and features are derived from a variety of perspectives through different relationships. Prior to calculating the effect of different views on the prediction outcomes, it is necessary to compute the influence between nodes within the same view. Given that each node plays a unique role in the node embedding process, they each have a disparate impact on the final result. The single-view attention layer can learn the influence of various neighbor nodes on the predicted nodes within each view. Initially, self-attention (Vaswani et al., 2017) is employed to ascertain the weights between different node. For instance, in a view m, for a given node pair (i, j), the significance of node j to node i can be calculated as:

$$\alpha_m(i,j) = \frac{\exp\left(LeakyReLU\left(a_m^T \cdot \left[x_i \parallel x_j\right]\right)\right)}{\sum_{k\in N} \exp\left(LeakyReLU\left(a_m^T \cdot \left[x_i \parallel x_k\right]\right)\right)}, \tag{8}$$

here, $\parallel$ denotes the concatenation operation; x_i, x_j, and x_k are the node features of nodes i, j, and k, respectively; and in the single view m, a_m^T is the attention vector.

Next, the node embedding of node i in the view can be achieved by aggregating the neighbor nodes' features with feature coefficients. To enhance the stability of the training process, we utilize multi-headed attention. Specifically, the single-view attention layer is repeated K times, and the acquired embedding is connected to the embedding of a specific view. Finally, the learned node embedding and the node feature matrix are concatenated to yield $X_{m(att)}$, where $z_m(i)$ represents the embedding of node i as learned in view m.

$$z_m(i) = \|_{k=1}^{K} \; Sigmoid\left(\sum_{j\in N} \alpha_m(i,j) \cdot x_j\right). \tag{9}$$

Multi-View Attention Layer. In order to learn node embeddings effectively, it is crucial to amalgamate multiple node embeddings acquired from different views. Given that weights assigned between various views differ across nodes or datasets, there is a need for a module capable of automatically assigning weights to different views in order to address this challenge.

First, the output from the preceding stage is utilized as input for this step, which entails using the single-view graph convolution Z_k and $\left\{Z_m\right\}_{m=1}^{M}$ alongside the multi-view convolution Z_c. For node i, a nonlinear transformation is performed on the node embedding, followed by employing the shared attention vector q to compute the attention value ω^i:

$$\omega^i = q^T \cdot \tanh\left(W \cdot \left(z^i\right)^T + b\right), \tag{10}$$

here, W represents the weight matrix, and b denotes the bias. The attention value of node i in other embedding matrices can be obtained through a similar approach, with q serving as the attention vector employed to gauge the significance of node embedding z^i.

Then, a function is employed to normalize multiple attention values, thereby determining the final weight:

$$\alpha_k^i = \frac{\exp\left(\omega_k^i\right)}{\exp\left(\omega_k^i\right) + \exp\left(\omega_m^i\right) + \exp\left(\omega_c^i\right)}, \tag{11}$$

a larger α suggests that the embedding holds greater importance. The same process can be applied to calculate α_m^i and α_c^i.

Lastly, the three embeddings are combined to yield the final embedding. A higher α^i value indicates a more crucial view.

$$Z = \alpha_k \cdot Z_k + \alpha_m \cdot Z_m + \alpha_c \cdot Z_c. \tag{12}$$

4.4. Objective function

In order to achieve high classification accuracy, we use the single-view loss function L_s, multi-view loss function L_m, reconstruction loss function L_{res}, and cross-entropy loss L_t.

Single-View Loss Function L_s. In the single-view and multi-view convolution parts, since the input is the same, they are both k-NN graphs and have a specific relationship graph. To allow our model to capture richer information, we try to let our model learn different node representations. In other words, the goal is to amplify the differences among Z_k, Z_m, and Z_c. To quantify the independence between these diverse outputs, we adopt the Hilbert–Schmidt Independence Criterion (HSIC) (Wang et al., 2020).

$$HSIC\left(Z_1, Z_2\right) = (n-1)^{-2} tr\left(RK_1RK_2\right), \tag{13}$$

here, K_1K_2 represents the Gram matrix; $k_{1,ij}$ is equivalent to $k_1\left(z_1^i, z_1^j\right)$, and similarly, $k_{2,ij}$ corresponds to $k_2\left(z_2^i, z_2^j\right)$. The term R is given by $I - \frac{1}{n} ee^T$, where I denotes the identity matrix and e is a column vector entirely composed of ones. The inner product kernel function is utilized for the computation of K_1K_2.

$HSIC$ also calculates all other views; the single-view loss function set as:

$$L_s = HSIC\left(\tilde{Z}_k, Z_m\right) + HSIC\left(\tilde{Z}_k, Z_c\right) + HSIC\left(Z_m, Z_c\right). \tag{14}$$

Multi-View Loss Function L_m. In the case of the multi-view loss function, our objective is to enhance the consistency across various views via convolution. Initially, the model employs $L2$ normalization to standardize the matrices $\{Z\}\, i = 1^M$, yielding $\{Z\text{nor}\}\, i = 1^M$. Following this, we use these two normalized matrices to ascertain the similarity between nodes, denoted by $\{Si\}_{i=1}^{M}$:

$$\left\{S_i\right\}_{i=1}^{M} = Z_{\text{nor}} \cdot Z_{\text{nor}}^{T}. \tag{15}$$

Because we want the two similarity matrices to be as similar as possible, the loss function is:

$$L_m = \left\|S_k - S_m\right\|_F^2 + \left\|S_k - S_c\right\|_F^2 + \left\|S_c - S_m\right\|_F^2. \tag{16}$$

Reconstruction Loss Function L_{res}. Given that the output of the decoder reconstructs the original data, we can derive the following objective function:

$$L_{res} = \frac{1}{2N}\left\|X_i - \hat{X}_i\right\|_F^2, \tag{17}$$

where X_i is the node feature matrix, $\hat{X}$ is the feature matrix restored by the automatic decoder, that is, H_d^l in Eq. (6).

Optimization Objective. Embed the model Z output into Eq. (14), and use the softmax function for semi-supervised multi-class classification. Denote the class prediction of n nodes as $\widetilde{Y} = [\widetilde{y}_{ic}]$, where $\widetilde{y}_{ic}$ represents the probability that the node belongs to class c. The formula of $\widetilde{Y}$ is:

$$\hat{Y} = softmax(W \cdot Z + b), \tag{18}$$

Suppose the training set is L, the node's label is Y_i, and the predicted label is $\hat{Y}_i$. Then, calculate the cross-entropy loss of all training nodes as:

$$L_t = -\sum_{l \in L}\sum_{i=1}^{C} Y_{li} \ln \hat{Y}_{li}. \tag{19}$$

Combining the node classification task and constraints, the final loss function is L:

$$L = L_t + aL_m + bL_s + cL_{res}. \tag{20}$$

where a, b, and c are the parameters of the multi-view module, multi-layer attention module, and autoencoder module, respectively. We can optimize the model by backpropagation and learn the node embedding for node classification.

5. Experimental results and analysis

In this section, we assess our model's performance by comparing it to state-of-the-art baseline models. All models were run using the same parameters for ten trials, and the average results were recorded. Model performance was evaluated based on accuracy and F1 scores, with higher values indicating better classification performance. The experiments were carried out under the following conditions:

- Operating system: Windows 10 64-bit 20H2.
- CPU: AMD Ryzen 7 5800X.
- GPU: NVIDIA GeForce RTX 3090.
- Software versions: Python 3.7.9; Pytorch 1.8.1; Numpy 1.19.2; SciPy 1.5.2; NetworkX 2.5; scikit-learn 0.23.2.
- Training: 6 h to train our model for 30–300 epochs with the learning rate of 1e-04 to 1e-03 and the batch size of 64.

5.1. Datasets

The MVMA model has been validated on several real-world datasets. The details of the datasets are summarized in Table 1.

- ACM[1]: The data is sourced from the ACM database, where each node corresponds to a research paper, with features extracted from the paper's keywords (bag-of-words). The labels denote the paper's academic field: Database, Wireless Communication, or Data Mining. The dataset comprises three types of node relationships: co-authorship, co-conference, and co-keyword usage.
- DBLP[2]: This data comes from the DBLP database. Here, nodes represent authors, with features generated from the keywords used by each author. Node labels indicate the author's area of research: Database, Data Mining, Machine Learning, or Information Retrieval. There are three types of relationships between nodes: co-publishing, co-conference, and co-keyword usage.
- IMDB[3]: Data is extracted from the IMDB website. In this case, nodes represent movies, with features derived from keywords in the movie's plot summary. Node labels indicate the movie's genre: Action, Comedy, or Drama. Three types of relationships between nodes are defined: co-actor, co-director, and co-release year.
- BlogCatalog[4]: This data was sourced from the BlogCatalog website, a social networking site encompassing bloggers and their interconnected relationships. In this context, a node symbolizes a user, and the features of the node are constructed from the keywords within the user profiles. The node label corresponds to one of the six topic categories provided by the authors.
- Flickr[5]: This data was derived from an image and video hosting platform, where users interact through the sharing of photos. A node in this instance represents a user. Node labels correspond to the users' interest groups.
- CoraFull[6]: This is a **larger version** of the well-recognized Cora citation network dataset. Here, a node represents a scientific paper, and the node features are constructed using a bag-of-words approach from paper keywords. Node labels are categorized based on paper topics.
- Chameleon[7]: This dataset, sourced from Wikipedia, contains two page-page networks. Nodes represent web pages, and edges symbolize hyperlinks between these pages. Each node possesses a feature set that corresponds to the informative nouns on the page. The node label indicates the monthly traffic of the respective page.
- Pubmed[8]: This dataset comprises scientific papers. Each paper is depicted as a node in the network, and citation relationships between papers are represented as edges. The labels for each paper are associated with distinct academic fields, and the node features are represented using a bag-of-words approach, encapsulating the content of the papers.

5.2. Baselines

To ascertain the efficacy of our MVMA-GCN model, we set it up against several cutting-edge baseline models. Further, we assessed three distinct variants of MVMA-GCN to confirm the utility of the multi-view and multi-layer attention mechanisms, as well as the autoencoder modules. These comparative evaluations provide a comprehensive understanding of the relative strengths of our proposed model.

[1] https://dl.acm.org/.
[2] https://dblp.uni-trier.de/.
[3] https://www.imdb.com/.
[4] https://github.com/mengzaiqiao/CAN.
[5] https://github.com/mengzaiqiao/CAN.
[6] https://github.com/abojchevski/graph2gauss/.
[7] https://github.com/BUPT-GAMMA/Graph-Structure-Estimation-Neural-Networks.
[8] https://linqs.org/datasets/.

Table 1
The detailed descriptions of the datasets.

Dataset	# Nodes	# Edges	# Classes	# Features
ACM	3025	13 128/1 103 870/1 109 339	3	1870
DBLP	4057	3530/2 498 221/3 386 141	4	334
IMDB	4780	46 617/8121/809 074	3	1732
BlogCatalog	5196	171 743/65 713/97 514	6	8189
Flickr	7575	239 738/64 284/54 135	9	12 047
CoraFull	19 793	65 311/9751/5732	70	8710
Chameleon	2277	15 101/14 685/6315	5	2325
Pubmed	19 717	22 338/10 573/11 427	3	500

Table 2
Model hyperparameters for the reproducibility of our proposed model.

Dataset	Dropout	lr	$Epoch_{max}$	k
ACM	0.07	0.0005	30	7
DBLP	0.06	0.0005	30	8
IMDB	0.07	0.0005	30	6
BlogCatalog	0.06	0.0002	55	7
Flickr	0.07	0.0003	60	8
CoraFull	0.08	0.001	300	7
Chameleon	0.05	0.001	60	5
Pubmed	0.05	0.001	60	5

- Deepwalk[9]: This graph-embedding method utilizes depth-first random walks to acquire contextual information and employs the skip-gram algorithm to learn network representations.
- LINE[10]: A graph-embedding approach that leverages breadth-first random walks to gather contextual information, and learns network representations through first- and second-order similarity.
- GCN[11]: A semi-supervised graph convolutional network model that learns node features by aggregating neighbor information. This model's motivation for the convolutional architecture stems from a localized first-order approximation of spectral graph convolutions.
- kNN-GCN: In this variation, we used a sparse k-nearest neighbor graph computed from the feature matrix as the input graph for GCN, instead of the conventional topology graph.
- GAT[12]: A semi-supervised graph convolutional network model that incorporates the attention mechanism to aggregate node features. Stacking layers in which nodes can attend over their neighborhoods' features allows the model to assign different weights to distinct nodes without necessitating costly matrix operations or prior knowledge of the graph structure.
- DEMO-Net[13]: A degree-specific graph neural network for node classification. This model introduces a novel graph-level pooling/readout scheme for learning graph representations, which lies provably within a degree-specific Hilbert kernel space.
- MixHop[14]: A GCN-based method that integrates feature representations of higher-order neighbors into a single graph convolution layer. The model is capable of learning a general class of neighborhood mixing relationships, including difference operators, by continuously mixing feature representations of neighbors at varying distances.
- AM-GCN[15]: AM-GCN is an innovative model designed for efficient node classification and representation learning in graph-structured data. By incorporating attention mechanisms and leveraging information from multiple views, AMGCN effectively captures both node features and the underlying graph structure.
- MVMA_{SV}: A variant of MVMA-GCN where the multi-view module is deleted; we only used single view as the input.
- MVMA_{SA}: A variant of MVMA-GCN where the multi-layer attention module is deleted, and only a single layer of attention is retained.
- MVMA_{EN}: A variant of MVMA-GCN where the autoencoder module is deleted.
- MVMA-GCN: The proposed semi-supervised graph convolutional network includes a multi-view module, a multi-layer attention module, and an autoencoder module.

[9] https://github.com/phanein/deepwalk.
[10] https://github.com/tangjianpku/LINE.
[11] https://github.com/tkipf/pygcn.
[12] https://github.com/Diego999/pyGAT/.
[13] https://github.com/jwu4sml/DEMO-Net.
[14] https://github.com/benedekrozemberczki/MixHop-and-N-GCN.
[15] https://github.com/zhumeiqiBUPT/AM-GCN.

5.3. Parameters setting

To undertake a thorough assessment of the MVMA-GCN model, we initialized the parameters randomly and employed Adam optimization. Training sets were established with three different label rates, entailing 20, 40, and 60 labeled nodes for each respective category. For the sake of consistency, all baseline comparisons were initialized using parameters recommended in their original publications.

For our model, we allocated 768 and 512 dimensions respectively to each GCN's hidden layer. A comprehensive summary of the model's hyperparameters is provided in Table 2. This meticulous setup ensures a rigorous and fair evaluation of our model's performance against the state-of-the-art alternatives. Our code is based on the AM-GCN.[16]

5.4. Node classification

The node classification results are shown in Table 3, which contains the node classification results for the three datasets. In the table, L/C indicates the number of nodes with labels in each class.

For each evaluation metric, MVMA achieved the best results in almost all datasets. In particular, for accuracy, the model achieved a significant improvement of 2.5% and 2% for the classification task on ACM and DBLP, respectively. In F1 scores, MVMA achieved an average improvement of 1.5%. This means MVMA-GCN, unlike MixHop or AM-GCN, successfully merged the information between the nodes contained in multi-view.

The model performed better than GCN and GAT on all datasets, which shows that the multi-view fusion mechanism is effective. We observed that our model results are better than the most robust baseline AM-GCN (usually a 2% improvement), which verifies that our model can more accurately capture the relationship between nodes. Additionally, our model uses a multi-layer attention mechanism to reveal the hidden associations between nodes, which provides more accurate training guidance for node classification.

The node classification results of the variants are shown in Table 4. MVMA_{SA} only uses a single layer of attention, which cannot assign the best weights to different nodes and views. So, MVMA usually achieved better node classification results than MVMA_{SA}. MVMA achieved much better results in the IMDB dataset than MVMA_{SA}. Many movies are usually not related in the IMDB dataset except for the release year, which causes large amounts of noise in the co-year view. However, the co-year view cannot be removed because the release year can be used to understand the audience's preferences and the trend in movies, which also reflects the importance of multi-layered attention.

MVMA achieved more accurate node classification results than MVMA_{EN}. Compared with MVMA_{EN}, which does not use the autoencoder module, MVMA relies on the structural information between the data to learn the representation of nodes and the representation between the data through the kNN graph. Therefore, MVMA_{EN} performance decreases when the data structure is unclear or has errors. In addition, the autoencoder can overcome the problem due to GCN's limited ability to learn structured information, thus improving the node classification results.

[16] https://github.com/zhumeiqiBUPT/AM-GCN.

Table 3
Node classification results on different baselines.(%) (bold = best).

Dataset	Metrics	L/C	DeepWalk	LINE	GCN	*k*NN-GCN	GAT	DEMO-Net	MixHop	AM-GCN	MVMA-GCN
ACM	ACC	20	62.69	41.28	87.80	78.52	87.36	84.48	81.08	90.40	**92.74**
		40	63.00	45.83	81.64	89.06	88.60	85.70	82.34	90.76	**92.83**
		60	67.03	50.41	85.43	90.54	90.40	86.55	83.09	91.42	**93.75**
	F1	20	62.11	40.12	87.82	78.14	87.44	84.16	81.40	90.43	**92.62**
		40	61.88	45.79	89.00	81.53	88.55	84.83	81.13	90.66	**92.75**
		60	66.99	49.92	90.49	81.95	90.39	84.05	82.24	91.36	**93.66**
DBLP	ACC	20	79.37	86.89	90.71	90.42	89.96	90.16	90.64	90.91	**92.33**
		40	82.73	86.94	91.01	91.37	90.14	90.82	91.04	91.32	**92.54**
		60	83.89	87.25	91.62	91.46	90.84	91.32	91.44	91.72	**92.82**
	F1	20	77.43	85.46	90.79	90.61	89.97	90.53	90.73	91.02	**91.17**
		40	81.02	85.57	91.48	91.07	90.20	91.03	91.07	91.56	**91.59**
		60	83.46	87.21	91.89	91.81	90.80	91.48	91.39	91.78	**92.43**
IMDB	ACC	20	40.72	42.68	49.78	50.24	55.28	41.16	55.10	58.35	**59.50**
		40	45.19	43.69	51.71	50.44	55.91	44.22	55.94	59.56	**60.42**
		60	48.13	47.19	52.29	51.79	56.44	45.11	56.68	59.80	**60.65**
	F1	20	46.38	29.86	45.73	46.79	49.44	45.65	55.28	50.29	**51.54**
		40	49.99	30.35	48.01	48.96	50.64	48.24	55.47	51.44	**52.90**
		60	50.70	30.75	49.15	49.62	51.90	49.09	55.64	53.75	**54.23**
BlogCatalog	ACC	20	38.67	58.75	69.84	75.49	64.08	54.19	65.46	81.98	**82.03**
		40	50.80	61.12	71.28	80.84	67.40	63.47	71.66	84.94	**85.26**
		60	55.02	64.53	72.66	82.46	69.95	76.81	77.44	87.30	**88.09**
	F1	20	34.96	57.75	68.73	72.53	63.38	52.79	64.89	81.36	**82.55**
		40	48.61	60.72	70.71	80.16	66.39	63.09	70.84	84.32	**85.14**
		60	53.56	63.81	71.80	81.90	69.08	76.73	76.38	86.94	**89.03**
Flickr	ACC	20	24.33	33.25	41.42	69.28	38.52	34.89	39.56	75.26	**75.69**
		40	28.79	37.67	45.48	75.08	38.44	46.57	55.19	80.06	**82.24**
		60	30.10	38.54	47.96	77.94	38.96	57.30	64.96	82.10	**83.94**
	F1	20	21.33	31.19	39.95	70.33	37.00	33.53	40.13	74.63	**75.69**
		40	26.90	37.12	43.27	75.40	36.94	45.23	56.25	79.36	**81.02**
		60	27.28	37.77	46.58	77.97	37.35	56.49	65.73	81.81	**82.50**
CoraFull	ACC	20	29.33	17.78	56.68	41.68	58.44	54.50	47.74	58.90	**59.02**
		40	36.23	25.01	60.60	44.80	62.98	60.28	57.20	63.62	**64.59**
		60	40.60	29.65	62.00	46.68	64.38	61.58	60.18	65.36	**67.81**
	F1	20	28.05	18.24	52.48	37.15	54.44	50.44	45.07	54.74	**56.97**
		40	33.29	25.43	55.57	40.42	58.30	56.26	53.55	59.19	**60.78**
		60	37.95	30.87	56.24	43.22	59.61	57.26	56.40	61.32	**63.72**
Chameleon	ACC	20	30.51	25.72	39.53	33.56	45.94	43.21	40.25	51.24	**51.67**
		40	31.40	26.33	44.24	34.75	51.79	47.65	43.64	53.08	**53.19**
		60	31.91	28.45	49.18	34.92	53.77	55.52	46.33	53.29	**53.42**
	F1	20	30.24	25.61	40.52	33.41	45.25	42.98	41.84	**50.64**	50.42
		40	30.87	26.12	43.84	34.79	50.72	46.53	43.52	52.47	**52.75**
		60	31.22	26.89	48.77	34.88	53.47	56.72	45.39	52.93	**52.99**
Pubmed	ACC	20	60.25	58.86	79.06	79.62	79.27	80.54	80.89	**81.65**	79.49
		40	60.78	59.34	79.76	80.24	80.15	81.54	81.92	82.01	**82.41**
		60	62.31	59.96	80.48	80.95	80.94	81.69	82.33	82.35	**82.45**
	F1	20	60.14	58.43	79.15	79.14	79.02	80.34	80.52	**81.46**	79.24
		40	60.89	59.61	79.25	80.17	79.58	80.96	80.98	81.87	**82.29**
		60	61.34	59.64	79.68	80.39	80.12	81.45	81.23	82.24	**82.41**

5.5. Model analysis

Effectiveness of the Multi-View Module. We used three datasets to verify the effectiveness of multiple views. In each dataset, each view was used as the model's input. Then the results were compared with the results of inputting all views simultaneously, as shown in Fig. 3.

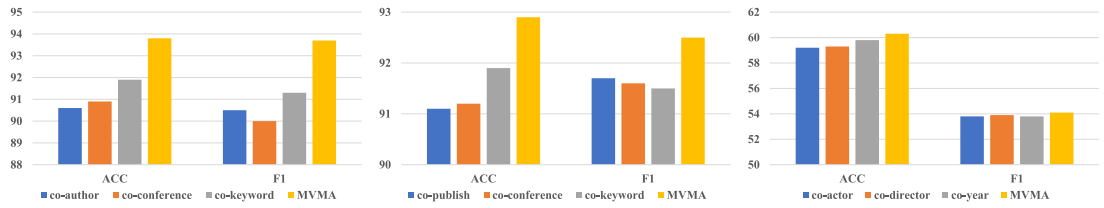
Taking the ACM dataset as an example, the input contains three views: co-conference, co-keywords, and co-author. It can be seen that different views had different effects on the results, but overall, the accuracy and F1 score of using a single view as the input are lower than when using multi-views as the input.

Various views encompass varying degrees of information density. Views with a higher information content tend to exert a more substantial positive influence on the outcomes. Consequently, utilizing a single view as the sole input may lead to poor prediction results. Nonetheless, it is crucial for the model to retain these low-density information links, as they continue to contribute positively to the prediction outcomes. As a result, it is essential for a multi-view model to incorporate as many views as feasible to optimize performance.

Effectiveness of the Multi-Layer Attention Module. Another feature of our model is the multi-layer attention mechanisms, that is, attention mechanisms within a single view and attention mechanisms between multiple views. Here, we analyze the effects of different levels of attention, as shown in Fig. 4.

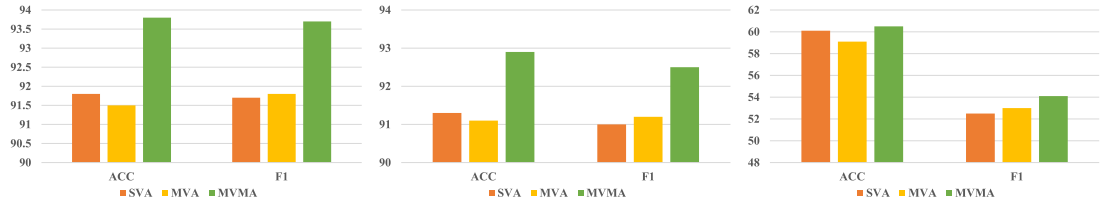
First, we only used attention within a single view: SVA. Next, we only used the attention between multi-views: MVA. Finally, the complete multi-layer attention mechanism was used. It can be seen that SVA and MVA are inferior to MVMA in terms of node classification accuracy and F1 score because the influence between nodes is different, and different views have different influences on the results. If all views and all nodes are assigned the same weight, the model's performance decreases.

For example, in the IMDB dataset, some actors prefer collaborating with fixed directors and actors to shoot multiple genres of movies. Conversely, some actors prefer to make films in the same genre but



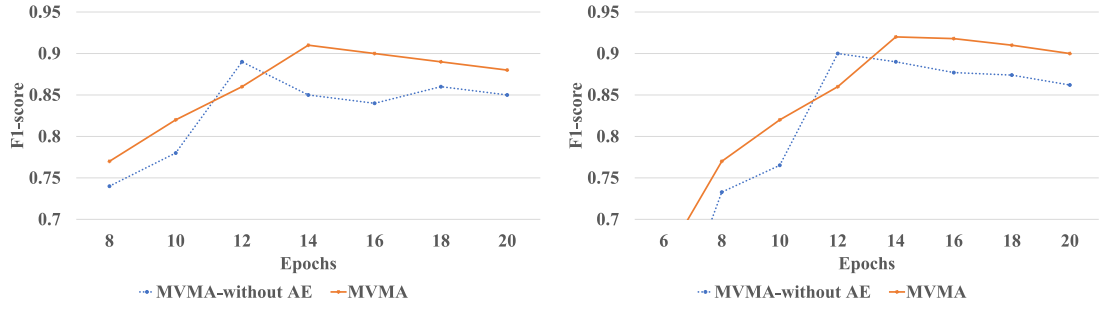
(a) ACM (b) DBLP (c) IMDB

Fig. 3. Effectiveness of the multi-view module on three datasets. A view that contains more information has a more significant positive impact on the results.



(a) ACM (b) DBLP (c) IMDB

Fig. 4. Effectiveness of the multi-layer attention module. When the inputs contain multiple relationships, the model can learn more information, and the result is more accurate.



(a) ACM (b) DBLP

Fig. 5. Effectiveness of the autoencoder module.

Table 4
Node classification results of the variants on all dataset (%).

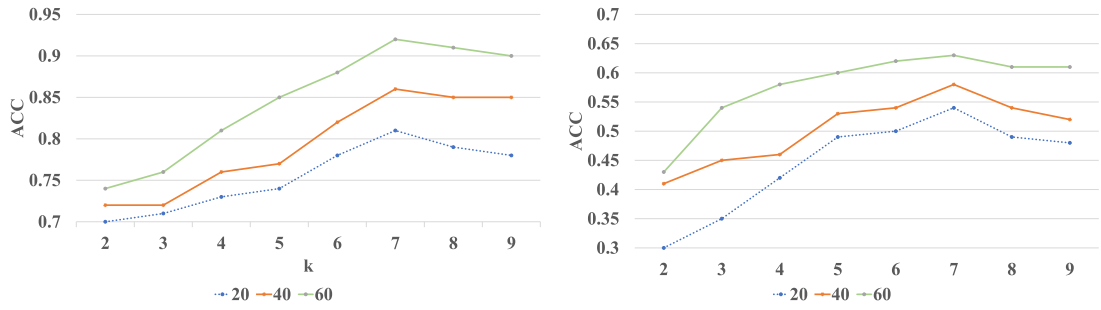
Datasets	Metrics	$MVMA_{SV}$	$MVMA_{EN}$	$MVMA_{SA}$	MVMA
ACM	ACC	92.73	92.22	91.95	93.75
	F1	92.48	92.23	92.18	93.66
DBLP	ACC	92.00	92.10	92.40	92.82
	F1	92.18	91.98	92.05	92.43
IMDB	ACC	60.50	59.90	60.17	60.65
	F1	54.00	53.82	53.94	54.23
BlogCatalog	ACC	87.82	87.65	87.69	88.09
	F1	87.52	88.06	87.44	89.03
Flickr	ACC	82.48	82.71	82.77	83.94
	F1	81.90	82.34	82.19	82.50
CoraFull	ACC	66.02	66.64	66.47	67.81
	F1	62.72	62.25	62.12	63.72
Chameleon	ACC	51.98	52.76	52.47	53.42
	F1	51.67	51.65	52.76	52.99
Pubmed	ACC	81.24	82.01	82.12	82.45
	F1	81.64	81.72	81.96	82.41

tend to collaborate with different directors and actors. The examples illustrate that it is essential to consider the impact of different nodes and the impact of different views. This verifies the effectiveness of the multi-layer attention module.

Effectiveness of the Autoencoder Module. Fig. 5 illustrates the necessity of using an autoencoder. The horizontal coordinates are the numbers of training epochs, and the vertical coordinates are the accuracy and F1 scores. Experiments were performed on each dataset separately, and the figure shows the model output with and without the autoencoder.

By adding the autoencoder, the hidden layer representation does not tend to converge prematurely to the same value. After using the autoencoder module, the model's optimal result changed from around the 9th epoch to around the 14th. Additionally, the autoencoder ensures that the model can learn the nodes' structural and nodes' relationships, which verifies the effectiveness of the autoencoder module.

Effectiveness of k-Nearest Neighbor. Fig. 6 illustrates the impact of the top k neighborhoods in the k-NN graph. The horizontal coordinates are the number of neighbors, and the vertical coordinates are the accuracy. The experiments were performed on the BlogCatalog and Flick datasets separately, and the figure shows that the accuracies increased first and then started to decrease. We believe this result occurred because a larger k may cause a more significant number of noisy edges.



(a) BlogCatalog (b) Flickr

Fig. 6. Effectiveness of k-nearest neighbor. Analysis the model's accuracy with the different numbers of neighbors.

6. Conclusions and future direction

In this paper, we present MVMA-GCN, a multi-view-based model for semi-supervised node classification tasks. To aggregate neighbor features of each node in a single view, we propose a single-view convolution. To capture the rich information in multiple relationships between nodes, we introduce a multi-layer attention mechanism that learns the influence between different neighboring nodes and the influence between different relationship graphs. Additionally, we maximize the difference between different relationship graphs by calculating the Hilbert–Schmidt independence criterion. We also redesigned an autoencoder module to improve node classification accuracy. Our extensive experiments on several benchmark datasets demonstrate the superior performance of the proposed model, and we verified the effectiveness of different modules through ablation experiments.

In future research, a more challenging task is to propose a new large-scale node classification dataset composed of yearly snapshots of nodes and study how the node evolutionary scenario impacts performance. Since nodes change over time, and previously non-existent nodes appear in the real world, this task is particularly relevant. Moreover, investigating the use of directed graphs as input could be an interesting area of exploration for future research.

CRediT authorship contribution statement

Pengyu Zhang: Writing – review & editing, Software. **Yong Zhang:** Conceptualization, Validation. **Jingcheng Wang:** Data curation, Writing. **Baocai Yin:** Supervision.

Declaration of competing interest

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

Funding

This work was jointly sponsored by the National Natural Science Foundation of China under grants 62072015, U19B2039, and U1811463.

Code availability

All code utilized in the analyses during this study has been made publicly available in the repository linked in the paper's abstract.

Data availability

Data will be made available on request

References

Aburahmah, L., AlRawi, H., Izz, Y., Syed, L., 2016. Online social gaming and social networking sites. Procedia Comput. Sci. 82, 72–79.
Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., Cui, P., 2020. Structural deep clustering network. In: Proceedings of the Web Conference 2020. pp. 1400–1410.
Cao, S., Lu, W., Xu, Q., 2016. Deep neural networks for learning graph representations. In: Proceedings of the AAAI Conference on Artificial Intelligence.
Cheung, M., Moura, J.M., 2020. Graph neural networks for covid-19 drug discovery. In: 2020 IEEE International Conference on Big Data (Big Data). IEEE, pp. 5646–5648.
Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP.
Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural Inf. Process. Syst. 29, 3844–3852.
Fan, S., Wang, X., Shi, C., Lu, E., Lin, K., Wang, B., 2020. One2multi graph autoencoder for multi-view graph clustering. In: Proceedings of the Web Conference 2020. pp. 3070–3076.
Gao, H., Ji, S., 2019. Graph u-nets. In: International Conference on Machine Learning. PMLR, pp. 2083–2092.
Gao, H., Pei, J., Huang, H., 2019. Conditional random field enhanced graph convolutional neural networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 276–284.
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.
Hamilton, W.L., Ying, R., Leskovec, J., 2017. Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035.
Hoang, N., Maehara, T., 2019. Revisiting graph neural networks: All we have is low-pass filters. Stat 1050, 26.
Li, M., Zhang, Y., Li, X., Cai, L., Yin, B., 2022. Multi-view hypergraph neural networks for student academic performance prediction. Eng. Appl. Artif. Intell. 114, 105174.
Luong, M.-T., Pham, H., Manning, C.D., 2015. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421.
Ma, Y., Wang, S., Aggarwal, C.C., Tang, J., 2019. Graph convolutional networks with eigenpooling. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 723–731.
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, R., Biswas, M., Sen, S., de Souza Júnior, L.A., Papa, J.P., Sarkar, R., 2022a. A fuzzy distance-based ensemble of deep models for cervical cancer detection. Comput. Methods Programs Biomed. 219, 106776. http://dx.doi.org/10.1016/j.cmpb.2022.106776, URL https://www.sciencedirect.com/science/article/pii/S0169260722001626.
Pramanik, R., Dey, S., Malakar, S., Mirjalili, S., Sarkar, R., 2022b. TOPSIS aided ensemble of CNN models for screening COVID-19 in chest X-ray images. Sci. Rep. 12 (1), 1–19.
Pramanik, R., Sarkar, S., Sarkar, R., 2022c. An adaptive and altruistic PSO-based deep feature selection method for Pneumonia detection from chest X-rays. Appl. Soft Comput. 128, 109464.
Qu, M., Bengio, Y., Tang, J., 2019. Gmnn: Graph markov neural networks. In: International Conference on Machine Learning. PMLR, pp. 5241–5250.
Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G., 2008. The graph neural network model. IEEE Trans. Neural Netw. 20 (1), 61–80.
Syed, L., Jabeen, S., Manimala, S., Alsaeedi, A., 2019. Smart healthcare framework for ambient assisted living using IoMT and big data analytics techniques. Future Gener. Comput. Syst. 101, 136–151.

Tang, Z., Qiao, Z., Hong, X., Wang, Y., Dharejo, F.A., Zhou, Y., Du, Y., 2021. Data augmentation for graph convolutional network on semi-supervised classification. arXiv preprint arXiv:2106.08848.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q., 2015. Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web. pp. 1067–1077.

Tong, Y., Zhang, X., Cao, C.C., Chen, L., 2014. Efficient probabilistic supergraph search over large uncertain graphs. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 809–818.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S., 2017. Community preserving network embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence.

Wang, D., Cui, P., Zhu, W., 2016. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1225–1234.

Wang, Q., Ding, Z., Tao, Z., Gao, Q., Fu, Y., 2021. Generative partial multi-view clustering with adaptive fusion and cycle consistency. IEEE Trans. Image Process. 30, 1771–1783.

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., Pei, J., 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1243–1253.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K., 2019. Simplifying graph convolutional networks. In: International Conference on Machine Learning. PMLR, pp. 6861–6871.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention. In: International Conference on Machine Learning. PMLR, pp. 2048–2057.

Xu, K., Hu, W., Leskovec, J., Jegelka, S., 2018. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Xue, J., Yabe, T., Tsubouchi, K., Ma, J., Ukkusuri, S., 2022. Multiwave covid-19 prediction from social awareness using web search and mobility data. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 4279–4289.

Yin, M., Huang, W., Gao, J., 2020. Shared generative latent representation learning for multi-view clustering. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 6688–6695.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J., 2018. Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 974–983.

Zhang, M., Cui, Z., Neumann, M., Chen, Y., 2018. An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI Conference on Artificial Intelligence.