Deep Multi-Attributed-View Graph Representation Learning

By

Xiaoxiao Ma

A thesis submitted to Macquarie University

for the degree of Master of Research

Department of Computing

December 2020

MACQUARIE University

SYDNEY·AUSTRALIA

Examiner's Copy

Typeset in LaTeX 2_ε.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) _____ Date: 25/12/2020

Xiaoxiao Ma

Dedication

This thesis is dedicated to my parents, my wife and my son for their endless love, support and encouragement in the hard time of Covid-19.

Acknowledgements

I would like to sincerely thank the people who have helped, encouraged, and supported me through the year spent on this work.

First of all, I would like to express my sincere appreciation to my supervisor Prof Jian Yang and my associate supervisors Dr Jia Wu and Dr Shan Xue for their kindness, patience, and detailed guidance. All of them are good listeners and advisors.

I would also like to thank Dr Weiliang Zhao and Dr Young Lee for their constant support and suggestions.

Additionally, I would like to acknowledge Macquarie University, the Faculty of Science and Engineering and the Department of Computing for supporting me with a scholarship.

Last but not least, I would like to thank my family. My parents, my wife and my son have always been with me. With their love, support, and encouragement, I was able to persevere.

List of Publications

- **Xiaoxiao Ma**, Jia Wu, Shan Xue, and Jian Yang, *Deep Multi-Attributed-View Graph Representation Learning*. (In Progress)

Abstract

Graph representation learning (Graph embedding) aims at encoding a graph into a lower-dimensional feature space. Deep representation learning on the attributed graph utilizes both the graph structure and the graph attributes, which has shown significance in graph learning. Rich information in a graph can be expressed by attributes from different perspectives, which is employed as *attributed views* in the research field. Taking the social networks as an example, a user's profiles and its posted contents can be regarded as two separate attributed views. The majority of existing attributed graph representation learning methods focus on single attributed view, which inherently limits the capability of the techniques to multi-attributed-view graphs. In this work, we present a novel model for generating representations with graphs containing multiple attributed views. The model, deep Multi-attributed-view graph Convolutional Autoencoder model (MagCAE), is built based on an unsupervised autoencoder framework with graph convolutional neural network layers. In addition, a novel multi-attributed-view proximity measurement and similarity loss function are proposed to further improve the effectiveness of generated embeddings. Extensive experiments and comparisons with 10 baselines on 5 real-world multi-attributed-view graphs demonstrate the superiority of MagCAE for link prediciton with respect to average precision (AP) and area under the ROC curve (AUC), and node classification with respect to Micro and Macro F1-scores.

Contents

Statement of Originality iii

Dedication v

Acknowledgements vii

List of Publications ix

Abstract xi

List of Figures xvii

List of Tables xix

List of Symbols xxi

1 Introduction 1

2 Literature Review 7

2.1 Graph Representation Learning 7

2.2 Plain Graph Representation Learning 8

2.3 Attributed Graph Representation Learning 10

3 Preliminaries 13

3.1 Autoencoder 13

3.2 Plain Graph 14
3.3 Attributed Graph 15
3.4 Pairwise-node Proximity 15

4 Multi-attributed-view Graph Convolutional Autoencoder 17
4.1 The Overall Framework 18
4.2 Multi-attributed-view Graph Convolutional Encoder 19
4.3 Multi-attributed-view Graph Decoder 20
4.4 Three Levels of Proximity: Single-view, Multi-view, and Embedding 20
4.4.1 Single-attributed-view Proximity (Attribute Proximity) 20
4.4.2 Multi-attributed-view Proximity 21
4.4.3 Embedding Proximity 21
4.5 The Loss Function 22
4.6 Optimisation 23
4.7 Algorithm and Complexity Analysis 23

5 Experiments 27
5.1 Datasets 27
5.2 Baselines 29
5.3 Experiment and Parameter Settings 30
5.3.1 Training, Validation, and Test set 30
5.3.2 Experiment Settings 30
5.3.3 Evaluation Metrics 31
5.4 Link Prediction Results 31
5.5 Node Classification Results 34
5.6 Discussion 37
5.6.1 MagCAE's Effectiveness 37
5.6.2 Convergence Analysis 37
5.6.3 Robustness to Incomplete Graphs 37
5.7 Parameter Analysis 39
5.7.1 Hidden Layer Dimensionality 39

5.7.2 λ Coefficient Sensitivity 39
5.8 Visualization of the Node Embeddings 40

6 Conclusion and Future Works **41**

References **43**

List of Figures

1.1 Attributed graphs. 3

2.1 The process of plain graph representation learning. 9
2.2 Different random walk directions. 10
2.3 The process of attributed graph representation learning. 11

3.1 The framework of Autoencoder. 14

4.1 The framework of MagCAE: a convolutional autoencoder for multi-attributed-view graphs. 18
4.2 Single-attributed-view Encoder. 19

5.1 The training curve of MagCAE. 38
5.2 Training ratios v.s. link prediction performance. 38
5.3 p Embedding dimensions v.s. link prediction performance. 39
5.4 λ Coefficient v.s. link prediction performance. 40
5.5 Node visualization results of different graph embedding models on Cora. 40

List of Tables

5.1 Dataset statistics 29
5.2 Link prediction results on citation networks in scenario A 32
5.3 Link prediction results on citation networks in scenario B 33
5.4 Link prediction results on customer review networks in scenario A 33
5.5 Link prediction results on customer review networks in scenario B 34
5.6 Node classification results on citation networks in scenario A 35
5.7 Node classification results on citation networks in scenario B 35
5.8 Node classification results on customer review networks in scenario A . . 36
5.9 Node classification results on customer review networks in scenario B . . 36

List of Symbols

G_M ..... a multi-attributed-view graph

V ..... the node set

v_i ..... the i-th node

E ..... the edge set

$e_{i,j}$ ..... an edge between v_i and v_j

n ..... the number of nodes

A ..... the adjacency matrix

$a_{i,j}$ ..... the adjacency value

$\mathscr{X}$ ..... the multi-attributed-view set

m ..... the number of attributed views

X_ξ ..... attributes in the ξ-th view

k_ξ ..... the number of attributes in the ξ-th view

$\mathbf{x}_i$ ..... a vector of attribute values of v_i

$\mathbf{x}_i^{a,\xi}$ ..... a vector of attribute values of v_i in attributed view ξ

S_a ..... node-pairwise attribute proximity

S^a_ξ ..... node-pairwise attribute proximity in attributed view ξ

$s^a_{i,j}$ ..... v_i and v_j-pairwise attribute proximity

$s^{a,\xi}_{i,j}$ ..... v_i and v_j-pairwise attribute proximity in attributed view ξ

S_v ..... node-pairwise multi-attributed-view proximity

$s^v_{i,j}$ ..... v_i and v_j-pairwise multi-attributed-view proximity

S_e ..... node-pairwise embedding proximity

$s^e_{i,j}$ ..... v_i and v_j-pairwise embedding proximity

d ..... the dimension of embeddings

W ..... weight matrix in GCN

D ..... the node degree matrix

A' ..... reconstructed graph

Z ..... node embedding

$\tilde{A}$ ..... normalized symmetric adjacency matrix

L ..... autoencoder loss

L_{res} ..... graph reconstruction loss

L_{sim} ..... node pairwise similarity loss

1

Introduction

Many of the real-world networks that are valuable to contemporary data analytics are represented as graphs [1–3]. For instance, a social network can be presented as a graph where the users are the nodes and the relationships between them are the edges. Accordingly, this makes it possible to visualise communities, identify key entities and patterns, and statistically analyse metrics such as distance, centrality, and co-occurrence. As humans, we find it easier to understand complex relationships when we can digest the information visually from the graph. For machines, graphs offer unique and efficient mathematical constructs for processing data. Because of the efficiency of graphs in both storing and representing network data, graph data mining has become one of the hottest topics in data science in recent years and researchers have been producing promising results in a huge range of downstream applications, including social recommendation, anomaly detection, cybersecurity, and more.

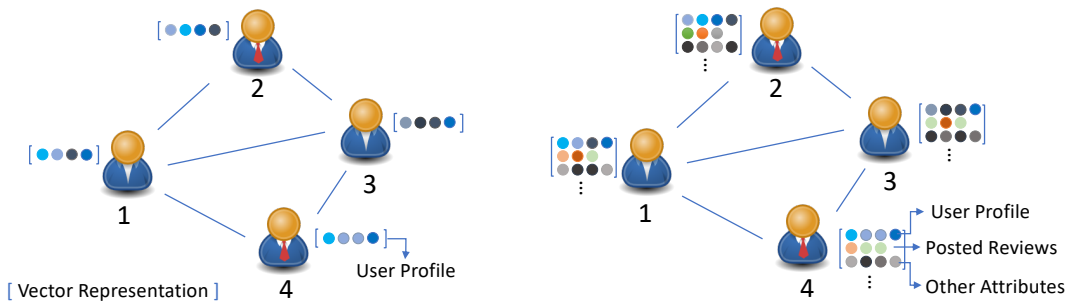
However, the complexity of many networks today is beyond the capability of traditional machine learning techniques. To store the graph structure, the most convenient and widely-adopted method is using one-hot vectors. Given a real-world network with millions of nodes and edges, the one-hot vector representation would have a large number of dimensions and it is impossible to use it directly for data mining under computational constraints. This motivates the work of graph representation learning to encode graphs into a lower-dimensional space.

The linking nature of graphs also makes them particularly ill-suited to parallel computing. In traditional machine/deep learning research fields, the data items are assumed to follow an independent and identical distribution (*i.e., i.i.d.*), which means each data item can be processed independently and in parallel. By contrast, nodes are linked to each other in graphs and hence not independent and identically distributed. These dependencies render parallel computing with graph data extremely difficult.

There have been numerous attempts to tackle these challenges with new types of data representations and new embedding methods specifically designed for graphs. Those works, known as graph representation learning/graph embedding/network embedding, aim to represent a graph in a lower-dimensional space while retaining as much valuable information as possible. With the lower-dimensional representation, traditional machine learning and deep learning can then be applied to perform downstream graph analysis, such as link prediction and node classification [4, 5].

Among the most successful embedding methods are the ones that encode the graph structure into a low-dimensional latent feature space [6, 7] using either deep learning models [8] or graph neural networks [1, 9]. Early graph representation learning methods, such as Laplacian eigenmaps [10], LPP [11], DeepWalk [12], Node2Vec [13], and LINE [14] take the graph structure as input and output node embeddings in return.

However, a graph's structure is not the only valuable information it can contain. The attributes of the nodes and edges, *e.g.,* a user's hobbies or the length of time two users have been friends, are very useful information for graph representations [7, 15–18]. To explore this knowledge, attributed graph embedding models take the graph structure, the nodes' attributes as well as other side information as inputs. For example, TADW [15]



(a) Single-attributed-view graph (b) Multi-attributed-view graph

(a) Each node in a single-attributed-view graph is represented by attributes from a unique perspective – for example, a user profile in an online social network. (b) By contrast, in a multi-attributed-view graph, each node is represented from a number of different perspectives – for example, a user profile and a collection of posted reviews, resulting in graphs with multiple attributed views.

FIGURE 1.1: Attributed graphs.

which is based on DeepWalk, further considers contextual information associated with each node to learn node embeddings, while ASNE [7] takes two types of side information, *i.e.,* structural proximity and attribute proximity, additionally into account to train a deep neural network. These two proximities are quantified based on the graph structure and nodes' attributes, respectively. Graph convolutional network (GCN) models [19] and other state-of-the-art methods [20–22] also aggregate neighbourhood information into the targeting node's representation via graph neural networks and have achieved more effective representations to support graph analytics.

"Attributed views" is a further aspect of graph mining. An attributed view is a collection of features that form a way of viewing a network, *i.e.,* a perspective [1, 23–26]. For instance, the features associated with user profiles is one perspective, while the content they post is another. The attributes associated with different views are usually collected from different sources. Using Twitter as an example, user profile information would be gathered from the registration information, while their historical tweets would be taken from published posts as shown in Figure 1.1(b).

Although multiple attributed views describe the graph attribute information from different perspectives and provide more comprehensive information to graph data mining, the graph representation learning techniques discussed above are only designed to work

with graphs containing a single attributed view (shown in Figure 1.1(a)). In order to learn graph representations from multiple attributed views, one straightforward approach is to concatenate different attributed views into a single attributed view and then apply one of the single-view techniques. However, this approach implicitly assumes that each attributed view has the same significance on the learned representations and downstream graph analysis tasks, which is not always the case. As a concrete example, in online social networks, both users' demographic information and historical posts (those are two distinctive attributed views) influence the links formed between them, but, in most cases, historical posts are more influential than users' demographic information because people are more interested in the posted content.

To this end, graph representation learning with multiple attributed views have not been well explored yet. The known challenges are thus: different attributed views have different impacts on the learned representation. Consequently, to create a truly effective model, each view must be weighted accordingly. Besides, this weighting must be a trainable parameter that can adjust automatically during the process to fit different networks and application scenarios. Moreover, other side information, *e.g.,* node attribute proximity, that could benefit graph representation learning needs to be explored and utilized to improve the learning result.

To tackle these challenges, this work proposes a novel deep learning model called **M**ulti-**a**ttributed-view **g**raph **C**onvolutional **A**uto**E**ncoder, or MagCAE for short. The model inherits the advantages of the autoencoder framework and graph convolutional neural network (GCN) to learn graph representations in an unsupervised manner. Moreover, MagCAE explores pairwise-node proximity information and proposes a specially-designed pairwise similarity loss function to boost the performance of the generated graph representations.

The main contributions of this work include:

- We formulate the problem of multi-attributed-view graph representation learning by distinguishing various attributed views on node attributes. To the best of our knowledge, it is the first time to consider the impact of different attributed views on the learned graph representations and jointly optimise the multi-attributed-view

graph representation learning as well as the weights of attributed views through an effective deep unsupervised graph neural network.

- We propose a novel multi-attributed-view proximity measurement for representation learning. The measure explores node attribute similarity across different attributed views and provides meaningful side information to improve the effectiveness of the learned representations.

- We conduct extensive experiments to demonstrate the effectiveness of the proposed model by comparison with 10 baselines on 5 real-world multi-attributed-view graphs and show the superiority of MagCAE on link prediction with respect to average precision (AP) and area under the ROC curve (AUC), and node classification with respect to Micro and Macro F1-scores.

Hereinafter, graph embedding and graph representation learning are used interchangeably. The rest of this thesis is organized as follows. Chapter 2 reviews existing works on graph representation learning and autoencoder. Chapter 3 presents the basic concepts and notations. Chapter 4 provides the details of MagCAE and its key components. Chapter 5 presents the experiments, performance evaluations and baseline comparisons. Chapter 6 concludes this work with a brief summary of MagCAE and our intentions for future work.

2

Literature Review

This chapter begins with an introduction to graph representation learning in Section 2.1, followed by a brief review of the state-of-the-art graph representation learning methods in Sections 2.2 and 2.3. Section 2.2 covers plain graphs. Section 2.3 focusses on attributed graphs.

2.1 Graph Representation Learning

As mentioned in the prior chapter, the conventional graph representation has introduced significant challenges to graph data mining and has become a bottleneck in graph analysis. To alleviate those problems, graph representation learning has been committed to embedding network nodes into a low-dimensional feature space/embedding space where nodes are distributed independently, and their dependencies are captured by their distances in

the embedding space instead of using explicit edges [1].

One of the benefits of graph representation learning is that the low-dimensional node representations/node embeddings reduce the computation cost of graph data significantly. Downstream graph analysing methods will take less space to store graph data and consume less on computing when graphs are represented by low-dimensional vectors. Graph representation learning also makes traditional machine learning and deep learning techniques convenient for solving graph problems. Taken the lower-dimensional node representations as inputs, powerful machine learning techniques such as support vector machine (SVM) and k-nearest neighbours can be trained to identify node clusters, classify nodes, and predict links in a reasonable time.

In order to generate more effective node embeddings to support graph analytics, substantial efforts have been put into exploring different types of information that are contained in real-world networks. Some of the existing works generate node embeddings using the graph structure information, while more recent works also take into account nodes' attributes and other side information to meet the goal. From this perspective, existing techniques can be categorized into plain graph representation learning methods and attributed graph representation learning methods depending on the types of information leveraged for representation learning.

2.2 Plain Graph Representation Learning

Plain graph representation learning methods generate node embeddings by leveraging a graph's topological information. These methods generally take a graph's adjacency matrix or degree matrix as the input, and the output embeddings for nodes usually preserve the structural information. The general process of existing plain graph representation learning techniques is shown in Figure 2.1.

Earlier graph embedding methods, such as Laplacian Eigenmaps [10] and Locality preserving projection (LPP) [11], explore a graph's geometric structure and attempt to discover and represent the graph by the most structure-related features. Node embeddings

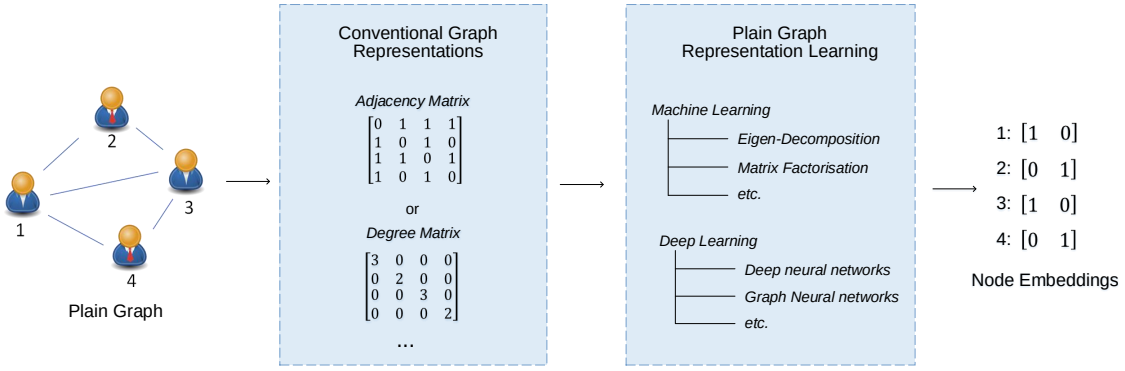


FIGURE 2.1: The process of plain graph representation learning.

generated by these methods can preserve useful local structure information and show promising performance on graph analysis tasks, such as node clustering. However, the price of these benefits is the high computation cost associated with computing eigenvalues and eigenvectors. As such, these methods are only suitable for small, relatively simple networks; their scalability to large networks is constrained by computational complexity.

To overcome the high computational overhead of eigen-decomposition operation and motivated by the success of deep learning in natural language processing, DeepWalk [12] generalises the idea of contextual word embeddings to graphs. The basic ideas of DeepWalk are to treat collections of nodes as sentences and embed a node in a vector given its neighbouring nodes. One of the keys to DeepWalk is that the generated node sentences should preserve the dependencies between nodes, and the random walk provides a feasible and convenient solution. In practice, DeepWalk generates node sentences through independent random walks that start from different nodes. Each node sentence can be regarded as a fixed-length depth-first-search on the graph, and it can effectively preserve the network relationships. Further, truncated random walks provide an avenue to parallel computing because each walk is independent. DeepWalk's success with multi-label classification and its efficiency has since inspired many other plain graph representation learning methods.

Inspired by DeepWalk, LINE [14] adopts a breadth-first-search strategy to generate node sequences and proposes a solution for learning node embeddings that preserve

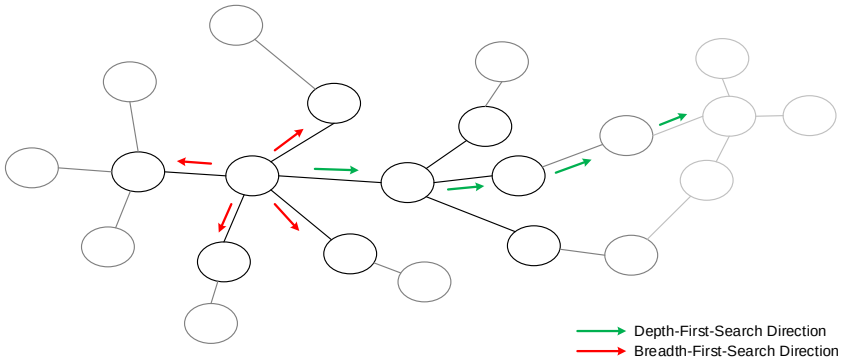


FIGURE 2.2: Different random walk directions.

both local and global structure information with large-scale networks. Another technique inspired by DeepWalk is Node2vec [13], which generates node sequences and learns embeddings by following a more comprehensive process that obeys two principles. First, nodes in the same network community should have similar embeddings. Second, nodes that have similar structural roles should also have similar embeddings. Those are achieved by biased random walks that move in both depth-first-search and breadth-first-search directions. Depth-first-search could reveal the graph's global structure and nodes' roles, while breadth-first-search reveals the local community structure (as shown in Figure 2.2). Each random walk can be treated as a combination of depth-first-search and breadth-first-search on the input graph. Hence, the generated node sequences preserve more structural information than that in DeepWalk and result in more effective node embeddings for graph analytics.

2.3 Attributed Graph Representation Learning

With the emergence of attributed graphs, analysts were given access to information beyond the simple existence of a relationship between two nodes – for example, the year a user joined a social network or the research field of a scientific paper in a citation network. Those attributes provide comprehensive character information about entities, and previous studies on homophily theory [27] suggest that a node's attributes influence

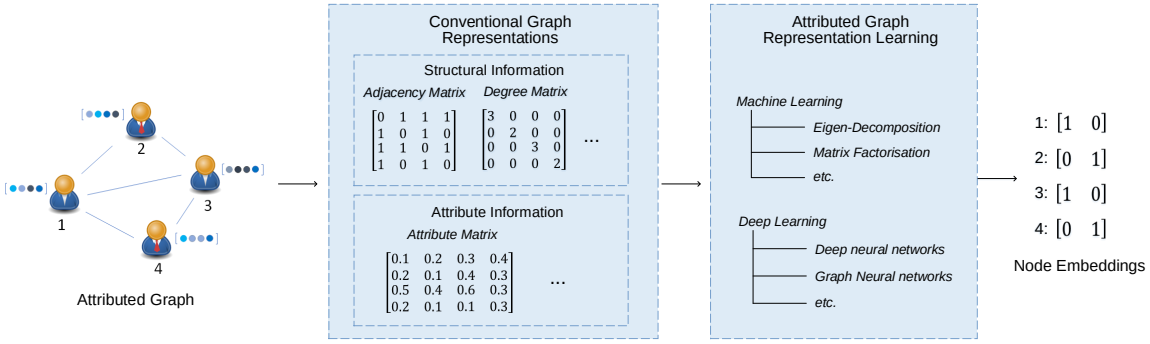


FIGURE 2.3: The process of attributed graph representation learning.

which relationships it will form with other nodes. As a result, significant efforts have been put into utilizing both the graph structure and nodes' attributes to generate more effective node embeddings. As shown in Figure 2.3, existing works typically take both the adjacency matrix and attribute matrix of real-world networks as inputs and output node embeddings by adopting machine learning or deep learning techniques. Following, the most representative methods are briefly described.

TADW [15] is one of the earliest attributed graph embedding methods. It proves that DeepWalk's learning process is equivalent to factorising a transition matrix M that reflects the random walk process. And, motivated by this, TADW attempts to incorporate additional text information associated with each node to perform matrix factorisation on the same matrix M by finding a local minimum. Experiments on real-world attributed networks show that TADW achieves higher node classification accuracy than models that only consider structural information and on par with DeepWalk.

Another earlier technique, ASNE [7] is introduced after TADW, which pays great attention to online social networks. It aims to embed each user into a latent space such that not only linked users are close to each other, but also users with similar attributes are located closely. ASNE takes both the structure information of each node, a one-hot vector, and the attribute information, a feature vector that contains all the attributes *e.g.,* gender, location, as inputs to train a deep neural network, and it generates node embeddings that accomplish the goal.

To generate node embeddings for unseen data, GraphSAGE [28] proposes an inductive

framework that is capable of handling this problem. For each node, GraphSAGE samples a set of its local neighbours and then aggregates those neighbours' features to its embeddings. The traditional mean-pooling, max-pooling, and concatenation are potential candidates for the aggregation function. However, different aggregation functions result in different node embeddings that perform distinctively on graph analytics. This framework has shown promising performance on node classification tasks concerning the classification accuracy.

Apart from GraphSAGE, scalable incomplete network embedding (SINE) [29] is designed to handle large-scale networks with missing information. This method is less sensitive to the scale networks and is robust to incomplete and noisy data. With the node's position information (represented as a one-hot vector) as input, SINE learns a predictive model that could predict its neighbours and its own observable attributes. Further experiments on node classification and link prediction tasks in the literature show that SINE achieves decent results while consuming comparable running time to that of DeepWalk and LINE.

Recently, the application of convolutional neural networks to graph representation has been a highly successful strategy [30]. GAE/VGAE [31] are two examples in this vein. Both are built based on the autoencoder framework with graph convolutional neural network layers. The difference between those two models is that VGAE takes a Gaussian distribution prior to node embeddings, while GAE is non-probabilistic. Both models generate node representations using a convolution operation based on graph signal processing [32], and both have shown promising performance on link prediction tasks with citation networks.

There are many techniques which aim to reduce the memory cost of graph data, from which Binarized attributed network embedding [21] is a representative model. It involves learning binary representations of nodes instead of continuous values in Euclidean space. A novel Weisfeiler-Lehman proximity matrix that captures the relationship between edges and node attributes is defined in this work, and by factorising this matrix, a binary matrix is extracted. Each node corresponds to a vector in this matrix, and the vector has only 1 or 0 in each dimension.

3

Preliminaries

This chapter provides preliminaries of the deep autoencoder framework, different kinds of graphs, and defines proximity measurements used in this work. A detailed list of symbols and notations is in the List of Symbols.

3.1 Autoencoder

In real scenarios, deep learning tasks might face training data without any labeled information when acquiring labels cost too much, or labeling the data is non-trivial for the lack of prior knowledge. To tackle this problem, various unsupervised learning techniques have been introduced to digest valuable information from the unlabeled data. Autoencoder, which is developed based on deep neural network, is at the forefront of those unsupervised deep learning methods [33] and has shown advancement in dimensionality reduction

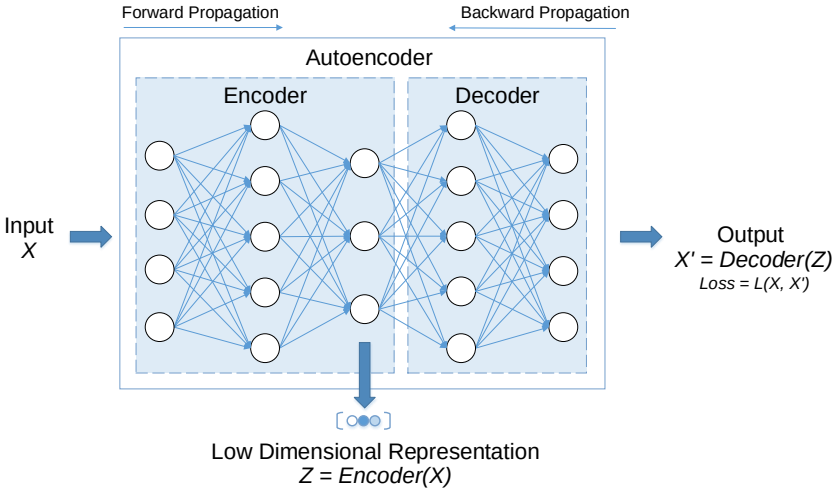


FIGURE 3.1: The framework of Autoencoder.

and feature extraction.

An autoencoder is a deep neural network that is trained to reconstruct its inputs and its general framework is shown in Figure 3.1. As illustrated, an autoencoder contains two main components, namely, the encoder and the decoder. The encoder maps the input data into a lower-dimensional space, while the decoder tends to produce the input data from the latent representations. The training process follows the conventional feed-forward and backward propagation. To be specific, in the forward propagation, the autoencoder first encodes the input X into a lower-dimensional representation Z via encoding neural network layers, then Z is passed through the decoder, and an approximation of X, X', is generated. In the backward propagation, each trainable variable is optimized accordingly to minimize a well-defined, task-oriented loss function $L(\cdot)$. The autoencoder framework is simple, and it shows no advancement to other deep neural network models. However, instead of focusing on the decoder's output, people typically expect the encoder's output, Z, to preserve valuable properties to support downstream applications.

3.2 Plain Graph

Definition 1 (Plain Graph). A plain graph $G = \{V, E\}$ consists of a node set V and an edge set E. The graph structure of a plain graph is formed by nodes $V = \{v_i\}_1^n$ and edges

$E = \{e_{i,j}\}$ where n denotes the number of nodes, and $e_{i,j} = (v_i, v_j)$ represents an edge between the nodes v_i and v_j. An adjacency matrix $A = [a_{i,j}]_{n\times n}$ stores the graph structure, where $a_{i,j} = 1$ if an edge exists between node v_i and v_j, and 0 otherwise.

3.3 Attributed Graph

Definition 2 (Single-attributed-view Graph/Attributed Graph). A single-attributed-view graph, or attributed graph for short, $G = \{V, E, X\}$ consists of a node set V, an edge set E and an attribute set X. The structure of an attributed graph is formed by the nodes $V = \{v_i\}_1^n$ and the edges $E = \{e_{i,j}\}$, where n denotes the number of nodes, and $e_{i,j} = (v_i, v_j)$ represents an edge between nodes v_i and v_j. An adjacency matrix $A = [a_{i,j}]_{n\times n}$ stores the graph structure, where $a_{i,j} = 1$ if an edge exists between node v_i and v_j, and 0 otherwise. The attribute set $X = \{\mathbf{x}_i\}_1^n$ consists of vectors of each nodes' attributes, where $\mathbf{x}_i$ is the attribute vector associated with node v_i.

Definition 3 (Multi-attributed-view Graph). A multi-attributed-view graph $G_M = \{V, E, \mathscr{X}\}$ consists of a node set V, an edge set E and a multi-attributed-view set $\mathscr{X}$. Given m attributed views in $\mathscr{X} = \{X_1, X_2, \ldots, X_m\}$, each attributed view $X_\xi \in \mathscr{X}$ consists of vectors of each nodes' attributes in the corresponding attributed view, ξ. If $m = 1$, the graph is a traditional attributed graph with a single attributed view.

3.4 Pairwise-node Proximity

In this work, three levels of proximity measures are employed to quantify the similarities between pairwise entities. Specifically, the single-attributed-view proximity, or attribute proximity for short, (*Definition 3*), the multi-attributed-view proximity (*Definition 4*), and embedding proximity (*Definition 5*) are considered for pairwise similarities between nodes from the aspects of a single attributed view, multiple attributed-views, and the learned embeddings respectively.

Definition 4 (Single-attributed-view Proximity/Attribute Proximity). Given a pair of nodes v_i and v_j, and their attributes $\mathbf{x}_i$ and $\mathbf{x}_j$ in the same attributed view, the attribute

proximity is denoted as $s^a_{i,j} \in S_a$. S_a presents the attribute proximity over the attributed graph. In this work, the attribute proximity between each pair of nodes is measured as pairwise attribute proximity at the node level in the same attributed view.

Definition 5 (Multi-attributed-view Proximity). In a multi-attributed-view graph, the attributes of a node v_i are stored in m attributed views. Multi-attributed-view proximity measures the pairwise similarity between nodes over multiple attributed views, denoted as $s^v_{i,j} \in S_v$, where $s^v_{i,j}$ is quantified based on attribute proximity $s^a_{i,j}$.

Definition 6 (Embedding Proximity). Embedding proximity is a measure of the similarity between two nodes according to the output embeddings of a multi-attributed-view graph. With d-dimensional embeddings, the embedding proximity of nodes v_i and v_j is denoted as $s^e_{i,j} \in S_e$, where S_e is the embedding proximity matrix.

4

Multi-attributed-view Graph Convolutional Autoencoder

Inspired by the success of graph convolutional neural network [34–36] and autoencoder, MagCAE aims to learn low-dimensional representations for nodes in multi-attributed-view graphs in an unsupervised manner. Section 4.1 introduces the general framework. Section 4.2 and Section 4.3 respectively cover the two main components of MagCAE, the multi-attributed-view graph convolutional encoder, and the multi-attributed-view graph decoder. Section 4.4 presents three levels of proximity. Section 4.5 and Section 4.6 formulate the loss function and optimising strategy.

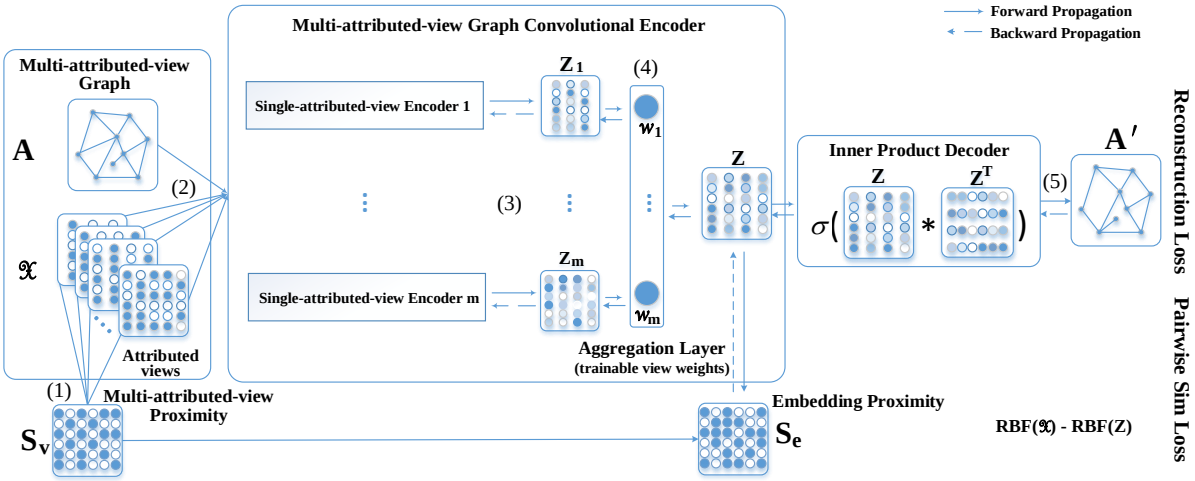


FIGURE 4.1: The framework of MagCAE: a convolutional autoencoder for multi-attributed-view graphs.

4.1 The Overall Framework

The MagCAE model, as shown in Figure 4.1, is built based on the conventional autoencoder framework and generates the low-dimensional representations of nodes that are stored in Z. Similar to other deep learning models, MagCAE is trained through the forward and backward propagation. There are five major steps in the forward propagation. Precisely, in step 1, the multi-attributed-view proximity S_v is measured with regard to nodes' attributes across all attributed views. Then, the multi-attributed-view graph convolutional encoder takes both the adjacency matrix A and the set of multiple attributed views $\mathcal{X} \ni X_\xi$ as inputs and generates m node embeddings, *i.e.,* $Z_1, \ldots, Z_m$, by each single attributed view through step 2 and step 3, respectively. The aggregation layer, which contains trainable view weights, aggregates all those intermediate embeddings and generates the low-dimensional node representations Z consequently in step 4. After that, MagCAE constructs the new adjacency matrix A' by employing an inner product decoder and quantifies the embedding proximity S_e based on Z. The backward propagation, on the other hand, fine-tunes all trainable variables to minimise both the reconstruction loss and the pairwise similarity loss that are formulated in Section 4.5 according to the optimisation strategy in Section 4.6.

4.2 Multi-attributed-view Graph Convolutional Encoder

By taking a multi-attributed-view graph G_M as input, the multi-attributed-view graph convolutional encoder is proposed to generate node embeddings that could preserve both the graph structural information and nodes' attributes. It comprises several independent encoders, each of which learns node embeddings from a single attributed view, and an aggregation layer.

To be specific, if the input multi-attributed-view graph has m views, the multi-attributed-view graph convolutional encoder will contain m independent single-attributed-view encoders. In particular, each single-attributed-view encoder learns node embeddings from a single attributed view as shown in Figure 4.2.

Denoting the ξ-th single-attributed-view encoder, which works on attributed view ξ, as $Encoder_\xi$, its output, Z_ξ, is generated by

$$Z_\xi^{t+1} = \phi_\xi(\tilde{A}Z_\xi^t W_\xi^t), \tag{4.1}$$

where $\phi_\xi(\cdot)$ is the activate function, $\tilde{A} = D^{-1/2}AD^{-1/2}$ is the normalized symmetric adjacency matrix, W_ξ^t contains all the trainable variables in layer t, Z_ξ^t is the output of the t-th layer in $Encoder_\xi$, and $Z_\xi^0 = X_\xi \in \mathbb{R}^{n\times k_\xi}$.

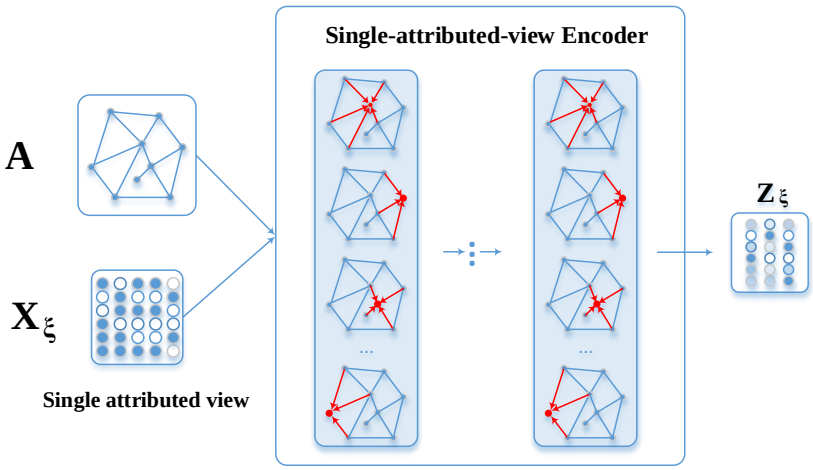


FIGURE 4.2: Single-attributed-view Encoder.

Moreover, an aggregation layer is applied to generate the node embeddings Z and

quantify the weights of different attributed views. This aggregation layer takes the weighted aggregation of all the intermediate node embeddings, $Z_1, \cdots, Z_m$, and generates Z concerning the weights of different attributed views. The process can be represented as

$$Z = \psi(w_1 \cdot Z_1, \cdots, w_m \cdot Z_m), \tag{4.2}$$

where $\psi(\cdot)$ is a weighted aggregation function, and w_ξ is the weight of attributed view ξ. Hence, the learned embeddings Z is a weighted concatenation of all the learned embeddings from the single attributed views.

4.3 Multi-attributed-view Graph Decoder

The traditional inner product decoder [37] is flexible and efficient to use. Therefore, MagCAE takes it to reconstruct the graph structure from the learned embeddings and perform link prediction via

$$A' = \sigma(Z \cdot Z^T), \tag{4.3}$$

where A' denotes the reconstructed adjacency matrix, and $\sigma(\cdot)$ is a Sigmoid activation function that determines the likelihood of two nodes in A' being connected.

4.4 Three Levels of Proximity: Single-view, Multi-view, and Embedding

This work quantifies pairwise-node proximity at three levels. First, the single-attributed-view proximity (attribute proximity for short) is measured between two nodes in a single attributed view, then across multiple views (multi-attributed-view proximity), and finally within the concatenated embedding (embedding proximity) [38].

4.4.1 Single-attributed-view Proximity (Attribute Proximity)

The attribute proximity for two nodes v_i and v_j in attributed view ξ can be denoted as

$$s_{i,j}^{a,\xi} = P(\mathbf{x}_j^{a,\xi} | \mathbf{x}_i^{a,\xi}). \tag{4.4}$$

In this work, the attribute proximity is quantified via $P(\cdot)$ based on two RBF kernels, $\text{RBF}(\mathbf{x}_i^{a,\xi}, \mathbf{x}_j^{a,\xi})$:

$$s_{i,j}^{a,\xi} = \text{RBF}(\mathbf{x}_i^{a,\xi}, \mathbf{x}_j^{a,\xi}) = \exp(-\gamma_\xi ||\mathbf{x}_j^{a,\xi} - \mathbf{x}_i^{a,\xi})||^2), \tag{4.5}$$

where $\gamma_\xi = 1/k_\xi$. Hence, the attribute proximity gets a large value ($s_{i,j}^{a,\xi} \to 1$) when the pairwise nodes share similar attributes; otherwise, $s_{i,j}^{a,\xi} \to 0$.

Other proximity measurements, for instance, the cosine similarity, $L1$ and $L2$ distances, are also available to apply to quantify hierarchical levels of pairwise node proximities. This research uses the RBF kernel for its convenience and follows previous works in this research direction. [38].

4.4.2 Multi-attributed-view Proximity

The multi-attributed-view proximity measures the pairwise node proximity across all attributed views. In this work, the multi-attributed-view proximity is quantified based on the assumption that the data distribution of each attributed view is independent and different because those attributed views are collected from diverse sources. This is calculated as

$$s_{i,j}^{v} = \prod_{\xi}^{m} P(\mathbf{x}_j^{a,\xi} | s_{i,j}^{a,\xi}, \mathbf{x}_i^{a,\xi}). \tag{4.6}$$

A high value in multi-attributed-view proximity indicates the homogeneous attribute distribution in m attributed views and vice versa.

4.4.3 Embedding Proximity

In addition to attribute proximity and multi-attributed-view proximity that are measured to illustrate the similarities between different entities within the input data space, MagCAE also measures their proximity in the embedding space, so-called, embedding proximity, and it is calculated as

$$s_{i,j}^{e} = \exp(-\frac{1}{d}||\mathbf{z}_j - \mathbf{z}_i||^2). \tag{4.7}$$

Similarly, if $s_{i,j}^{e}$ has a high value, then node i and j's learned embeddings z_i, z_j are similar in the embedding space.

The above multi-attributed-view proximity and node embedding proximity capture additional mutual similarity information between nodes to guide the representation learning and are reflected in the novel loss function proposed below.

4.5 The Loss Function

Because MagCAE learns representations that preserve both structural proximity and multi-attributed-view proximity, the framework contains two loss functions. The structural proximities are mapped with the conventional reconstruction loss function (L_{res}) in [33]. Designed for autoencoders, this function measures the likelihood between the input multi-attributed-view graph A and the reconstructed graph A'. The multi-attributed-view proximities are mapped with a novel function designed by us called *pairwise node similarity loss* (L_{sim}). The function preserves multi-attributed-view proximity information in the embeddings by maximising the likelihood between the multi-attributed-view proximity and the embedding proximity over pairwise nodes.

During the training process, L_{res} and L_{sim} are jointly measured in the following overall loss function L:

$$L = L_{res} + \lambda L_{sim}, \tag{4.8}$$

where the parameter λ is used to normalize both losses.

The reconstruction loss function L_{res} for the structural proximities exploits Sigmoid cross-entropy (CE) to control the graph convolutions:

$$L_{res} = \frac{1}{n^2} \sum_i \sum_j \mathrm{CE}(a_{i,j}, a'_{i,j}), \tag{4.9}$$

where $a_{i,j} \in A$ and $a'_{i,j} \in A'$.

The pairwise node proximity loss function L_{sim} is measured over the multi-attributed-view proximity S_v and the embedding proximity S_e:

$$L_{sim} = \frac{1}{n^2} \sum_i \sum_j |s^v_{i,j} - s^e_{i,j}|, \tag{4.10}$$

where $s^v_{i,j} \in S^v$ is calculated by Eq. (4.6), and $s^e_{i,j} \in S_e$ is calculated by Eq. (4.7).

4.6 Optimisation

MagCAE's optimisation objective is to minimise the loss function in Eq. (4.8). In this work, we adopt the adaptive moment estimation (Adam) algorithm to fine-tune all the variables W, $W \supset W_\xi$, in GCN and attributed view weights w_ξ over t iterations. Setting m views with m separate graph convolutional encoders, Adam adapts the learning rate for the attributed view weights w_ξ as per Eq. (4.11) and each variable W_ξ as per Eq. (4.12):

$$w_\xi^t = w_\xi^{t-1} - \alpha_v \frac{\partial L^t}{\partial Z^t} \frac{\partial Z^t}{\partial w_\xi^t}, \tag{4.11}$$

$$W_\xi^t = W_\xi^{t-1} - \alpha_\xi w_\xi^t \frac{\partial L^t}{\partial Z_\xi^t} \frac{\partial Z_\xi^t}{\partial W_\xi^t}, \tag{4.12}$$

where α_v denotes the learning rate of attributed view weights, and α_ξ denotes the learning rate of $Encoder_\xi$.

4.7 Algorithm and Complexity Analysis

The core algorithm of the proposed framework is summarised in Algorithm 1. It strictly follows the representation learning process shown in Figure 4.1. In general, this algorithm takes the multi-attributed-view graph G_M and a predefined hyper-parameter value γ as inputs and outputs the latent representation of each node.

Given a multi-attributed-view graph $G_M(V, E, \mathcal{X})$ with the graph structure represented by the adjacency matrix A and node attribute information restored in $\mathcal{X}$, the initialization of this algorithm contains two main parts. First, it randomly initializes all trainable variables, including variables in the multi-attributed-view graph convolutional encoder, W_ξ^0, and attributed view weights, w_ξ^0, respectively. Second, it calculates the normalized symmetric adjacency matrix $\tilde{A}$ of the input graph G_M, $\tilde{A} = D^{-1/2}AD^{-1/2}$, and measures the multi-attributed-view proximity S_v following Eq. (4.5) and Eq. (4.6).

After initialization, MagCAE is trained consequently to minimize the loss function described in prior sections. In each training epoch, all trainable variables are fine-tuned with regard to the loss caused by the new generated latent representation of each node

Algorithm 1: MagCAE

Input : Multi-attributed-view graph $G_M(V, E, \mathcal{X})$,
Parameter of γ in loss function

Output : Node embeddings Z

Initialization: $\tilde{A} \leftarrow$ normalized symmetric adjacency matrix;
for m **do**
  $W_\xi^0 \leftarrow$ initialized variables for $Encoder_\xi$;
  $Z_\xi^0 = X_\xi \leftarrow$ initialized embeddings;
  $S_\xi^a \leftarrow$ apply Eq. (4.5) for attributed proximity;
end
$w_\xi^0 \leftarrow$ initialized attributed view weight for each attributed view;
$S_v \leftarrow$ apply Eq. (4.6) for multi-attributed-view proximity;
while *Autoencoder* **do**
  for t **do**
    for m **do**
      $Z_\xi^{t+1} \leftarrow$ update embeddings by Eq. (4.1);
    end
  end
  $Z \leftarrow$ aggregate embeddings by Eq. (4.2);
  $A' \leftarrow$ reconstructed adjacency matrix via Eq. (4.3);
  $S_e \leftarrow$ embedding proximity applying Eq. (4.6);
  $L \leftarrow$ update loss function with L_{res} and L_{sim}, applying Eqs. (4.8)-(4.10);
  $W \leftarrow$ update graph convolutional variables;
  $w_\xi \leftarrow$ update attributed view weights;
  break on minimized L
end

and they are updated to minimize the loss function following the strategies denoted by Eq. (4.11) and Eq. (4.12). Specifically, the new node embeddings, Z, is generated by aggregating node embeddings learned from each single attributed view by Eq. (4.2). The training process terminates until a local minimum is reached and the node embeddings for the input graph G can be generated accordingly.

As a result, the proposed MagCAE could generate node embeddings that preserve both the graph structural information and pairwise node attribute proximity information across all attributed views. Given two nodes v_i and v_j, their multi-attributed-view proximity is maintained after multi-attributed-view graph convolutional encoding that is represented by features in Z. The high proximity indicates the similarity between nodes v_i and v_j. In other words, the embedding proximity should be close to the multi-attributed-view proximity. Otherwise, a lower embedding proximity value leads to a larger varies between node-pairwise embedded features.

The computational cost and complexity of graph embedding methods are also of great concern to their downstream applications. With regard to the input data and the model's training procedure, the computational cost of the proposed MagCAE can be divided into the cost of calculating hierarchical levels of the pairwise node similarities and the cost of training. Given a multi-attributed-view graph G with n nodes and m attributed views, the cost of calculating the pairwise node proximity from a single attributed view would be $\mathcal{O}(n^2)$, and the cost will be $\mathcal{O}(m * n^2)$ for measuring the multi-attributed-view proximity. The training cost of the MagCAE is determined by calculations related to the generation of node embeddings (line 12 in Algorithm 1) and measuring the embedding proximity (line 17 in Algorithm 1). In each training iteration, the cost of graph convolution in line 12 is $\mathcal{O}(n * d)$, where d is the dimension of node embeddings, and calculating the embedding proximity costs $\mathcal{O}(n^2)$. In real-world scenarios, the embedding dimension d and the number of attributed views m are far smaller than the number of nodes n. Hence, the overall computation complexity of this model is approximately $\mathcal{O}(n^2)$.

5 Experiments

To evaluate MagCAE, we have conducted an extensive series of comparative experiments with representative alternative methods on two common downstream analysis tasks: link prediction and node classification. The experiments are reproducible, and the model is open-sourced and available to download at https://github.com/MagCAE/magcae.

5.1 Datasets

We select five publicly-available real-world datasets, which have frequently been used for similar evaluations. Three, Cora, Citeseer, and ACM, are citation networks, and two are from well-known customer review sites. Further details follow.

- **Cora**[1] is a citation network of scientific publications. Each node represents a paper,

and each edge represents a citation. The node attributes are encoded as one-hot vectors – one per node – where 0/1 indicates the absence/presence of the corresponding word in the corpus. For our experiments, we constructed a multi-attributed-view graph with three views, based on inherent distinctions between all the attributes. The node label indicates the research topic of the paper.

- **Citeseer**[1] is another widely-used scientific citation network. We pre-processed the raw data to remove nodes with missing attributes and constructed a multi-attributed-view graph with three views based on three given research areas given as one-hot node labels.

- **The ACM**[2] citation network is commonly used to test network embedding performance. We extracted three attributed views from the raw dataset using Doc2Vec [39] – paper title, abstract, and reference titles – and assigned labels to the papers based on the venue they were published in.

- **Epinions**[3] is a consumer review site where users submit reviews of their own and rate the reviews of others. We processed the raw data[4] using Doc2vec [39] and and constructed a multi-attributed-view graph with users as nodes, trust relationships as edges, and three different categories of information as attributed views: user reviews, review ratings, and registration information. The node labels indicate the product category the review is about [40].

- **Ciao**[4] is another consumer review site. We followed the same techniques to extract node features, graph structures, and node labels to result in the same configurations as the Epinions dataset.

Statistics of the five datasets are provided in Table 5.1, including the number of nodes, edges, and attributed views.

[1] https://linqs.soe.ucsc.edu/data
[2] https://www.aminer.org/citation
[3] http://www.epinions.com/
[4] https://www.cse.msu.edu/ tangjili/trust.html

TABLE 5.1: Dataset statistics

	# Nodes	*# Edges*	*# Views*	*# Label Classes*
Cora	2607	5429	3	7
Citeseer	3312	4660	3	6
ACM	7152	12487	3	7
Epinions	36497	215043	3	28
Ciao	10948	99038	3	67

5.2 Baselines

The experiment evaluates the performance of MagCAE in two-stage: the loss function with only the reconstruction loss (MagCAE-res) and the loss function with the reconstruction loss and the pairwise similarity loss (MagCAE). They are further compared with the following baselines.

- **Node2Vec** [13] is a graph embedding model based on a biased random walk guided by two hyperparameters p and q. This method only embeds structural features.

- **LINE** [14] is a graph embedding model for large-scale networks that encodes first- and second-order structural proximities into the representations.

- **TADW** [15] is a DeepWalk-style model that learns graph representations by factorising text features from a matrix and combines the result with random walk-based graph structures.

- **ASNE** [7] is a single-view method that learns node embeddings based on both structural and attributed proximity.

- **MUSAE** [20] captures a node's information from the attributes of its neighbours and encodes the embeddings with a multi-scale approach.

- **SINE** [29] is another attributed graph embedding model that learns node representations from incomplete graph structures based on similarity evaluations.

- **BANE**[5] [21] learns binary node representations by capturing data dependence between edges and attributes and aggregating the information from neighbor nodes.
- **ANRL**[6] [22] represents nodes via an autoencoder neural network equipped with attribute-aware Skip-gram, and formulates the correlations between nodes and (in)direct neighbours through an attribute encoder.
- **GAE** [31] is a single-view unsupervised graph convolutional autoencoder that creates embeddings with both structure and attributes. It is the superior choice for link prediction tasks.
- **VGAE** [31] is a probabilistic variant of GAE that employs Gaussian prior for variational learning.

5.3 Experiment and Parameter Settings

5.3.1 Training, Validation, and Test set

Our treatment of the data is different for the two different tasks. For the link prediction experiments, each dataset is split into a training, validation, and test set. The training set contains all node features and 85% of the existing edges. The validation set, which is used for hyper-parameter optimisation, contains 5% of the existing edges plus the same number of new edges added between randomly selected nodes. The test set contains the remaining 10% of the existing edges and a further equivalent number of additional random edges. For the node classification trials, we randomly sample 80% of the labelled nodes for the training set, and the rest of the nodes are used for the testing set.

5.3.2 Experiment Settings

For a fair comparison, we preprocess all five multi-view datasets based on two different testing scenarios.

[5] https://github.com/benedekrozemberczki/BANE
[6] https://github.com/cszhangzhen/ANRL

- Scenario A (a single concatenated view) concatenates all node attributes from multiple attribute views into a single-attributed view. For plain graph embedding methods, Node2Vec, and LINE, we input the graph structure and ignore node attributes for testing. For attributed graph embedding methods, TADW, ASNE, SINE, MUSAE, BANE, ANRL, GAE, and VGAE, they are trained with all node attributes.

- Scenario B (best of multiple views) trains all attributed graph embedding baselines with node attributes in each attributed view separately and the best result is reported.

Because plain graph embedding methods do not utilize nodes' attributes in the learning process and for simplicity, their test results are reported in Scenario A only. The parameters for all methods are set following the recommendations in the original papers or published implementation.

5.3.3 Evaluation Metrics

The link prediction results are measured in terms of average precision (AP) and area under the ROC curve (AUC) [30]. The node classification task [12] involved applying linear SVM[7] [41] to the learned node embeddings. The results are evaluated in terms of both Micro and Macro F1-scores following 10-fold cross-validation.

5.4 Link Prediction Results

The results of the link prediction tasks appear in Tables 5.2 – 5.5. Across the two scenarios, we make the following observations.

- In general, the proposed MagCAE model outperforms other baselines with respect to both evaluation matrices. The results illustrate that embedding from multi-attributed views can improve the link prediction performance of learned representations.

- The models that use GCN layers are generally more competitive, significantly outperforming the methods that do not in most cases. This result is credited to the graph

[7] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

convolutional algorithm [19, 30] and its ability to encode neighbors' information into the learned embedding as the training progresses.

- MagCAE shows superiority over MagCAE-res on all test datasets, demonstrating the effectiveness of our proposed pairwise similarity loss on improving the embedding performance.

- In comparing the results for Scenarios A and B, we notice that the link prediction performance is not proportional to the number of input node features. Although a concatenation of all node features from different attributed views to a single attributed view extends each node's feature vector, it does not improve the embedding performance. This is a clear demonstration of the motivation behind our work, i.e., that not all attributed views are equal; producing high-quality embeddings requires each to be weighted individually.

TABLE 5.2: Link prediction results on citation networks in scenario A

Method	Input	Cora		Citeseer		ACM	
		AP	AUC	AP	AUC	AP	AUC
Node2Vec	A	80.93±1.4	75.00±0.2	73.41±0.4	61.72±0.3	82.16±0.2	73.04±0.2
LINE	A	74.15±0.7	69.11±0.6	76.44±0.6	65.26±1.1	77.35±0.2	68.17±0.1
TADW	A & X_{con}	76.92±0.4	73.04±0.5	72.18±0.2	65.77±0.5	69.74±0.2	64.46±0.3
ASNE	A & X_{con}	52.22±0.2	52.82±0.1	49.68±0.1	50.37±0.1	50.52±0.1	50.62±0.1
SINE	A & X_{con}	75.20±0.4	70.76±0.3	70.30±0.4	62.34±0.3	79.29±0.3	69.86±0.2
BANE	A & X_{con}	50.62±1.8	51.89±1.7	59.78±1.4	55.42±1.4	48.70±1.6	47.29±2.6
MUSAE	A & X_{con}	74.08±0.3	67.06±0.4	60.50±0.5	52.52±0.3	**83.55±0.2**	**73.80±0.3**
ANRL	A & X_{con}	51.27±0.3	51.20±0.2	59.58±0.7	59.28±1.6	51.70±0.1	51.08±0.3
GAE	A & X_{con}	92.17±0.8	91.10±0.6	91.54±1.0	90.80±0.8	67.00±1.9	65.72±1.1
VGAE	A & X_{con}	92.23±0.6	91.46±0.8	92.50±0.9	91.28±1.0	69.60±0.5	66.89±0.7
MagCAE-res	A & $\mathcal{X}$	92.21±0.8	92.56±0.4	92.41±0.6	91.86±0.7	63.70±0.6	63.72±1.1
MagCAE	A & $\mathcal{X}$	**92.83±0.7**	**93.02±0.6**	**94.78±0.5**	**94.58±0.6**	73.10±0.7	69.78±0.5

1 A = Adjacency matrix. X_{con} = Concatenated single attributed view. $\mathcal{X}$ = Multiple attributed views.

2 *(left)* Mean score. *(right)* Standard deviations. The best result appears in bold.

TABLE 5.3: Link prediction results on citation networks in scenario B

Method	Input	Cora		Citeseer		ACM	
		AP	AUC	AP	AUC	AP	AUC
TADW	A & X_{best}	75.81±0.2	71.19±0.2	73.04±0.4	68.19±0.7	69.22±0.2	64.76±0.2
ASNE	A & X_{best}	52.16±0.1	52.80±0.1	49.67±0.1	50.37±0.1	51.55±0.1	51.34±0.2
SINE	A & X_{best}	80.07±0.4	75.14±0.4	74.62±0.4	65.29±0.3	74.93±0.4	66.27±0.3
BANE	A & X_{best}	58.94±1.6	57.48±1.3	62.15±1.5	57.80±1.4	54.78±2.2	53.23±1.8
MUSAE	A & X_{best}	74.50±0.3	68.20±0.3	58.38±0.5	50.44±0.4	**83.90±0.1**	**74.14±0.2**
ANRL	A & X_{best}	49.24±0.3	51.75±0.2	47.67±0.2	52.92±0.3	60.65±0.1	60.69±0.2
GAE	A & X_{best}	91.65±0.5	89.54±0.7	92.98±0.5	91.19±0.9	68.63±0.2	68.33±0.2
VGAE	A & X_{best}	92.09±0.3	90.35±0.5	92.52±0.7	90.77±0.8	66.68±0.2	65.38±0.3
MagCAE-res	A & $\mathscr{X}$	92.21±0.8	92.56±0.4	92.41±0.6	91.86±0.7	63.70±0.6	63.72±1.1
MagCAE	A & $\mathscr{X}$	**92.83±0.7**	**93.02±0.6**	**94.78±0.5**	**94.58±0.6**	73.10±0.7	69.78±0.5

1 A = Adjacency matrix. X_{best} = Single attributed view with best performance. $\mathscr{X}$ = Multiple attributed views.

2 *(left)* Mean score. *(right)* Standard deviations. The best result appears in bold.

TABLE 5.4: Link prediction results on customer review networks in scenario A

Method	Input	Epinions		Ciao	
		AP	AUC	AP	AUC
Node2Vec	A	45.84±0.1	50.65±0.1	63.32±0.1	66.73±0.1
LINE	A	85.54±0.2	81.40±0.2	90.96±0.1	89.49±0.1
TADW	A & X_{con}	79.17±0.1	78.08±0.1	80.06±0.1	78.74±0.1
ASNE	A & X_{con}	55.77±0.2	56.90±0.3	51.84±0.2	52.87±0.2
SINE	A & X_{con}	39.34±0.1	35.42±0.1	37.05±0.2	26.81±0.5
BANE	A & X_{con}	55.51±5.4	58.73±7.3	51.31±3.1	52.45±5.4
MUSAE	A & X_{con}	OOM	OOM	80.97±0.3	81.98±0.3
ANRL	A & X_{con}	49.85±0.1	50.22±0.1	64.04±0.1	65.73±0.2
GAE	A & X_{con}	92.97±0.3	89.54±0.4	89.41±0.2	87.48±0.5
VGAE	A & X_{con}	91.95±1.1	89.26±0.3	83.33±0.3	85.17±0.2
MagCAE-res	A & $\mathscr{X}$	93.02±0.8	90.45±1.7	90.94±0.5	89.37±0.3
MagCAE	A & $\mathscr{X}$	**93.66±0.9**	**91.28±1.8**	**91.21±0.1**	**89.64±0.2**

1 A = Adjacency matrix. X_{con} = Concatenated single attributed view. $\mathscr{X}$ = Multiple attributed views.

2 *(left)* Mean score. *(right)* Standard deviations. The best result appears in bold. OOM = Out of memory.

TABLE 5.5: Link prediction results on customer review networks in scenario B

Method	Input	Epinions		Ciao	
		AP	AUC	AP	AUC
TADW	A & X_{best}	77.08±0.1	79.89±0.1	77.50±0.1	80.02±0.1
ASNE	A & X_{best}	80.64±0.2	79.99±0.2	69.31±0.2	70.81±0.2
SINE	A & X_{best}	43.15±0.2	44.71±0.4	66.19±0.2	68.17±0.2
BANE	A & X_{best}	73.99±3.9	74.98±2.4	70.34±6.1	72.18±5.7
MUSAE	A & X_{best}	76.02±0.1	76.31±0.1	81.54±0.1	81.54±0.1
ANRL	A & X_{best}	49.55±0.4	50.86±0.5	50.21±0.2	50.33±0.2
GAE	A & X_{best}	92.95±0.4	89.68±0.6	90.29±0.3	88.39±0.5
VGAE	A & X_{best}	92.82±0.6	88.66±0.8	90.11±0.3	88.19±0.4
MagCAE-res	A & $\mathscr{X}$	93.02±0.8	90.45±1.7	90.94±0.5	89.37±0.3
MagCAE	A & $\mathscr{X}$	**93.66±0.9**	**91.28±1.8**	**91.21±0.1**	**89.64±0.2**

1 A = Adjacency matrix. X_{best} = Single attributed view with best performance. $\mathscr{X}$ = Multiple attributed views.

2 *(left)* Mean score. *(right)* Standard deviations. The best result appears in bold.

5.5 Node Classification Results

Tables 5.6 - 5.9 show the Micro and Macro F1-scores for the node classification task. Similar to link prediction, the result analysis is as follows.

- MagCAE and MagCAE-res significantly outperform other baseline models on the Cora and Citeseer datasets with scores around 1.2% higher and generally above-average scores with the remaining datasets.
- Most of the attributed graph embedding methods outperform Node2Vec and LINE, indicating that using both graph structural information and node attributes improves the classification accuracy.
- Comparing Scenarios A and B, we again see little difference between embedding the individual views and concatenating all the views into one embedding.

TABLE 5.6: Node classification results on citation networks in scenario A

Method	Input	Cora		Citeseer		ACM	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
Node2Vec	A	27.56±3.1	11.17±2.3	19.82±2.8	16.34±2.0	19.82±1.6	10.66±0.9
LINE	A	29.74±2.1	6.55±0.4	19.91±2.5	13.75±1.8	20.71±1.7	10.97±1.0
TADW	A & X_{con}	46.46±3.4	40.47±4.6	39.97±5.1	36.81±5.4	39.50±2.3	35.36±2.1
ASNE	A & X_{con}	29.74±2.1	6.55±0.4	30.53±3.7	25.81±3.2	34.65±2.6	28.07±2.3
SINE	A & X_{con}	25.44±2.6	11.39±2.1	60.66±2.2	56.52±2.1	20.06±1.3	10.77±0.9
BANE	A & X_{con}	24.70±3.8	11.58±2.9	18.67±2.4	13.19±1.8	21.32±1.5	10.88±1.5
MUSAE	A & X_{con}	20.28±2.5	14.61±2.8	28.05±2.0	26.07±1.9	18.10±2.2	12.50±1.4
ANRL	A & X_{con}	47.21±2.7	36.03±2.8	61.81±3.5	54.80±3.9	**43.29±3.2**	**41.61±3.1**
GAE	A & X_{con}	78.25±2.9	76.68±3.1	60.27±5.0	52.62±4.0	25.64±2.3	13.39±1.0
VGAE	A & X_{con}	79.17±2.7	77.32±3.2	60.71±3.7	53.37±2.6	27.94±2.6	14.49±1.1
MagCAE-res	A & $\mathscr{X}$	**88.19±2.4**	**86.97±3.1**	72.29±3.8	69.85±3.8	27.73±2.1	19.97±1.8
MagCAE	A & $\mathscr{X}$	85.98±2.4	84.41±2.8	**73.66±4.0**	**70.16±3.8**	34.30±2.4	29.48±2.4

A = Adjacency matrix. X_{con} = Concatenated single attributed view. $\mathscr{X}$ = Multiple attributed views.
(left) Mean score. *(right)* Standard deviations. The best result appears in bold.

TABLE 5.7: Node classification results on citation networks in scenario B

Method	Input	Cora		Citeseer		ACM	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1	Mic-F1	Mac-F1
TADW	A & X_{best}	40.20±3.1	34.77±4.3	35.61±5.0	31.51±4.9	45.20±1.9	41.57±2.4
ASNE	A & X_{best}	29.74±2.1	6.55±0.4	30.56±3.4	25.86±3.0	43.14±1.8	37.23±1.9
SINE	A & X_{best}	53.25±2.6	49.66±4.2	53.50±3.0	49.69±2.6	21.66±1.6	11.65±0.8
BANE	A & X_{best}	29.74±2.1	6.55±0.4	20.27±2.2	15.19±2.3	21.20±2.0	9.15±1.5
MUSAE	A & X_{best}	24.69±4.7	19.38±4.2	25.20±3.2	23.71±3.3	18.87±1.4	12.75±1.1
ANRL	A & X_{best}	29.74±2.1	6.55±0.4	43.12±3.6	33.11±2.3	**47.34±3.5**	**46.72±3.2**
GAE	A & X_{best}	76.85±2.3	75.02±3.3	62.17±4.5	54.39±4.1	32.29±1.8	20.65±2.1
VGAE	A & X_{best}	75.81±2.9	74.61±4.7	62.34±4.1	54.70±3.6	32.52±1.9	21.16±1.7
MagCAE-res	A & $\mathscr{X}$	**88.19±2.4**	**86.97±3.1**	72.29±3.8	69.85±3.8	27.73±2.1	19.97±1.8
MagCAE	A & $\mathscr{X}$	85.98±2.4	84.41±2.8	**73.66±4.0**	**70.16±3.8**	34.30±2.4	29.48±2.4

A = Adjacency matrix. X_{best} = Single attributed view with best performance. $\mathscr{X}$ = Multiple attributed views.
(left) Mean score. *(right)* Standard deviations. The best result appears in bold.

TABLE 5.8: Node classification results on customer review networks in scenario A

Method	Input	Epinions		Ciao	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1
Node2Vec	A & X_{con}	11.54±0.3	1.43±0.1	14.56±1.0	1.76±0.2
LINE	A & X_{con}	11.29±0.6	1.36±0.1	14.71±1.6	1.70±0.2
TADW	A & X_{con}	**16.84±1.0**	**4.24±0.1**	29.14±2.8	13.56±2.9
ASNE	A & X_{con}	12.43±0.6	1.58±0.1	25.08±0.9	8.76±1.3
SINE	A & X_{con}	11.62±0.6	1.44±0.1	14.83±1.2	1.97±0.3
BANE	A & X_{con}	11.56±0.4	1.59±0.1	15.14±1.8	1.56±0.3
MUSAE	A & X_{con}	OOM	OOM	11.39±1.2	3.29±0.5
ANRL	A & X_{con}	13.32±2.7	1.77±2.0	28.08±1.7	13.50±6.2
GAE	A & X_{con}	11.72±0.7	0.94±0.1	15.61±2.1	1.20±0.4
VGAE	A & X_{con}	12.69 ± 1.0	1.89±0.3	19.03±2.1	3.79±0.4
MagCAE-res	A & $\mathcal{X}$	13.86±0.6	2.14±0.1	16.54±2.1	2.53±0.3
MagCAE	A & $\mathcal{X}$	14.66±1.1	3.67±0.4	**29.43±1.8**	**13.84±0.5**

A = Adjacency matrix. X_{con} = Concatenated single attributed view. $\mathcal{X}$ = Multiple attributed views.
(left) Mean score. *(right)* Standard deviations. The best result appears in bold. OOM = Out of memory.

TABLE 5.9: Node classification results on customer review networks in scenario B

Method	Input	Epinions		Ciao	
		Mic-F1	Mac-F1	Mic-F1	Mac-F1
TADW	A & X_{best}	14.60±0.8	3.46±0.3	**41.91±1.8**	**25.87±2.6**
ASNE	A & X_{best}	12.52±0.6	1.59±0.1	25.17±1.4	10.43±1.1
SINE	A & X_{best}	12.90±0.7	2.10±0.5	17.07±1.3	3.90±0.9
BANE	A & X_{best}	11.56±0.7	1.26±0.4	15.13±1.7	1.67±0.4
MUSAE	A & X_{best}	10.61±0.6	2.29±0.3	12.17±1.1	3.05±0.5
ANRL	A & X_{best}	14.32±2.1	2.30±0.3	15.50±2.0	1.04±0.1
GAE	A & X_{best}	12.37±0.9	1.47±0.1	19.69±1.4	4.44±0.3
VGAE	A & X_{best}	12.56±0.8	1.53±0.1	22.17±1.6	5.75±0.6
MagCAE-res	A & $\mathcal{X}$	13.86±0.6	2.14±0.1	16.54±2.1	2.53±0.3
MagCAE	A & $\mathcal{X}$	**14.66±1.1**	**3.67±0.4**	29.43±1.8	13.84±0.5

A = Adjacency matrix. X_{best} = Single attributed view with best performance. $\mathcal{X}$ = Multiple attributed views.
(left) Mean score. *(right)* Standard deviations. The best result appears in bold. OOM = Out of memory.

5.6 Discussion

5.6.1 MagCAE's Effectiveness

Although all attributed graph embedding baseline models utilize node features from all attributed views and achieve decent performance in test Scenario B, they are not capable of digesting the inter-correlations between different attributed views. In comparing the results for Scenarios A and B, it is clear that applying a simple aggregation method like concatenation, which treats each attributed view equally important, generally leads to lesser performance because it ignores the weight of each attributed view. MagCAE shows its superiority by learning the weights of multiple attributed views in an adaptive manner.

5.6.2 Convergence Analysis

We assess MagCAE's convergence by measuring its link prediction result on the validation set. As shown in Figure 5.1, performance improves rapidly over the first 25 training iterations and begins to converge after 50 epochs. The chosen Adam algorithm tunes each trainable variables and manages to find the sub-optimal result in about 65 training loops.

5.6.3 Robustness to Incomplete Graphs

To evaluate the robustness of our proposed model on incomplete graphs, we conduct a series of experiments with edges removed from the training set. More specificly, we set the training edges ratio to $\{0.5, 0.6, 0.7, 0.8, 0.85\}$ and repeat the test ten times for each setting. The average results are reported in Figure 5.2. As can be seen, MagCAE achieves respectable link prediction performance with only 50% training edges. The average precision score and area under curve score are above 90% on both the Cora and Citeseer datasets. This outstanding performance may be because the correlations between multiple attributed views contribute to the formation of edges between node pairs and MagCAE is capable of leveraging such information for the embedding. Thus, our model shows robustness to incomplete multi-attributed-view graphs.

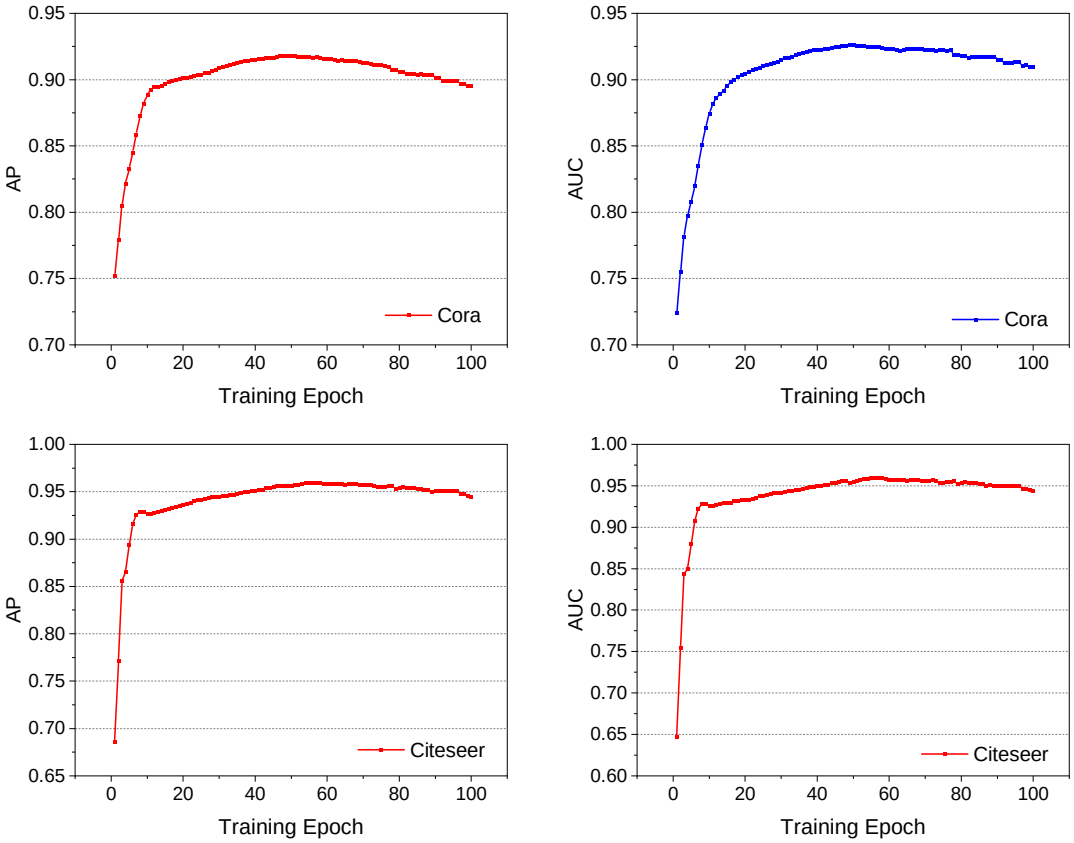


FIGURE 5.1: The training curve of MagCAE.

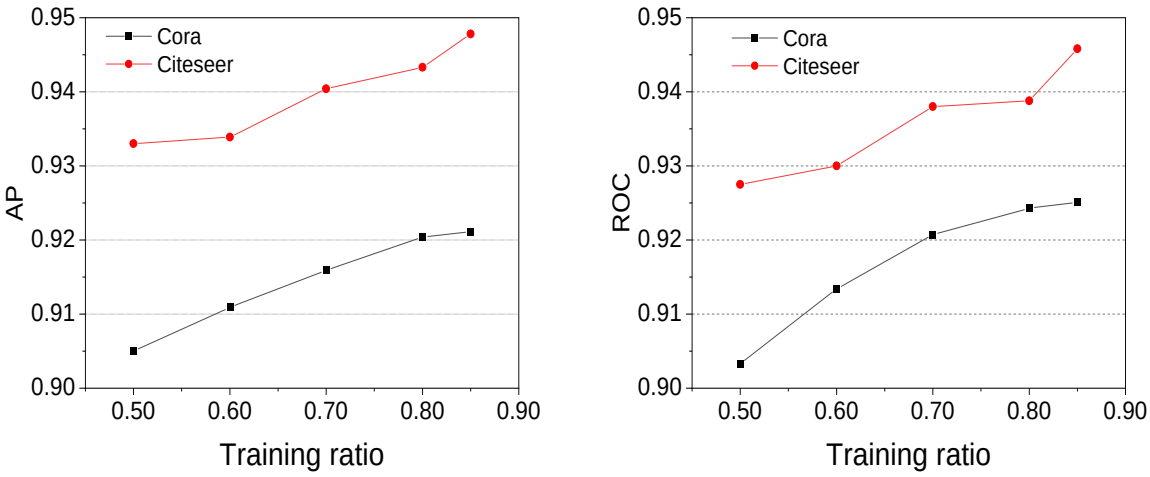


FIGURE 5.2: Training ratios v.s. link prediction performance.

5.7 Parameter Analysis

5.7.1 Hidden Layer Dimensionality

Instead of fixing the dimensions of hidden neural network layers, in MagCAE, we set the number of neurons in each hidden layer to different proportions of the input attribute dimension. Denote the portion as p, then the size of the embedding layer and the first hidden layer is set to $p * m$, and $2 * p * m$, where m is the input attributed view dimension, respectively. Figure 5.3 shows the effect of different values of p. On the whole, performance remains stready, peaking at $p = 0.3$.

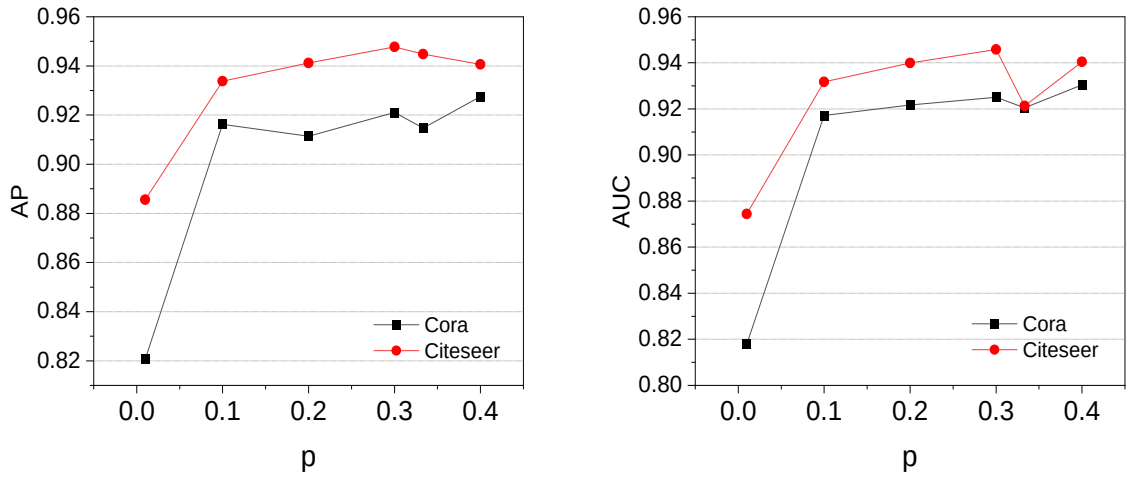


FIGURE 5.3: p Embedding dimensions v.s. link prediction performance.

5.7.2 λ Coefficient Sensitivity

The coefficient λ determines the weight of pairwise similarity loss versus the total training error. We set λ to 0.1, 0.5, 1, 1.5, and 2 and investigate its impact on our model, noting that at λ is 0, MagCAE becomes MagCAE-res. Figure 5.4 shows the results. Here, we observe that a heavier weight of pairwise similarity loss function leads to a better performance up to a tipping point. For the Cora dataset, that point is $\lambda = 1$, and 1.5 for the Citeseer dataset.

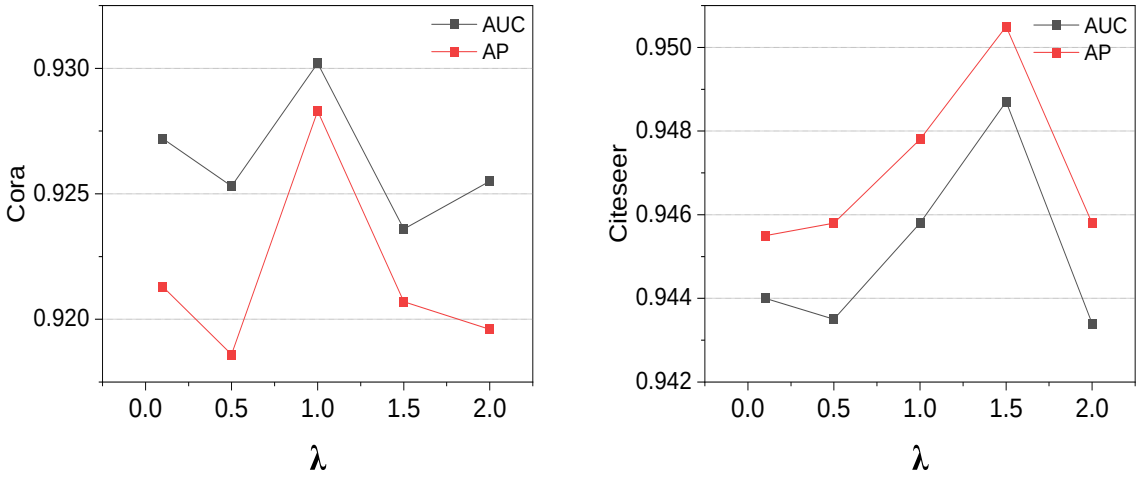
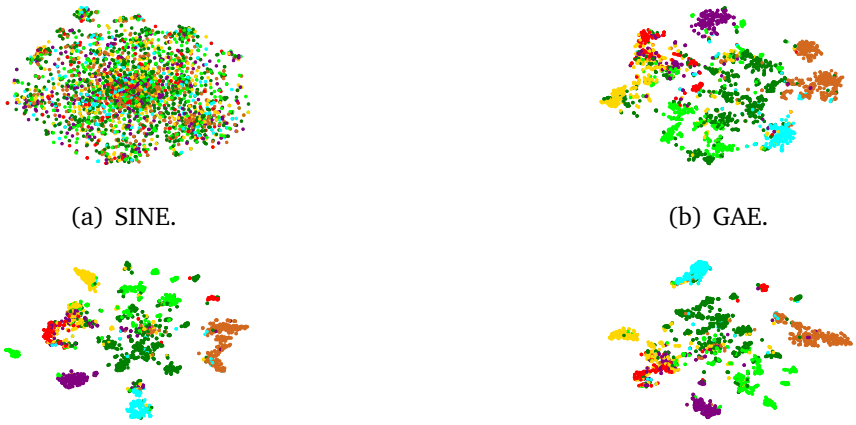


FIGURE 5.4: λ Coefficient v.s. link prediction performance.



(a) SINE. (b) GAE.

(c) MagCAE-res. (d) MagCAE.

FIGURE 5.5: Node visualization results of different graph embedding models on Cora.

5.8 Visualization of the Node Embeddings

We visualize the embedding results of our approach in a two-dimensional space by applying t-SNE [42], as shown in Figure 5.5. Due to space constraints, we have only included visualizations for two of the representative baselines with the Cora dataset. From these, we can see that the graph convolutional autoencoder is generally able to generate distinguishable node embeddings. However, MagCAE's embeddings are better separated - nodes are more evident with less overlap and the boundaries between different groups are more explicit.

6

Conclusion and Future Works

Learning low dimensional representations of graphs can bring in benefits for downstream applications and graph analysis tasks. However, existing works have not paid enough attention to the impact of multiple attributed views on the learned embeddings. In this work, we studied the multi-attributed-view graph representation learning problem and proposed a novel model MagCAE to learn node embeddings from multiple attributed views. We adopted graph convolutional neural network to form a multi-attributed-view graph convolutional autoencoder and proposed a novel loss function to preserve multi-attributed-view proximity in graph embeddings. Extensive experiments on five real-world datasets validated the effectiveness of MagCAE by comparisons with ten state-of-the-art models.

While this work has shown promising performance beyond existing works, graph

representation learning still faces great challenges due to real-world networks' complexity. Consequently, one of our future works will be focusing on developing more effective deep graph learning models for dynamic [43], large-scale [44], and heterogeneous graphs [45, 46]. Moreover, we tend to explore potential downstream applications of this research, for instance, utilizing multi-attributed-view graph representation learning for social recommendation, anomaly detection, cybersecurity, *etc*.

References

[1] P. Cui, X. Wang, J. Pei, and W. Zhu. *A survey on network embedding*. IEEE Transactions on Knowledge and Data Engineering **31**(5), 833 (2018). 1, 2, 3, 8

[2] S. A. Myers, A. Sharma, P. Gupta, and J. Lin. *Information network or social network? the structure of the twitter follow graph*. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, p. 493–498 (Association for Computing Machinery, New York, NY, USA, 2014). URL https://doi.org/10.1145/2567948.2576939.

[3] T. Ji, Z. Chen, N. Self, K. Fu, C. Lu, and N. Ramakrishnan. *Patent citation dynamics modeling via multi-attention recurrent networks*. In S. Kraus, ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 2621–2627 (ijcai.org, 2019). URL https://doi.org/10.24963/ijcai.2019/364. 1

[4] S. Gao, L. Denoyer, and P. Gallinari. *Temporal link prediction by integrating content and structure information*. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 1169–1174 (2011). 2

[5] J. Tang, C. Aggarwal, and H. Liu. *Node classification in signed social networks*. In *Proceedings of the 2016 SIAM international conference on data mining*, pp. 54–62 (SIAM, 2016). 2

[6] Q. Zhang, L. T. Yang, Z. Chen, and P. Li. *A survey on deep learning for big data*. Information Fusion **42**, 146 (2018). 2

[7] L. Liao, X. He, H. Zhang, and T.-S. Chua. *Attributed social network embedding*. IEEE Transactions on Knowledge and Data Engineering **30**(12), 2257 (2018). 2, 3, 11, 29

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. *A comprehensive survey on graph neural networks*. IEEE Transactions on Neural Networks and Learning Systems (2020). 2

[9] W. L. Hamilton, R. Ying, and J. Leskovec. *Representation learning on graphs: Methods and applications*. arXiv preprint arXiv:1709.05584 (2017). 2

[10] M. Belkin and P. Niyogi. *Laplacian eigenmaps and spectral techniques for embedding and clustering*. In *Advances in neural information processing systems*, pp. 585–591 (2002). 2, 8

[11] X. He and P. Niyogi. *Locality preserving projections*. In *Advances in neural information processing systems*, pp. 153–160 (2004). 2, 8

[12] B. Perozzi, R. Al-Rfou, and S. Skiena. *Deepwalk: Online learning of social representations*. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710 (2014). 2, 9, 31

[13] A. Grover and J. Leskovec. *node2vec: Scalable feature learning for networks*. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864 (2016). 2, 10, 29

[14] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. *Line: Large-scale information network embedding*. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077 (2015). 2, 9, 29

[15] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang. *Network representation learning with rich text information*. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015). 2, 11, 29

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. *Graph attention networks*. arXiv preprint arXiv:1710.10903 (2017).

[17] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo. *Graphgan: Graph representation learning with generative adversarial nets*. arXiv preprint arXiv:1711.08267 (2017).

[18] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. *Graph neural networks: A review of methods and applications*. CoRR **abs/1812.08434** (2018). 1812.08434, URL http://arxiv.org/abs/1812.08434. 2

[19] M. Henaff, J. Bruna, and Y. LeCun. *Deep convolutional networks on graph-structured data*. arXiv preprint arXiv:1506.05163 (2015). 3, 32

[20] B. Rozemberczki, C. Allen, and R. Sarkar. *Multi-scale attributed node embedding*. arXiv preprint arXiv:1909.13021 (2019). 3, 29

[21] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang. *Binarized attributed network embedding*. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 1476–1481 (IEEE, 2018). 12, 30

[22] Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang. *Anrl: Attributed network representation learning via deep neural networks*. In *IJCAI*, vol. 18, pp. 3155–3161 (2018). 3, 30

[23] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang. *Tri-party deep network representation*. Network **11**(9), 12 (2016). 3

[24] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. *Heterogeneous graph neural network*. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19, p. 793–803 (Association for Computing Machinery, New York, NY, USA, 2019). URL https://doi.org/10.1145/3292500.3330961.

[25] J. Wu, S. Pan, X. Zhu, Z. Cai, and C. Zhang. *Multi-graph-view learning for complicated object classification*. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[26] J. Wu, S. Pan, X. Zhu, C. Zhang, and S. Y. Philip. *Multiple structure-view learning for graph classification*. IEEE transactions on neural networks and learning systems **29**(7), 3236 (2017). 3

[27] M. McPherson, L. Smith-Lovin, and J. M. Cook. *Birds of a feather: Homophily in social networks*. Annual review of sociology **27**(1), 415 (2001). 10

[28] W. Hamilton, Z. Ying, and J. Leskovec. *Inductive representation learning on large graphs*. In *Advances in neural information processing systems*, pp. 1024–1034 (2017). 11

[29] D. Zhang, J. Yin, X. Zhu, and C. Zhang. *Sine: Scalable incomplete network embedding*. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 737–746 (IEEE, 2018). 12, 29

[30] T. N. Kipf and M. Welling. *Semi-supervised classification with graph convolutional networks*. arXiv preprint arXiv:1609.02907 (2016). 12, 31, 32

[31] T. N. Kipf and M. Welling. *Variational graph auto-encoders*. arXiv preprint arXiv:1611.07308 (2016). 12, 30

[32] B. A. Miller, N. T. Bliss, and P. J. Wolfe. *Toward signal processing theory for graphs and non-euclidean data*. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5414–5417 (IEEE, 2010). 12

[33] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning* (MIT press, 2016). 13, 22

[34] X. Wei, R. Yu, and J. Sun. *View-gcn: View-based graph convolutional network for 3d shape analysis*. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 1847–1856 (IEEE, 2020). URL https://doi.org/10.1109/CVPR42600.2020.00192. 17

[35] L. Wang, Z. You, Y. Li, K. Zheng, and Y. Huang. *GCNCDA: A new method for predicting circrna-disease associations based on graph convolutional network algorithm*. PLoS Computational Biology **16**(5) (2020). URL https://doi.org/10.1371/journal.pcbi.1007568.

[36] J. Zhang, X. Shi, S. Zhao, and I. King. *STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems*. In S. Kraus, ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4264–4270 (ijcai.org, 2019). URL https://doi.org/10.24963/ijcai.2019/592. 17

[37] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang. *Attributed graph clustering: A deep attentional embedding approach*. arXiv preprint arXiv:1906.06532 (2019). 20

[38] J. Zhang, B. Cao, S. Xie, C. Lu, P. S. Yu, and A. B. Ragin. *Identifying connectivity patterns for brain diseases via multi-side-view guided deep architectures*. In S. C. Venkatasubramanian and W. M. Jr., eds., *Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, Florida, USA, May 5-7, 2016*, pp. 36–44 (SIAM, 2016). URL https://doi.org/10.1137/1.9781611974348.5. 20, 21

[39] Q. Le and T. Mikolov. *Distributed representations of sentences and documents*. In *International conference on machine learning*, pp. 1188–1196 (2014). 28

[40] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang. *Multi-dimensional graph convolutional networks*. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 657–665 (SIAM, 2019). 28

[41] C.-C. Chang and C.-J. Lin. *Libsvm: A library for support vector machines*. ACM transactions on intelligent systems and technology (TIST) **2**(3), 1 (2011). 31

[42] L. v. d. Maaten and G. Hinton. *Visualizing data using t-sne*. Journal of machine learning research **9**(Nov), 2579 (2008). 40

[43] F. Béres, D. M. Kelen, R. Pálovics, and A. A. Benczúr. *Node embeddings in dynamic*

graphs. Applied Network Science **4**(1), 64:1 (2019). URL https://doi.org/10.1007/s41109-019-0169-5. 42

[44] B. Hettige, Y. Li, W. Wang, and W. L. Buntine. *Gaussian embedding of large-scale attributed graphs*. CoRR **abs/1912.00536** (2019). 1912.00536, URL http://arxiv.org/abs/1912.00536. 42

[45] C. Zhao, H. Wang, Y. Li, and K. Mu. *Combining meta-graph and attention for recommendation over heterogenous information network*. In *International Conference on Database Systems for Advanced Applications*, pp. 383–397 (Springer, 2019). 42

[46] J. Yin, Y. Guo, and Y. Chen. *Heterogenous information network embedding based cross-domain recommendation system*. In P. Papapetrou, X. Cheng, and Q. He, eds., *2019 International Conference on Data Mining Workshops, ICDM Workshops 2019, Beijing, China, November 8-11, 2019*, pp. 362–369 (IEEE, 2019). URL https://doi.org/10.1109/ICDMW.2019.00060. 42