Received 31 August 2023, accepted 24 September 2023, date of publication 27 September 2023, date of current version 11 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3320059

RESEARCH ARTICLE

Attributed Network Embedding Using an Improved Weisfeiler-Lehman Schema and a Novel Deep Skip-Gram

AMR AL-FURAS[1,2], **MOHAMMED F. ALRAHMAWY**[2], **ABDULAZIZ ALBLWI**[3], **WALEED MOHAMED AL-ADROUSY**[2], **AND SAMIR ELMOUGY**[2]

[1]Computer Science Department, Ibb University, Ibb, Yemen
[2]Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt
[3]Department of Computer Science, Applied College, Taibah University, Medina 41477, Saudi Arabia

Corresponding author: Amr Al-Furas (amroso783@gmail.com)

ABSTRACT Attributed Network Embedding (ANE) and the representation of its nodes in a low-dimensional space is a pivotal step in the analysis of real-world networks. One of the biggest challenges in the embedding process of nodes in complex networks is to capture any dynamic changes in both the node itself and in its adjacent. To address the above challenge, in this paper, we propose a novel ANE model that combines an improved Weisfeiler-Lehman Information Aggregation (WLIA) schema with a novel Deep Skip-Gram (DSG) approach. First, an information aggregation of network data is performed using an improved Weisfeiler-Lehman, which captures each node's attributes and combines them with the attributes of its adjacent nodes in a mathematically proven balanced and fair manner. Next, a novel deep autoencoder model that adopts the Skip-Gram approach to capture the high non-linearity among the nodes and between nodes with their attributes is proposed. In the DSG approach, a deep encoder is paired with a set of deep decoders; the main decoder is for the node itself and the secondary deep decoders act as attention decoders to extract common features from its neighbors. Extensive experimental evaluations have demonstrated that the proposed method is superior in performance compared to recent network embedding models.

INDEX TERMS Attributed network embedding, Weisfeiler-Lehman, skip-gram, random walks, autoencoder.

I. INTRODUCTION

An attributed network describes an extensive set of complex networks in which the links between nodes are important for analysis as node attributes. Networks of protein-protein interactions [1], citation networks [2], [3], and social networks [4], [5] are typical examples of attributed networks. The analysis of attributed networks is highly important because these networks represent an integral part of the real world. For instance, social networks increasingly reflect intricate details of our daily lives, harnessing the wealth of information they provide. Consequently, this research area is highly active, engaging researchers and developers from diverse fields such as security, economics, and humanitarian pursuits [6], [7]. An effective analysis of this type of network depends deeply on the way these networks are represented [8]. Several network-embedding models have been proposed to represent nodes in low-dimensional vectors to facilitate analysis. Network embedding is a mapping function for representing network nodes in a low-dimensional space while preserving the distinctive features of each node, which facilitates the use of machine-learning methods in analyzing network data. Attributed Network Embedding allows us to incorporate rich node information such as node attributes, node features, and node labels into the embedding process. This can help us to better understand the structure and function of the network, as well as to identify important nodes and relationships within the network [8], [9]. Moreover, attributed network embedding can be used for a wide range of applications, such as node classification [10], [11], link prediction [12], [13], and

The associate editor coordinating the review of this manuscript and approving it for publication was Feiqi Deng.

community detection [14], [15], where the learned representations can be used as input features for downstream machine-learning tasks. The design of a mapping function for network embedding involves several important considerations. Efficiency, adaptability, proximity, and latent information are crucial considerations when designing network embedding models for large-scale, dynamic networks [16]. To address computational challenges, the mapping function should be efficient in terms of time and space utilization [17]. Additionally, the mapping function should possess adaptability to accommodate changes in network relationships without necessitating complete retraining [18]. Proximity, or the level of relatedness between nodes, is an essential aspect to consider, as it varies within a network. Capturing these varying levels of proximity accurately is vital for an accurate representation of the network structure [19]. Moreover, real-world networks often contain latent information that is not explicitly present in the network data itself. To provide a more comprehensive representation, the mapping function should be able to capture and incorporate this latent information into the embedding model [20]. By considering these factors, network embedding models can better handle the complexities and nuances of real-world networks.

Owing to these considerations, several embedding methods have been proposed. Some of these embedding methods depend on the degree of proximity between the nodes [2], [21], [22], where each node is placed with its adjacent nodes of the first, second, or higher degrees of proximity in the embedding space. Another group of embedding methods relies on graph kernel techniques, which are embedding network techniques that deal with the network from a micro-level perspective to show the nodes in a comparable manner based on both their structural role in the network and their location, in addition to their features [23], [24]. The Weisfeiler-Lehman Graph kernel [25] is a technique that combines scalability with the ability to consider node attributes.

A new set of methods has been presented in recent research, such as DeepWalk [26], Node2Vec [27], and Walklets [28]. These methods are inspired by word embedding methods that aim to maintain word convergence based on skip-grams [29]. In these methods, a series of random walks are generated from the nodes and their neighbors, where each node represents a word, each path represents a sentence, and the paths collectively represent the entire text. These methods significantly improve the embedding performance. However, these methods only consider the network topology and ignore the node attributes. To address this challenge, several enhanced methods have been proposed as a generalization of these methods, as they consider the network structure along with node attributes [30], [31]. Although these embedding methods have shown promising results, they still have certain drawbacks. This is because the nature of complex networks cannot be compared to the architecture of language texts and words, as words associations within a certain context are much less dynamic compared with nodes associations in large networks which are very dynamic, we can conclude that the traditional and shallow skip-grams approaches commonly used for NLP applications are insufficient to be applied to complex networks.

In summary, the above-mentioned methods used shallow network embedding models to solve the data variance problem using trait information during the modeling process. However, because the underlying patterns of the attributable network are very complex and non-linear, the limited representation capabilities of the shallow units cannot reflect the complex patterns of the attributed networks. To address this problem, several recent studies proposed frameworks based on deep learning for node representation in attributed networks in order to capture the complex patterns of nodes in attributed networks [8], [32]. Based on a recursive Neural Network (NN), A graph neural network (GNN) model was proposed based on a recursive NN [33]. In this model, a transfer function is used to transfer the graph or its vertices to an m-dimensional Euclidean space. Several deep GNN methods, such as graph convolutional networks [34], [35], [36], graph attention networks [37], [38], and graph auto-encoders [39], [40], [41] have been presented.

From the above, it is clear that each of the proposed embedding methods mainly deals with one or two of the challenges of network embedding. In this study, we extended the proposed network embedding model, DANE-WLA, in [42]. We propose a mathematically proven novel network embedding model that addresses the main challenges of the network embedding process in an effective and balanced manner.

The main contributions in this paper are summarized as follows:

- We present a deep network embedding model (ANE-WLDSG) consisting of three stages. In the first stage, we used a modified Weisfeiler-Lehman (WLIA) schema to capture the integration of links and attributes. In the second stage, a training group was created based on random walking. In the third stage, a Deep Skip-Gram (DSG) is trained to extract the high non-linearity between the nodes and embed them in a low-dimensional space.
- We proposed an improved Weisfeiler-Lehman schema that integrates the attributes of the nodes with the attributes of their neighboring nodes with K iterations. We proved by mathematical induction that the improved Weisfeiler-Lehman model preserves all k-order proximity between nodes, where $1<=k<=K$.
- We proposed a new function to improve the flow of information in the WL schema, called the Rectified Aggregate Element (ReAE), and used it as an activation function for the last layer in the autoencoder.
- The novel DSG model involves learning to combine structural regularity in a short random walk with node attributes within an attributed network. This innovative approach aims to improve the performance of network embedding by leveraging the strengths of both deep autoencoders and the Skip-Gram model.

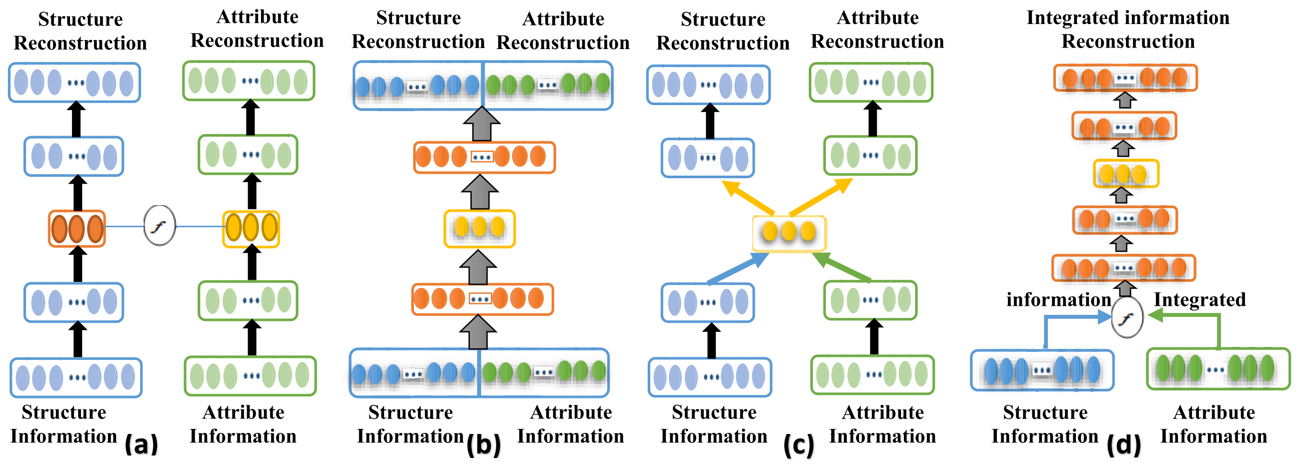


FIGURE 1. **Four modes of attribute network embedding models that use autoencoder. (a) Separate modes. (b) coupled modes. (c) Joint Bottleneck modes. (d) Combined modes.**

The proposed model was experimentally evaluated on four datasets, including node classification, link prediction, and visualization, to determine whether it is applicable to real-world networking tasks. The results showed that ANE-WLDSG is a significant and reliable network embedding method that outperforms several other recent network embedding techniques, even when the test factors differ greatly.

II. RELATED WORK

The following is a brief summary of the relevant work, which is divided into three major categories as follows in the following subsections.

A. NETWORK EMBEDDING BASED ON WEISFEILER-LEHMAN SUBTREE KERNEL

The Weisfeiler-Lehman graph kernel framework [25] is an effective graph kernel method based on the Weisfeiler-Lehman idea of testing the isomorphism of graphs [43], which adopts the renaming of vertices iteratively by collecting neighborhood information. The Weisfeiler-Lehman subtree kernel has attracted considerable interest in the field of graph kernels. The relabeling operation of the Weisfeiler–Lehman algorithm can be considered as a neighborhood aggregation scheme. Neighborhood aggregation algorithms are based on the principle that each vertex receives messages from its neighbors and uses them to update its representation [44]. Yang et al. [16], [45] introduced the concept of a Weisfeiler-Lehman matrix to capture the interplay between a node's structure and its attributes. Ma et al. [46] built upon this idea with their own model based on Weisfeiler-Lehman Graph kernels, which combines the attributes of nodes with those of their neighborhoods. Most graph neural networks use Weisfeiler-Lehman as a pooling function of node features with those of neighboring nodes [10], [35].

The above studies showed that the Weisfeiler-Lehman diagram is able to capture the integration of links and attributes. However, our study aims to address the remaining question: What is the effect of each node in the Weisfeiler-Lehman subtree on the root node during the integration process, and what is the most appropriate equation to balance and fair this effect?

B. NETWORK EMBEDDING BASED ON RANDOM WALK

A random walk is a process that specifies a path in a mathematical space that includes a series of random steps [47]. This approach can construct node sequences while maintaining the original relationships of the nodes [8]. DeepWalk [26] is the first node embedding method inspired by the Skip-Gram word embedding model, which generates word embedding by implicit analysis of an array of mutual information in a corpus of text. The sequences of the nodes were created as a corpus using random paths starting at each node in the network. Node embedding is then obtained by passing these sequences into a Skip-Gram model. By modifying these parameters, Node2vec [27] created a random walk sampling method that can sample networks with a preference for depth-first sampling or breadth-first sampling. Dozens of nodes embedding methods have been proposed based on the DeepWalk concept [3], [28], [48], [49], [50]. It appears that above methods utilized network architecture but did not take into account the valuable features of nodes when representing them in an embedding space.

TADW [30] is a proposed algorithm for embedding attributed networks based on the DeepWalk model, which considers the text information of the nodes. The MMDW [51] model takes advantage of the information accompanying the contract to improve the inclusion process, along with its dependency on DeepWalk. MUSAE [31] uses an approach similar to that of Skip-gram to build an embedding algorithm for the attributed networks by capturing the node information from the attributes of its neighboring nodes and through

TABLE 1. Attributed network embedding models based on autoencoder.

Method	Year	Mode	Technique	Time complexity
DANE**[39]**	2018	Separate	Uses a separate double autoencoder, and uses negative sampling strategy to integrate the representation of nodes attributes and the representation of the network structure	$O(n^2)$
UWMNE**[52]**	2018	Combined	Suggests embedding nodes by inserting a modular matrix and a Markov attribute matrix into the deep autoencoder.	$O(n^2)$
SNE**[4]**	2018	Joint Bottleneck	Uses an autoencoder for the attributes of the nodes and an autoencoder for the network structure, which are combined by a multi-layer NN structure. The last layer is the probability value of all nodes relative to the target node.	$O(n^2)$
DANE Richang et al. **[54]**	2019	Combined	Uses Personalized PageRank model for catch different proximity levels, then merge the proximity matrices by taking the mean, and stacked denoising autoencoder for embedding and capture nonlinearity.	$0(n^2)$
Net-VAE **[53]**	2019	Coupled	Uses shared encoder and Two decoding units with an independent mechanism for each unit.	$0(n\ \ )$
NETTENTION**[55]**	2019	Separate	Uses RNN autoencoder for network structure, and MLP autoencoder for node attributes. To integrate the two representations into a common space, the two representations are passed through an adversarial regularized self-attentive approach.	$0(n + m)$
IINE**[56]**	2019	Coupled	Uses two shallow autoencoders, one for node attributes and one for network structure, feeding a coupled autoencoder integrate the structure and attribute information	$0(n\ \ )$
NEATVGA**[57]**	2022	Coupled	Passes the adjacency matrix, and the preprocessing attribute information using Metropolis-Hastings random walk algorithm, and Doc2Vec model through GCN encoder. Then use two separate decoders to reconstruct both adjacency matrix and attributes matrix.	$0(n^2)$
FSADA**[58]**	2022	Joint Bottleneck	Uses two encoder units one for structure and other for attributes, and use cosine similarity to measure the node closeness and use it to enhance their model.	$0(n^2)$
DANE-WLA**[42]**	2022	Combined	Uses Weisfeiler-Lehman schema to integrate node's attributes and links, then uses deep autoencoder for embedding and capturing nonlinearity.	$0(n + m)$

i

random walks. These studies have shown promising results in many tasks related to analyzing complex networks, however they are designed to work with static graphs and may not be effective in capturing temporal changes or the dynamic evolution of complex networks.

C. NETWORK EMBEDDING BASED ON AUTOENCODER

Autoencoders and their derivatives are deep learning models that are most widely used for feature representation and have been used in a variety of network-based applications. FIGURE 1 shows various modes of models that depend on the autoencoder. The network structure and node attributes were reconstructed using a separate double autoencoder in the DANE) [39]. The UWMNE model [52] suggests embedding nodes by inserting a modular matrix and Markov attribute matrix into the deep autoencoder. SNE [4] is an attributable social network embedding model that learns to represent nodes by maintaining structural proximity and attribute proximity. Net-VAE [53] was proposed to represent both the network topology and node attributes in the same encoder for information integration and learning transmission. Richang et al. [54] proposed an integrated framework consisting of three processes that address data variation, graph topology, and node attributes for attribute network embedding. NETTENTION is a proposed model [55] that uses deep autoencoders to capture the high non-linearity in both network structure and node attributes. The first autoencoder is a discrete recurrent neural network (RNN) autoencoder that learns the representation of nodes within the network topology. The second autoencoder is a multilayer perceptron (MLP) autoencoder for learning the distribution in the attribute spaces. The two representations are passed into an adversarial regularized self-attentive approach to integrate them into a common space. In [56], the authors proposed an IINE model based on three encoders: a basic coupled autoencoder, and two auxiliaries conventional autoencoders. The coupled autoencoder trains node representation by seamlessly combining the network topology and node attributes, while one auxiliary autoencoder is used to probe the internal property of the network structure, and the other auxiliary autoencoder is used to search for data inherent in the attributes of the nodes; then, the main autoencoder is fed with the auxiliary autoencoder outputs. The algorithm's [57] fundamental idea of the NEATVGA algorithm is to pre-process the node attributes first and then input the learning vectors along with the adjacency matrix into the variational graph autoencoder. FSADA [58] is an autoencoder-based framework proposed for integrating network structures and node features. The DANE-WLA model [42] consists of two stages, in which the Weisfeiler-Lehman schema is used in the first stage to integrate attributes and links between the nodes, while the second stage uses a deep autoencoder to exploit nonlinear patterns by applying it to the proximity matrix. TABLE 1 summarize the above models.

Above studies have demonstrated the widespread use of deep autoencoder methods for embedding attributed networks, as they can capture the relationship between node attributes and network structure. However, these methods also have some limitations and disadvantages:

Firstly, deep autoencoders can be computationally expensive. Secondly, they may not be effective in modeling noisy or incomplete data or in adapting to dynamic network changes. Lastly, the high-dimensional embeddings produced by deep autoencoder methods may be difficult to interpret. Therefore,

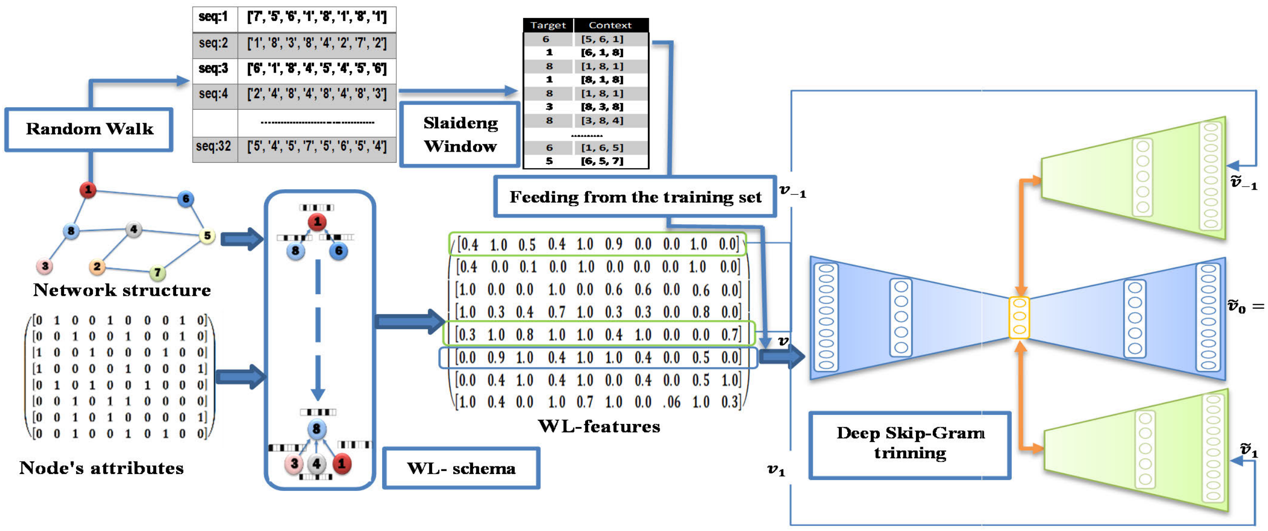


FIGURE 2. An illustration of the stages of training the ANE-WLDSG model: the first stage is to integrate the attributes of the nodes with their neighbors using modified Weisfeiler-Lehman schemes, the second stage is to build the training set using random walking, and the third stage is to train the novel DSG model using inputs from the first two stages.

while deep autoencoder methods are promising, their limitations and challenges should be taken into account when applying them to real-world problems.

The objective of this study is to develop a model that improves the performance of network embedding by combining the advantages of both deep autoencoders and the Skip-Gram model. By leveraging the strengths of these two models, the proposed approach aims to capture both the structural and attribute information of nodes in the network, while also addressing some of the limitations of previous embedding methods.

III. THE PROPOSED MODEL

This section introduces the proposed network embedding model. We list some notations and definitions used in Section III-A are first presented. Then, ANE using an improved Weisfeiler-Lehman schema and a novel Deep Skip-Gram (ANE-WLDSG) model is described in detail in Section III-B.

A. NOTATIONS AND DEFINITIONS

Assume that the attributed network can be represented as an undirected graph $G(V, E, P)$, where:

$V = \{v_i\}_{i=1}^{n}$ is a set of n network nodes, $E = \{e_j\}_{j=1}^{m}$ is a set of m network edges, and $P = \{p(v_i)\}_{i=1}^{n} \in R^{n\times t}$ is a set of n vectors representing node attributes, and t is the attribute dimension. Also, let $A = [a_{i,j}]_{i,j=1}^{|V|} \in R^{n\times n}$ represents the adjacency matrix, where etch element $a_{i,j}$ is given by:

$$a_{i,j} = \begin{cases} 1 & when\ e(v_i, v_j) \in E \\ 0 & otherwie \end{cases} \tag{1}$$

Definition 1 (Attributed Network Embedding (ANE)): Given adjacency matrix A, node attributes P, ANE is a function $f(\cdot)$ of re-representing each node within the network at a low-dimensional vector, while preserving the several-order proximity of adjacency and the attribute's proximity to the other nodes:

$$f(A, p) \rightarrow H, \quad H \in R^{n*d} \tag{2}$$

where d is the dimension of the low-dimensional vector and each row vector $h_i \in H$ is the d-dimensional representation of node v_i.

Some characteristics and levels of proximity among the nodes must be considered when designing embedding models to learn vertex representations accurately [59]. Below, we provide the definitions of some common proximity levels.

Definition 2 (First-Order Proximity): The local proximity that preserves the direct relationships between nodes is referred to as first-order proximity. For each pair of nodes (v_i, v_j), if $a_{i,j} = 1$ then there is first-order proximity between v_i and v_j otherwise, the first-order proximity is 0.

Definition 3 (Second-Order Proximity): The second-order proximity between a pair of nodes (v_i, v_j) indicates that there is a common neighborhood between the first and second nodes. For each pair of nodes (v_i, v_j), if $(\mathcal{N}_{v_i} \cap \mathcal{N}_{v_j} \neq \emptyset)$, then there is second-order proximity between v_i and v_j, otherwise the second-order proximity is 0, where $\mathcal{N}_{v_i}$ is the neighbor set of v_i.

Definition 4 (k-Order Proximity): The k-order proximity between two pairs of nodes (v_i, v_j) occurs if and only if there is at least one shortest path of length k between v_i and v_j.

Definition 5 (Weisfeiler-Lehman Scheme):

Let $G = (V, E, lb)$ be a graph with initial node labels lb. Let $lb^0(v_i) = lb(v_i)$ for each $v_i \in$ V and K be the Weisfeiler-Lehman iteration amount; for all $0 \le k < K$ recursively computes the new label of v_i in iteration $k+1$ by:

$$lb^{k+1}(v_i) = relable(lb^k(v_i), \left[lb^k(u) : u \in \mathcal{N}(v_i)\right]\Big) \tag{3}$$

where $\mathcal{N}(vv_i)$ is the neighbour set of v_i.

Definition 6 (Skip-Gram): Skip-gram is an unsupervised machine-learning mechanism that imp.roves a word's representation based on a specified number of other words in the same context.

B. THE PROPOSED ANE-WLDSG MODEL

The architecture of the proposed ANE-WLDS model consists of three main stages, as illustrated in FIGURE 2.

Stage1 (Information Aggregation): In this stage, we propose an improved Weisfeiler-Lehman schema that is defined based on Weisfeiler-Lehman graph kernels. It aims to integrate the attributes of a node with the attributes of its neighboring nodes at the K-level of proximity, and to produce a new representation of the node that preserves its attributes and integrates them with the network structure.

Stage2 (Training Set Building): In parallel with the previous stage, a corpus is built to learn the representation of the attributed network using the random walk method, depending on the network structure. Then, we build a training set that consists of the input node and target nodes that represent the node context with a specific size.

Stage3 (Node Representation Using Deep Skip-Gram): To capture the high non-linearity of each node's patterns, a novel autoencoder is proposed in this study; we call it Deep Skip-Gram (DSG). This autoencoder consists of a single encoder and several decoders representing the number of nodes in the node context window. The autoencoder maps the aggregated information and embeds it into a low-dimensional vector, which can reconstruct the node context as closely as possible. The encoding in the hidden layer represents the node. This autoencoder obtains its input from the outputs of the previous two stages.

In the following, we present the details of each of the above stages.

1) WEISFEILER-LEHMAN INFORMATION AGGREGATION (WLIA)

Weisfeiler-Lehman Information Aggregation (WLIA) is used to aggregate information from neighboring network nodes to a target node. It uses the K parameter to control the adjacent node layers that are involved in the aggregation process. The primary idea of a network with attributes P is to generate a propagation subtree that captures the current node's attributes and combines them with the attributes of its adjacent nodes in balanced proportionality. FIGURE 3 shows the propagation subtree for a specific node of the network, and the steps of the WLIA scheme are presented in Algorithm 1.

Definition 7 (Weisfeiler-Lehman Information Aggregation):

Let $G = (V, E, P)$ be a graph with initial node attributes $P^{(0)}(v) = P(v)$ for each $v \in$ V, and let K be the number of Weisfeiler-Lehman iterations. In iteration k+1, the new attributes of v are computed as follows:

$$P^{k+1}(v) = aggregation(P^k(v), \left[p^k(u) : u \in \mathcal{N}(v)\right]) \quad (4)$$

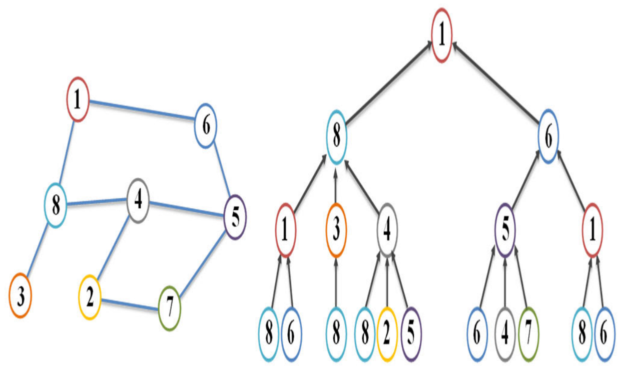


FIGURE 3. Illustrating the WL sub-tree propagation of node (1) according to network G.

Algorithm 1 WLIA Information Aggregation

```
Input:
Attributed Network: $G(V, E, P)$,
parameters: $WL$ iteration K.
Output:
$WL$ features $WL\_f \in R^{|V|*t}$,
Begin
$P^0 = P$
  for $k = 1$ to $K$
    for $v$ in $V$
    $P^k(v) = P^{k-1}(v)$
    for $u$ in $\mathcal{N}(v)$
$P^k(v) = ReAE(P^{k-1}(v) + \frac{1}{\sqrt{d(v)*d(u)}} P^{k-1}(u))$
return $P^K$
```

For all $0 \leq k < K$.

We derive an aggregation function based on the Laplacian matrix element because the Laplacian matrix has the ability to preserve the local topological properties of the graph (first-order proximity).

Given a network $G(V, E, P)$ with an adjacency matrix A, the Laplacian matrix L of A is given by

$$L = D - A \quad (5)$$

where $D = diag(d_1, d_2 \ldots, d_n)$ is the degree matrix of A, with $d_i = \sum_{j=1}^{n} a_{i,j}$ is the degree of each node v_i.

Meanwhile, the signless Laplacian matrix is calculated as $L^+ = D + A$. For the undirected graph G, the normalized signless Laplacian matrix is defined as [60]:

$$L^+ = D^{-1/2} A D^{-1/2} = I + D^{-1/2} A \quad (6)$$

The elements of L^+ are given by:

$$L^+_{i,j} = \begin{cases} 1 & if\ i = j \\ \dfrac{1}{\sqrt{d_i \times d_j}} & if\, i \neq j\ and\ e\left(v_i, v_j\right) \in E \\ 0 & otherwise \end{cases} \quad (7)$$

To integrate the attributes of a node with the attributes of its neighbor nodes, the Laplacian matrix is multiplied by the attribute matrix as:

$$P^{(1)} = L^{+} P^{(0)} \tag{8}$$

where $P^{(1)} \in R^{n \times t}$ is a matrix of node attributes aggregated with attributes of nodes with first-order proximity, and each i row in the matrix expresses the modified attributes of node v_i. Thus, each node vector $v_i \in \{v_i\}_{i=1}^{n}$ in $P^{(1)}$ can be represented as follows:

$$P^{(1)}(v_i) = \sum_{j=1}^{n} L^{+}_{i,j} \times P^{(0)}(v_j) \tag{9}$$

Depending on the definition of the Laplacian matrix in Equation (6) and ignoring the zero values, $P^{(1)}(v_i)$ is expressed as:

$$P^{(1)}(v) = P^{(0)}(v) + \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d(v) \times d(u)}} P^{(0)}(u) \tag{10}$$

Equation (10) represents the 1-iteration Weisfeiler-Lehman scheme.

Equation (10) can be generalized for the Weisfeiler-Lehman scheme with K iterations, in which for each $v \in V$, $P^{(K)}(v)$ can be expressed as

$$P^{(K)}(v) = P^{(K-1)}(v) + \sum_{u \in \mathcal{N}(v_i)} \frac{1}{\sqrt{d(v) \times d(u)}} P^{(K-1)}(u) \tag{11}$$

Rectified Aggregate Element (*ReAE*) function is defined to normalize each value of the node attributes after each integrative operation, motivated by the assumption that each node has a probability value of having any attribute. We can define *ReAE* as:

$$ReAE(x) = min(x, 1)\ where\ x \in R^{+} \tag{12}$$

Then, we can define our proposed WLIA as:

$$P^{(K)}(v_i) = ReAA\left[P^{(K-1)}(v_i) + \sum_{u \in \mathcal{N}(v_i)} \frac{1}{\sqrt{d(v) \times d(u)}} P^{(K-1)}(u)\right] \tag{13}$$

Theorem 1: Weisfeiler-Lehman Scheme defined in Equation (11) preserves all k-order proximity where $1 <= k <= K$. Moreover, k-order proximity between two pairs of nodes (v_i, v_j) can be defined as:

$$k\text{-}order\ proximitye(v_i, v_j) = \frac{\binom{K}{k}}{\sqrt{d(v_i) \times d(v_j)} \times \prod_{u_l \in k-\mathrm{path}(v_i, v_j)} d(u_l)} \tag{14}$$

Details of the proof using mathematical inductor of the above theorem is presented in Appendix.

Corollary1: Weisfeiler-Lehman scheme, which is defined in Equation (12), preserves all k-order proximity where $1 <= k <= K$.

Proof:

Because Equation (13) is a normalization of Equation (11), we conclude that whatever holds for Equation (11) also holds for Equation (13), and the range of Equation (13) becomes [0,1].

Corollary2: For each pair of nodes $(u, v) \in V$, the $WLIA - proximity(u, v)$ can be defined as:

$$WLIA - proximity(u, v) = \frac{1}{\sqrt{d(v_i) \times d(v_j)}} \sum_{path \in paths_K(u,v)} \frac{\binom{K}{|\,path\,|}}{\prod_{u_l \in path} d(u_l)} \tag{15}$$

where $paths_K(u, v)$ is a set of all bath between (u, v) with $length <= K$, and $|path|$ is a length of path.

The advantages of the proposed WLIA scheme can be summarized in the following points.

- The Weisfeiler-Lehman scheme efficiently preserves and integrates the attributes of a node target with those of its neighboring nodes, where parameter k represents the number of proximity levels for the neighboring nodes that join the integration.
- The proposed equation for the Weisfeiler-Lehman proximity schema deals fairly well with the levels of proximity and degrees of nodes. Through the coefficient $\frac{\binom{K}{k}}{\sqrt{d(v) \times d(u)} \times \prod_{u_l \in 1-path(v_i,v_j)} d(u_l)}$, it is clear that the influence of child node $\boldsymbol{u}$ is inversely proportional to both the level of proximity and the strength of the node. In other words, the closer the level of the child node to the target node, the higher its impact. In addition, the greater the degree of the child node, the lower is its impact.
- There is a greater influence of any child node on the aggregation process if there are more noncircular paths between the root node and child node.
- The WLIA scheme automatically handles missing data in node attributes by aggregating them from neighboring nodes according to the proposed equation.
- Furthermore, Equation (12) ignores the nodes that appear in the same path more than once and takes into account the first appearance only to avoid the duplication effect of the child nodes on the aggregation of the root node, as shown in FIGURE 4.

2) TRAINING SET BUILDING

To train our proposed sub-model DSG, a corpus is first created by running recursive random walks, starting with a randomly chosen root node v_i and following along with random walks through neighboring nodes until a specified length (τ) is reached, then moving randomly to a new root node and repeating the previous process until all nodes are

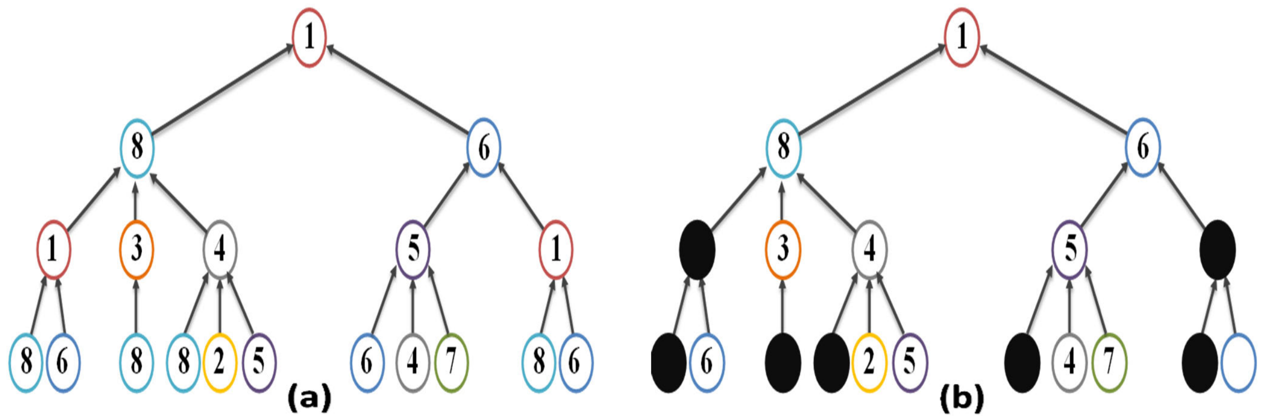


FIGURE 4. Illustration of the difference between WL and WLIA with ReAE, where sub-figure (a) repersent the sub-tree propagation according to WL schema and sub-figure(b) repersent the propagation sub-tree according to WLIA with ReAE funcation, the black color represent the nodes that are neglected in the process of integrating the root node with the child nodes in the propagation sub-tree.

Algorithm 2 Training Set Building

```
Input: $G(V, E, P)$, parameters: walks length $\tau$, number of walks $\gamma$ per node, context size C.
Output: Training set $T$
for $i = 1$ to $\gamma$
    $V$ = Shuffle($V$)
    for $v$ in $V'$
        Append $v$ to walk
        $u' = v$
        for $j = 1$ to $\tau$
            $u = Random$ – neighbor $(u')$
            Append $u$ to walk
            $u' = u$
        Append walk to Corpus
for $i = 1 : size(Corpus)$
    for $j = C + 1 : \tau - C$
        $T'.target = Corpus\ [i][j]$
        $T'.\ context = []$
        for $k = j - C : C + j$
            Append $Corpus\ [i][j]$ to $T'.\ context$
        Append $T'$ to $T$
return $T$
```

completed. The previous process was repeated for a specified amount γ for each node within the network. As a part of network embedding, all network nodes from the corpus are indexed and sorted by their frequency using a sliding window to extract instances. Each of the instances consists of a target node and its context of a fixed size. Algorithm 2 presents the steps involved in building the training set.

3) NODE REPRESENTATION USING DEEP SKIP-GRAM

Skip-Gram has achieved impressive success in text modeling, which is a model of the representation of words by increasing the probability of the word being in a specific context within the sentence. Motivated by the success achieved by the Skip-Gram model, a number of studies have suggested generalizing this model to the re-representation of nodes in complex networks. However, there are shortcomings in the proposed methods owing to the obvious differences between the network structure and text structure, as well as between network dynamics and text dynamics. Based on this, we propose in this paper a novel deep model based on SkipGram to efficiently represent nodes in a low-dimensional space. In this stage, low-dimensional deep embedding vectors are generated from the WLIA information obtained in the first stage using deep learning models. We capture complex nonlinear patterns using an autoencoder that learns the node representations through nonlinear learning and then reconstructs the context of the target node from the deep representation pattern of the target node. DSG architecture contains an encoder and $2C + 1$ decoders, which is defined next:

DSG-Encoder: This consists of Y layers of fully connected NNs as an encoder, which maps each target node v in the training set that is built in the second stage to a low-dimensional nonlinear embedding space. The i^{th} layer output is expressed as follows:

$$h_v^{(i)} = g^i \left(W^{(i)} h_v^{(i-1)} + b^{(i)} \right) \tag{16}$$

where $h_v^{(0)} = WL\_f(v)$, $W^{(i)}$ is the $i-th$ layer weight, $b^{(i)}$ is the bias, and g^i is a nonlinear activation function.

Representation vector $\varphi_v \in \mathbb{R}^d$ of target node v can be defined as:

$$\varphi_v = g^L \left(W^{(Y)} h_v^{(Y-1)} + b^{(Y)} \right) \tag{17}$$

DSG-Decoder: This consists of several parallel decoders that capture the resulting node-embedding vector φ_v of the encoder and rebuilds the context of the target node. The layers of each decoder are the same number and shape as the encoder layers, and in reverse. The output of each layer of each decoder of the context nodes can be represented as:

$$h_{v_c}^{(i)} = g_c^i \left(W_c^{(i)} h_{v_c}^{(i-1)} + b_c^{(i)} \right) \tag{18}$$

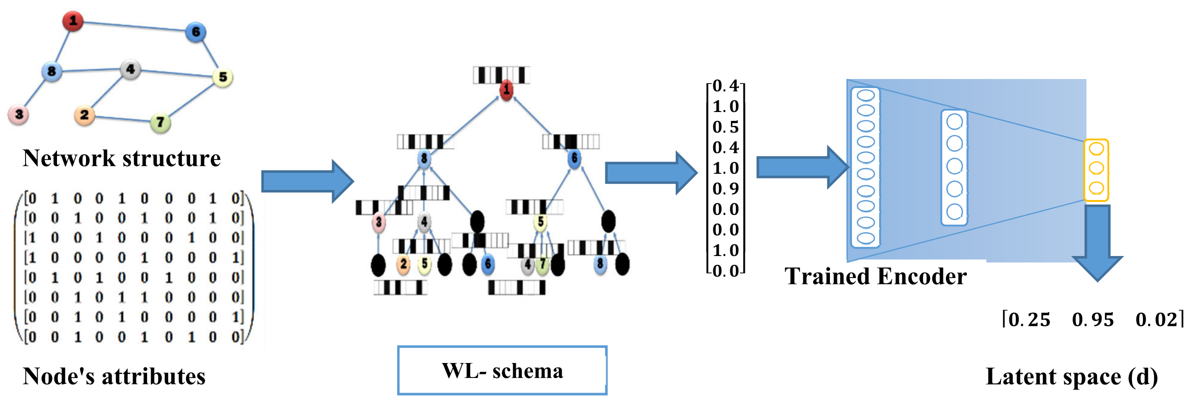


FIGURE 5. Production model of embedding a node in a low-dimensional vector based on the trained model.

where $h_{v_c}^{(0)} = \varphi_v$, $W_c^{(i)}$ is $i-th$ layer weights, $b_c^{(i)}$ is its bias and, g_c^i is a nonlinear activation function of decoder c, and $c \in [-C, C]$. The output of each decoder is defined by the following equation.

$$\tilde{v_c} = ReAE\left(ReLU\left(\left(W_c^{(Y)} h_{v_c}^{(Y-1)} + b_c^{(Y)}\right)\right)\right) \tag{19}$$

We use *ReAA* which was defined earlier as an activation function, so that the outputs are the same as the deep SkipGram inputs. We trained the model to obtain an embedded vector capable of reconstructing the node context. Thus, we aim to maximize the probability of obtaining the node context, depending on the embedding vector.

$$Pr\,(context(v) \mid \varphi_v) = \prod_{v_c \in context(v)} Pr\,(v_c \mid \varphi_v) \tag{20}$$

To obtain the maximum value of the previous conditional probability, we seek to reduce the following loss function.

$$L = \sum_{i=1}^{n} \|\tilde{v} - WL\_f(v)\|_2^2 + \sum_{v_c \in context(v)} \frac{1}{2C} \sum_{i=1}^{n} \left\|\tilde{v}_{c_i} - WL\_f\left(v_{c_i}\right)\right\|_2^2 \tag{21}$$

and

$$L_c = \frac{1}{2c} \sum\nolimits_{i=1}^{n} \left\|\tilde{v}_{c_i} - WL\_f\left(v_{c_i}\right)\right\|_2^2 \tag{22}$$

where $Pr\,(v_c|v)$ is the number of times a node v_c is presented within the context divided by the number of times the target node v_c appears in the training set T.

The model parameters:

$$\left\{W^{(l)}, b^{(l)}\right\}_{i\ in\ Encoder\ layers},$$
$$\left\{\left\{W_c^{(l)}, b_c^{(l)}\right\}_{l\ in\ c\ Decoder\ layers}\right\}_{c\in[-C,C]}.$$

were optimized using Stochastic Gradient Descent (SGD). Algorithm 3 summarizes the training algorithm for the DSG.

Algorithm 3 DSG Training

```
Input: WL features $WL_f$, Training set $T$, $N$ Number of epochs, $\beta$ batch size, $C$ context size.
Output: Trained encoder parameters $\{W^{(l)}, b^{(l)}\}_{l\ in\ Encoder\ layers}$
Begin
for $i = 1 : N$
    $T' = \text{Shuffle}\,(T)$
    for $e = 1$ to $size(t)/\beta$
        Predict $\varphi$ using equations 16,17
        for $v_c \in context(v)$
        Predict $\tilde{v}_c$ using equations 18,19
        Compute Lost $(L_c)$ using equation 21
      Cost=mean $(L_c)$
      Update $\{W_c^{(l)}, b_c^{(l)}\}_{l\ in\ Decoder\ layers}$ using SGD
    Update $\{W^{(l)}, b^{(l)}\}_{l\ in\ Encoder\ layers}$ using SGD
return $\{W^{(l)}, b^{(l)}\}_{i\ in\ Encoder\ layers}$
```

A comparison between DSG and Skip-Gram can be made by examining the differences between their structures, parameters, and characteristics. The main differences are listed in TABLE 2.

C. NODES EMBEDDING USING THE TRAINED MODEL

To embed any node in the network in the latent space, we first apply the proposed WLIA to this node to capture its aggregation information based on the node's propagation sub-tree by recurrently executing Equation 12. We then passed the information from WLIA to the pre-trained DSG encoder. The node-embedding process utilizing the learned model is illustrated in FIGURE 5.

D. TIME COMPLEXITY

The proposed ANE-WLDSG framework consists of three stages: aggregation of information, construction of training sets, and DSG. Our first part of the study utilized a Weisfeiler-Lehman model whose time-complexity was

TABLE 2. The main differences between DSG and skip-gram.

	Skip-Gram	DSG
Domain	Language modeling and generalized to complex networks embedding	ANE
Structure	Shallow	Deep
Input	one-hot vector	continuous vector
Latent space	$\Phi \in \mathbb{R}^{n*d}$	$\varphi_v \in \mathbb{R}^{d}$
Activation Function	Hierarchical SoftMax vector	Rectified Aggregate Element vector (ReAE)
Transfer learning	No	Yes
Dynamic	The representation of the node is fixed and is not affected by its activity after the training process.	The representation of the node is affected by the change in the network structure. Depending on the trained model, FIGURE 5 illustrates how to represent any node within a network in a low-dimensional vector.

$O(K \times m)$, where m is the number of edges in the network, and K is the number of iterations of Weisfeiler-Lehman. For the stage of constructing the dataset by random walk, the time complexity is $O(n \times \gamma \times \tau)$, where n represents the number of network nodes, τ is the walk length, and γ is the number of walks per node. We utilized an autoencoder with a time complexity of $O(\theta \times n)$, where θ is the dimension of the largest hidden layer. By neglecting constant values $(K, \gamma, \tau, and \theta)$, the ANE-WLDSG time complexity is $O\,(n + m)$.

The linear complexity of the proposed model means that its runtime will increase linearly with the size of the input data. This makes it a good candidate for larger networks, as the runtime will not increase exponentially as the size of the network increases.

IV. EXPERIMENTS

To determine whether the proposed model is effective and efficient in ANE, we applied it to real-world networks and compared it with other well-known models.

A. BENCHMARK DATASETS

The following four datasets, which are freely available on SNAP [61], were used in our experiments to evaluate the proposed model.

LastFM Asia: Users of the LastFM social network living in Asia are connected via their mutual follower relationships, and their favorite musicians are represented by their vertex features. The machine-learning challenge is the classification of users' nationality.

GitHub: Developers are represented as nodes, while the relationships between them and their followers are represented as links. Attributes such as location, metadata, and biography are used to calculate information about each developer, and they are classified into categories based on whether they are machine learning developers or web developers.

Facebook: The task was to classify the websites into several categories. The nodes correspond to official Facebook pages. The edges represent the likes exchanged between the pages. Extract node features from the page descriptions.

TABLE 3. Summary statistics of the benchmark attributed networks.

Datasets	Number of neurons in each layer
FACEBOOK	4714-1000-500-128
WIKI	13183-4000-1000-128
GITHUB	4050-1000-500-128
LastFM	7842-1000-500-128

Wikipedia: There are nodes representing Wikipedia articles, edges representing interconnected links, a binary target variable showing the amount of traffic on a website, and vertex features that specify whether there are nouns in the article.

Specification information of these dataset networks are outlined in TABLE 3.

B. BASELINES

The following are the descriptions of seven popular models that were compared with the proposed model (DANE-WLA):

HOPE [22]: This network-embedding model used here is a simplified version that employs extended singular value decomposition instead of an adjacency matrix.

DeepWalk [26]: Using samples taken during random network walks, embedding is calculated by predicting the immediate surroundings of the nodes, and the walking trails are analyzed using Skip-Gram.

Node2vec [27]: This is a generalized form of deep walk that combines depth-first search (DFS) with breadth-first search (BFS) to connect nodes in a graph.

TADW [30]: TADW uses textual features to supervise random walks on graphs and integrates node-content information.

BANE [62]: This model embeds network nodes by identifying the relationship between a node's attributes and network architecture. A proximity matrix is built by combining the attributes and links of adjacent nodes. The node embedding in the binary space is also depicted using cyclic coordinate descent (CCD).

TENE [63]: Using this method, ANE problems are subdivided into mutually non-negative matrix factoriza-

TABLE 4. Detail of the encoder structure for all datasets.

Dataset	LastFM	GitHub	Facebook	Wikipedia
No. of Nodes	7624	37,700	22,470	11,631
No. of Edges	27,806	289,003	171,002	170,918
No. of Attributes	7842	4,005	4,714	13,183
No. of Classes	18	2	4	2

TABLE 5. Hyperparameters of WL schema and DSG models.

Parameter	Notation	Value
WL iterations	K	4
walks length	τ	40
walks per node	γ	10
context size	C	2

tion problems, and their organization is based on node embeddings.

MUSAE [31]: MUSAE learns how to represent attributed nodes to factorize the creation of an attributed matrix and a normalized adjacency matrix.

FSADA [58]: FSADA is an autoencoder-based framework that integrates network structures and node features. It uses two encoder units, one for the structure and one for the attributes. The structure encoder learns a latent representation of the network structure, while the attributes encoder learns a latent representation of the node features. The two latent representations are then combined using cosine similarity to measure the node closeness.

DANE-WLA [42]: An approach for representing attributed networks involves a two-stage model. In the first stage, the attributes of nodes and their links are combined to create an integrated representation. This can be achieved using techniques such as concatenation or graph neural networks. In the second stage, a deep autoencoder is used to map the integrated representations to a lower-dimensional space.

C. PARAMETER SETTINGS

The proposed model required several control parameters to be determined in advance.

TABLE 4 presents the size of each encoder layer for each dataset, whereas TABLE 5 presents the hyperparameters applicable to the training process of the embedding model. Each baseline was constructed using Karate Club's [61] public source code using settings chosen to provide the best results across all datasets and trials.

The DSG autoencoder and discriminator were optimized using the Adam optimizer [64]. The TensorFlow deep learning tool was also used with a learning rate of 0.001 and ReEA activation function. We assure readers that the code for the ANE-WLDSG model is available for open access through the link https://github.com/amr-furas/ANE-WLDSG.

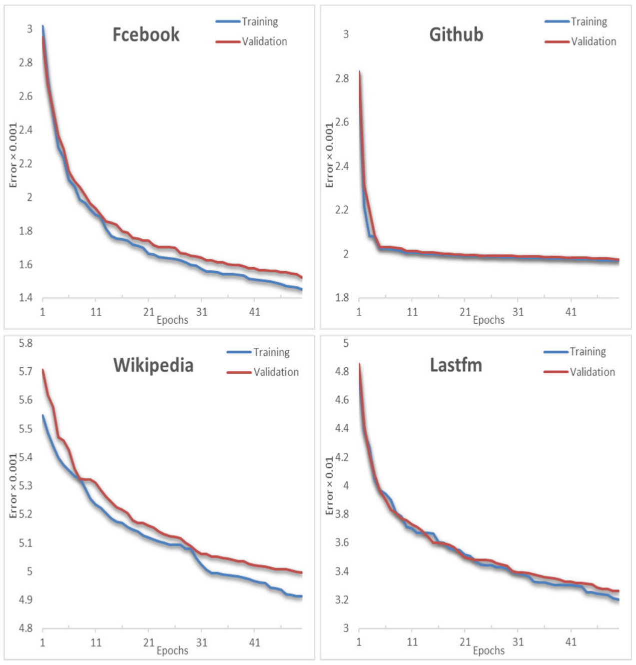


FIGURE 6. Training loss and validation loss of ANE-WLDSG over 50 iterations.

D. EXPERIMENTAL RESULTS

In this section, we present the results of evaluating the performance of the proposed embedding model (ANE-WLDSG). Specifically, Figure 5 illustrates the relationship between the training error and the validation error throughout the DSG training process.

The analysis of the loss function based on the FIGURE 6 reveals that both the training and validation errors decrease over the iterations, indicating that the model is learning and improving its performance. The convergence of the errors suggests that the model is stabilizing and approaching a point of minimal error. The small difference between the training and validation errors suggests that the model is not severely overfitting the training data. However, further evaluation using additional metrics is recommended to gain a more comprehensive understanding of the model's performance. Overall, the analyzed loss function demonstrates effectiveness in reducing errors and the model shows promising learning and generalization capabilities.

In the following experiments, we demonstrate the efficiency of our proposed model against a set of recent embedding models. We do this by applying and analyzing our model on node classification, link prediction, and visualization tasks. The results show that our model is effective and superior to the other models.

1) NODE CLASSIFICATION RESULTS

Classifying network nodes according to their labels based on the network's available information is a common task for determining the effectiveness of network embedding. In this experiment, the node information was first embedded in low-dimensional vectors, and the nodes were then

TABLE 6. Facebook, GITHUB, LastFM, and Wikipedia node classification result, as evaluated by Macro-F1, and Micro-F1 metrics.

Dataset	Model	Macro-F1 (%)					Micro-F1 (%)				
		10	30	50	70	90	10	30	50	70	90
FACEBOOK	HOPE	61.58	64.71	65.2	66.4	68.35	54.7	58.65	59.78	61	62.45
	DeepWalk	63.64	68.87	69.77	69.26	70.72	60.8	66.83	67.14	67.02	67.62
	Node2Vec	65.39	69.44	69.7	70.7	70.01	63.06	67.08	67.55	68.7	67.55
	BANE	60.39	64.02	64.03	64.74	63.73	48.96	57.42	57.13	58.42	57.61
	TENE	59.65	65.73	66.52	66.89	67.42	56.52	62.41	63.12	63.42	64.85
	TADW	63.04	64.62	64.74	65.38	64.35	59.73	60.75	60.79	61.59	59.96
	MUSAE	88.48	89.49	89.75	90.35	90.74	87.4	88.56	88.87	89.63	89.86
	FSADA	88.32	88.87	89.01	89.23	89.94	86.61	87.08	87.95	88.23	88.72
	DANE-WLA	89.28	92.89	93.46	93.65	94.35	89.28	92.89	93.46	93.65	94.35
	ANE-WLDSG	**90.13**	**93.29**	**93.86**	**93.92**	**94.42**	**90.13**	**93.01**	**93.66**	**94.15**	**94.85**
GITHUB	HOPE	61.49	64.47	64.88	66.87	66.17	78.7	79.42	79.72	80.51	79.95
	DeepWalk	77.16	77.36	78.2	78.18	77.7	84.41	84.49	84.94	84.94	84.48
	Node2Vec	46.44	77.34	78.28	78.22	77.8	84.18	84.4	85.08	85.08	84.85
	BANE	42.6	42.58	42.58	42.51	42.31	74.21	74.14	74.15	73.95	73.34
	TENE	76.65	76.81	76.9	76.29	78.31	83.62	83.77	83.64	83.32	85.09
	TADW	74.31	75.35	75.91	75.79	76.49	83.04	83.64	83.6	83.61	83.9
	MUSAE	80.15	80.22	80.32	**80.31**	79.23	86.16	86.13	86.31	86.42	85.52
	FSADA	79.21	79.37	79.35	79.78	**80.81**	86.06	86.20	86.21	86.52	86.62
	DANE-WLA	**81.43**	**80.53**	**80.63**	80.21	79.60	**86.65**	**86.63**	**86.74**	86.22	85.94
	ANE-WLDSG	79.6	80.21	80.37	**80.43**	79.84	85.94	86.22	**86.74**	**86.63**	**86.65**
LastFM	HOPE	78.82	79.59	80.11	79.83	82.82	69.19	70.14	70.8	68.43	72.74
	DeepWalk	75.32	79.87	78.93	80.63	82.41	75.01	79.16	77.64	81.22	80.36
	Node2Vec	77.43	80.6	82.19	80.81	80.81	76.79	79.82	81.03	79.48	79.73
	BANE	77.08	76.86	78.4	77.86	81.03	65.65	75.54	67.51	71.59	70.46
	TENE	78.34	81.31	81.48	79.64	82.23	76.71	78.53	77.36	74.31	80.13
	TADW	76.99	78.96	80.98	81.04	83.48	76.36	78.6	79.65	80.01	82.23
	MUSAE	78.82	81.79	80.9	81.74	82.54	77.95	80.54	79.15	80.04	80.7
	FSADA	78.71	81.57	80.57	81.30	81.98	77.84	80.32	78.82	79.60	80.14
	DANE-WLA	79.01	80.95	81.25	83.25	84.26	78.16	79.93	80.13	82.27	83.14
	ANE-WLDSG	**80.55**	**83.73**	**82.19**	**85.23**	**86.67**	**79.05**	**82.65**	**81.97**	**84.44**	**85.79**
WIKI	HOPE	64.59	68.19	70.42	72.18	70.7	78.31	79.23	80.62	81.35	80.58
	DeepWalk	74.51	80.21	84.56	86.39	86.61	81.75	85.28	88.08	89.31	89.86
	Node2Vec	76.06	81.31	84.47	85.32	88.54	82.13	85.97	88.17	88.73	89.06
	BANE	41.7	41.69	41.67	45.84	47.07	71.53	71.5	71.44	72.46	72.68
	TENE	72	77.76	77.85	80.38	82.25	80.6	83.92	83.85	85.67	86.85
	TADW	77.65	79.94	80.74	80.57	80.88	84.12	85.21	85.57	85.73	85.48
	MUSAE	81.59	86.81	86.8	87.28	86.47	85.34	88.21	88.47	88.85	89.43
	FSADA	81.52	86.68	86.60	87.01	86.14	85.27	88.08	88.27	88.58	89.10
	DANE-WLA	81.57	86.76	86.99	87.56	88.55	**85.73**	88.46	89.05	89.4	89.86
	ANE-WLDSG	**81.69**	**87.74**	**86.86**	**87.75**	**88.64**	**85.73**	**88.75**	**89.67**	**89.85**	**90.01**

divided into training and test sets. We trained a support vector machine (SVM) classifier using the training set. Then, we used Micro-F1 and Macro-F1 as the evaluation metrics. The experiment was repeated several times, using different ratios as the training set. Specifically, 90 percent, 70 percent, 50 percent, 30 percent, and 10 percent of the nodes were selected randomly as the training set and the remaining percentage as the testing set. Macro − F1 and Micro-F1 can be expressed by the following equations:

$$Macro - F1 = \frac{\sum_{j=1}^{c}\left(\frac{TP_j}{TP_j+\frac{1}{2}(FP_j+FN_j)}\right)}{C}$$
$$Micro - F1 = 2 \cdot \frac{recall \times precision}{recall + precision}, \tag{23}$$

where C represents the number of classes in the test set and TN_j, TP_j, FN_j, and FP_i are the true negative, true positive, true negative, and false positive of class j, respectively. Recall and precision metrics are computed as follows:

$$recall = \frac{\sum_{j=1}^{C} TP_j}{\sum_{j=1}^{C}(TP_j + FP_j)},$$
$$precision = \frac{\sum_{j=1}^{C} TP_j}{\sum_{j=1}^{C}(TP_j + FN_j)} \tag{24}$$

For the aforementioned datasets, TABLE 6 displays the results for each of the applied network embedding models. In each setting, the Macro-F1 and Micro-F1 values with the best results are highlighted in boldface, while the second-best performers are highlighted in underlined.

According to these results, comparing different methods for Plain network embedding, DeepWalk performs similarly or better than Node2vec across various datasets. Both methods, based on random walks, outperform HOPE, as they can capture diverse information about the network's structure. However, when comparing methods that integrate node attributes with those that only use the network structure, the ANE methods tend to achieve better results. Overall, these results highlight the advantages of incorporating both structural and auxiliary information for learning node embeddings in attributed networks. In particular, ANE-WLDSG and DANE-WLA appear to be the most effective approaches for combining both structural and attribute information, achieving the highest Micro-F1 and Macro-F1 scores among all tested methods in most cases. For example, on the FACEBOOK dataset, ANE-WLDSG outperforms other methods by a margin of 3%. These results highlight the efficiency and robustness of ANE-WLDSG and DANE-WLA for handling complex networks with both structural and attribute features. In general, the ANE-WLDSG method outperforms DANE-WLA across all datasets except for GITHUB. This suggests that incorporating the Deep Skip-Gram (DSG) model is beneficial for integrating the network structure with node attributes and improving the performance of the deep autoencoder. The superior results of ANE-WLDSG highlight the impact of combining WLIA with DSG in learning effectively attributed node embeddings.

2) LINK PREDICTION

This is one of the most prominent applications of network embedding, and is devoted to inferring the likelihood that edges might exist between nodes in the input network that are not connected to each other at the present time. To build a test set, we randomly selected 20% of the existing edges and combined them with an equal number of non-existent edges. First, the network nodes are projected onto a low-dimensional space, and the connections between nodes are represented based on these projections. Next, a One-class classifier is trained using these representations as the training set. Subsequently, we evaluate the absence of connections to identify any missing links or potential future links. To measure the novelty of the connections, we employed the Local Outlier Factor (LOF) algorithm [65], [66]. Subsequently, the AUC and precision metrics were measured to evaluate the link prediction process. The AUC is the average accuracy over the entire range of test values, defined by the area under the receiver-operating characteristic curve. To find the first μ position after sorting the unknown edges descending by precision, we used the precision index for link prediction. The prediction precision if we have ϵ edges in the test set are represented by

$$Precision = \frac{\mu}{\epsilon} \tag{25}$$

Based on the results obtained by applying the proposed embedding model and the other evaluated models listed in TABLE 7. Overall, the results show that ANE-WLDSG performs the best, achieving the highest precision and ACU scores on three out of the four datasets. DANE-WLA also performs well, achieving the highest precision and ACU scores on one dataset.

DeepWalk and Node2Vec also perform well, with high precision and ACU scores on most of the datasets when compared with HOPE, BANE, TENE, and TADW. These methods are based on biased random walks and can capture diverse structural information of the network.

In summary, ANE-WLDSG, DANE-WLA, and MUSAE are the most effective methods for link prediction on these datasets, while the other methods may not be as effective.

This highlights the importance of considering both the network structure and node attributes in predicting linkages in attributed networks. By leveraging both types of information, the link prediction models can capture more comprehensive knowledge about the network and improve their accuracy.

3) VISUALIZATION

Visualization is a form of network analysis that provides insights into the quality and efficiency of an embedding model. An embedding model can be evaluated by visualizing how nodes of the same type are condensed in the embedding space, where more condensed nodes in a smaller area refer to a higher degree of similarity of these nodes. In other words, points representing nodes with the same label are clustered together, and the closer these nodes are to one another, the better is the embedding model performance. The network visualization in FIGURE 7. depicts the distribution of generated points with different embedding methods applied to the Facebook dataset, where four clusters are depicted in four different colors for each model.

Based on the findings presented in Figure 6, it appears that node embedding methods that rely solely on network structure generally yield poor visualization results, as they fail to accurately identify distinct clusters within the network. While TENE, TADW, and BANE perform better than traditional embedding methods in terms of visualization results, there is still a significant overlap between different groups of nodes.

In contrast, ANE-WLDSG, DANE-WLA, and MUSAE produce more compact visualizations, with nodes from each cluster aggregating in close blocks and minimal overlap between clusters. Among these methods, ANE-WLDSG in particular stands out for its ability to generate highly distinct and well-separated groups.

Overall, these results suggest that incorporating both network structure and node attributes can enhance the ability of node embedding methods to accurately identify and visualize distinct clusters within complex networks.

4) EVALUATING ANE-WLDSG EFFECTIVENESS IN HANDLING UNSEEN DATA

In this experiment, the objective was to assess how well the proposed method could handle the unseen data in the network

TABLE 7. A performance analysis of link prediction with different models on GITHUB, Wikipedia, LastFM Asia, and Facebook.

Model	GITHUB		WIKI		LastFM		FACEBOOK	
	Precision	ACU	Precision	ACU	Precision	ACU	Precision	ACU
HOPE	64.36	56.49	82.35	78.88	79.25	77.14	65.05	57.76
DeepWalk	85.01	82.13	95.56	90.81	85.72	83.15	83.64	80.85
Node2Vec	85.44	82.52	95.88	90.99	85.64	82.89	79.7	76.91
BANE	60.81	50.6	76.13	73.6	69.67	65.38	59.76	58.85
TENE	98.24	91.5	75.42	71.27	82.90	81.50	79.76	76.06
TADW	85.00	81.24	89.75	85.24	84.32	83.07	71.9	67.37
MUSAE	92.06	92.21	98.92	**98.08**	<u>97.82</u>	**95.21**	92.47	92.53
FSADA	91.99	92.08	98.72	97.81	97.49	95.14	92.34	92.33
DANE-WLA	<u>96.79</u>	<u>92.91</u>	<u>99.44</u>	92.07	96.97	94.01	<u>92.94</u>	**92.89**
ANE-WLDSG	**98.35**	**94.83**	**99.72**	<u>95.07</u>	**98.46**	<u>94.95</u>	**93.83**	<u>92.73</u>

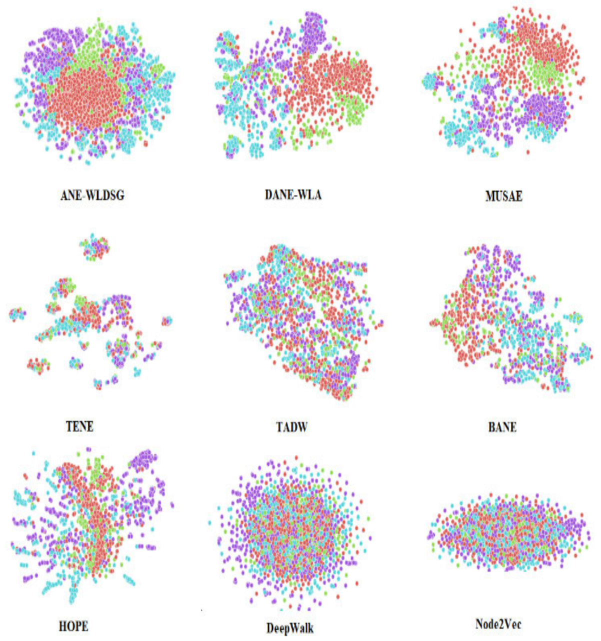
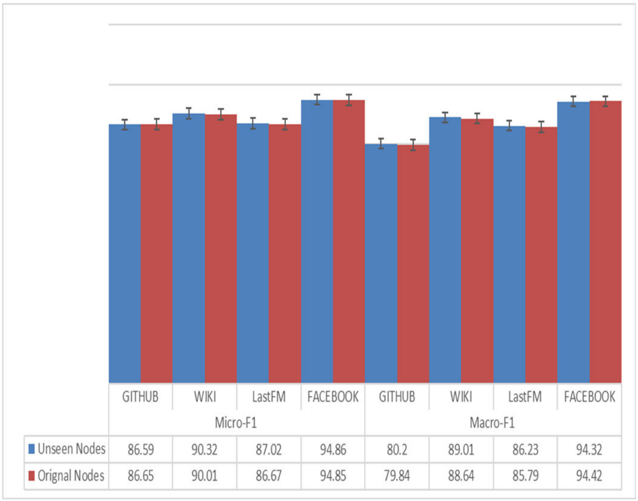


FIGURE 7. T-SNE visualization of different models on Facebook dataset.



	Micro-F1				Macro-F1			
	GITHUB	WIKI	LastFM	FACEBOOK	GITHUB	WIKI	LastFM	FACEBOOK
🟦 Unseen Nodes	86.59	90.32	87.02	94.86	80.2	89.01	86.23	94.32
🟥 Orignal Nodes	86.65	90.01	86.67	94.85	79.84	88.64	85.79	94.42

FIGURE 8. Generalization ability of the proposed model on unseen nodes.

during embedding prosses. To simulate this scenario, 5% of the network nodes were intentionally removed.

After removing the nodes, the modified network structure was used to create a new representation or embedding of the network. This embedding captured the underlying patterns and relationships in the network, incorporating the available data.

The next step involved evaluating the performance of a classifier on the deleted data. The classifier was trained on the embedded modified network nodes. By testing the classifier's ability to correctly classify the deleted data, we could assess how well it could generalize and make accurate predictions on unseen or missing data.

The performance of the classifier on the deleted data was then compared to its performance on the data generated from the embedding of the complete network. This comparison allowed for an understanding of the impact of missing data on the classifier's performance. It provided insights into the effectiveness of the proposed method in handling the invisible or missing data in the network.

FIGURE 8 shows that ANE-WLDSG is able to work with the same efficiency on unseen nodes as it does on original nodes. This is evident from the fact that the Micro-F1 and Macro-F1 scores for unseen nodes are very similar to those for original nodes.

The high Micro-F1 and Macro-F1 scores suggest that the proposed model is able to correctly identify a large number of nodes in each class, regardless of whether they were seen during t0raining or not. This indicates that the model is able to generalize well to new data and is not overfitting to the training data.

The fact that ANE-WLDSG model works with the same efficiency on unseen nodes as it does on original nodes is a promising sign. It suggests that the model is able to learn the underlying patterns in the data and is not simply

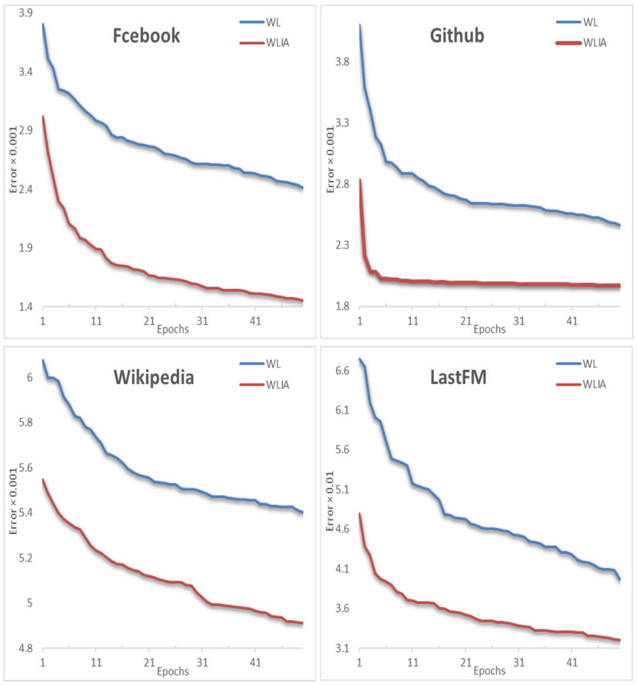


FIGURE 9. Training loss of ANE-WLDSG with WL and ANE-WLDSG with WLIA over 50 Iterations on the four datasets.

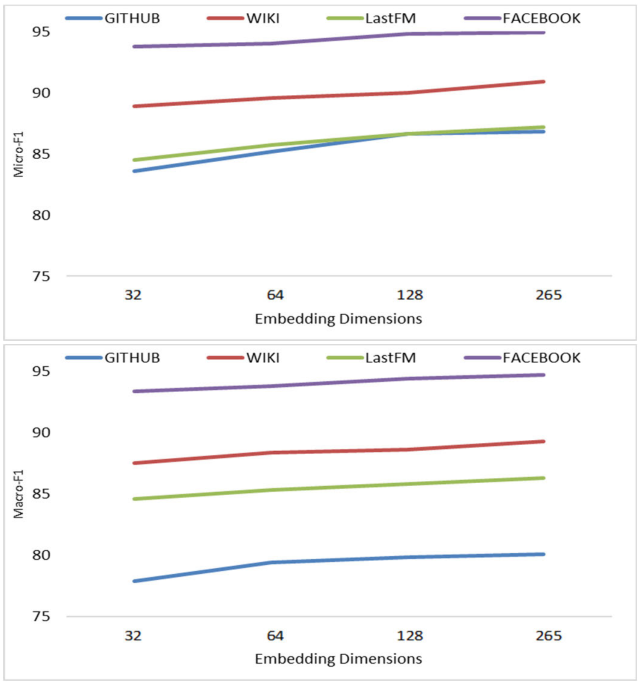


FIGURE 10. Comparisons of WLIA kernal depth size.

memorizing the training data. This makes the model a valuable tool for dealing with unseen data, such as data that is collected in the future or data that come from different domains.

5) THE EFFECT OF WLIA ON THE PERFORMANCE OF ANE-WLDSG

In this section, we explore the impact of WLIA on the performance of the ANE-WLDSG model. We conduct a comparative analysis of the training loss function between ANE-WLDSG with WL and ANE-WLDSG with WLIA across four datasets. Our findings demonstrate that WLIA consistently enhances the model's performance, showcasing its effectiveness as shown in FIGURE 9.

Analyzing the provided loss values for the four datasets, we observe the performance trends of the WL and WLIA models across iterations. Comparing the two models, the WLIA consistently outperforms the WL model at every iteration. The WLIA model starts with a lower initial loss value and achieves lower loss values compared to the WL model at each iteration. As the iterations progress, the performance gap between the two models gradually widens, highlighting the beneficial impact of the iterative adjustment in the WLIA model. Based on these observations, we can conclude that the WLIA model demonstrates superior performance in analyzing the four datasets, as evident by consistently achieving lower loss values at each iteration.

6) PARAMETERS ANALYSIS

To evaluate the impact of the depth parameter K on the performance of our ANE-WLIADSG model, we conducted an analysis with K varying from 1 to 6.

In FIGURE 10, we observe the Micro-F1 and Macro-F1 scores for the WLIA kernel size and their relationship with the performance of the model. Notably, as the kernel size increases, both the Micro-F1 and Macro-F1 scores show improvement. This indicates that the model becomes more proficient in accurately predicting the correct class for each individual instance and overall dataset as the kernel size increases.

The results reveal that the best Micro-F1 and Macro-F1 scores are achieved when using a kernel size of 4. This finding suggests that a kernel size of 4 is the optimal choice for the WLIA model when applied to these datasets. Choosing a larger or smaller kernel size may result in diminished performance compared to the peak performance attained with a kernel size of 4. These findings highlight the importance of selecting an appropriate kernel size, in this case, 4, for the WLIA model to achieve optimal performance on the evaluated datasets.

Second, we examined the Micro-F1 and Macro-F1 scores of ANE-WLIADSG using different embedding dimensions ranging from 32 to 256, as depicted in FIGURE 11.

In general, both the Micro-F1 and Macro-F1 scores exhibit an upward trend as the embedding dimension increases. This indicates that employing a larger embedding dimension has the potential to enhance the model's performance across all four datasets.

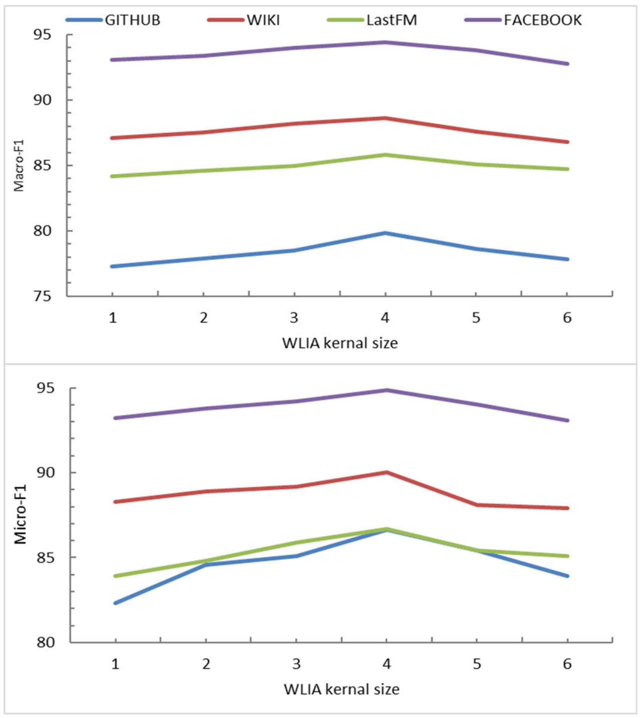


FIGURE 11. **The effect of embedding dimension on node classification performance on different datasets.**

However, the performance improvement is not consistently linear. The disparity between the scores achieved with 128 and 256 embedding dimensions is relatively negligible. This suggests that there might be diminishing returns when augmenting the embedding dimension beyond 128.

V. DISCUSSION

The proposed ANE-WLDSG model for network embedding presents several advancements in capturing network structure and relationships. The linear complexity of the model enables its scalability and efficient handling of large networks. The incorporation of enhancements to the Weisfeiler-Lehman schema and the ReAE normalization function contributes to a balanced and fair estimation of proximity levels between network nodes. Additionally, the integration of the Deep Skip-Gram (DSG) sub-model, which utilizes random walks and leverages deep autoencoders and the Skip-Gram model, enhances the model's performance in capturing non-linear relationships among nodes.

Moreover, the embedded representations obtained from the ANE-WLDSG model hold the potential for broader reuse in various downstream applications. These representations capture valuable structural information and relationships within networks, which can be leveraged in tasks such as link prediction, node classification, and community detection. By providing a flexible and adaptable framework, the ANE-WLDSG model offers opportunities for researchers and practitioners to explore and apply the learned representations in different domains and application scenarios. This opens avenues for cross-domain knowledge transfer and transfer learning, where the insights gained from one domain can be utilized to enhance performance in related domains.

Furthermore, an important aspect to highlight is the ANE-WLDSG model demonstrates the ability to handle invisible data by effectively embedding unseen network nodes, making it suitable for dynamic network analysis. It also exhibits high efficiency and comparable error rates in dealing with both training and test data. This ensures consistent performance and reliable results, particularly in real-world scenarios where networks evolve or labeled data may be limited. Overall, these attributes make the model a valuable tool for various network analysis tasks.

However, while the ANE-WLDSG model shows promise, it also has certain limitations that should be considered. It is designed for homogeneous networks and may not be directly applicable to heterogeneous networks. Additionally, while the model exhibits linear complexity, practical challenges may arise when dealing with extremely large networks. The generalizability of the model to completely new networks and the interpretability of its embedded representations are also areas of concern. Furthermore, the model's application scope may vary across different domains, necessitating further investigation and adaptation.

In short, the ANE-WLDSG model presents advancements in network embedding by capturing proximity, non-linearity, and evolving network structure. However, its limitations in handling heterogeneous networks, scalability for extremely large networks, generalization to new networks, interpretability of embedded representations, and domain-specific applicability should be considered for future research and development. Addressing these limitations will contribute to the broader effectiveness and utility of the ANE-WLDSG model in various network analysis tasks.

VI. CONCLUSION

In this paper, we introduce the ANE-WLDSG model, a unified embedding model that effectively captures proximity, non-linearity, and the dynamic evolving structure of network data. The model exhibits linear complexity, enabling scalability and efficient handling of large networks. Through enhancements to the Weisfeiler-Lehman schema and the introduction of the ReAE normalization function, the model achieves a balanced and fair estimation of proximity levels between network nodes. The incorporation of the Deep Skip-Gram (DSG) sub-model, which utilizes random walks and leverages deep autoencoders, and the Skip-Gram model, further enhances the performance of network embedding by capturing the non-linear relationships among nodes. Notably, the proposed model accommodates the embedding of network nodes that were not present during training, and it dynamically updates node representations as links are added or removed. Comparative evaluations demonstrate that the ANE-WLDSG model outperforms existing embedding models in node classification, link prediction, and visualization tasks. While currently designed for homogeneous networks, future extensions of the model aim to address heterogeneous networks. Furthermore, the model holds promise for applications in diverse fields, including natural language processing and bioinformatics.

APPENDIX

Proof1: We will use the proof by mathematical induction.

Base Case:

When K = 1, this satisfies equation (9), which is close to the first-order proximity, (26), as shown at the bottom of the page.

When K=2, (27) and (28), as shown at the bottom of the page.

Induction Step:

Assume that $P^{(K)}(v) = P^{(K-1)}(v) + \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(K-1)}(u)$ is true, so (29), as shown at the bottom of the page For k+1

$$P^{(K+1)}(v) = P^{(K)}(v) + \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(K)}(u) \quad (30)$$

We aim to prove that (31), as shown at the bottom of the page

$$1 - order\ proximitye\left(v_i, v_j\right) = \begin{cases} \dfrac{1}{\sqrt{d\left(v_i\right)\times d\left(v_j\right)}} & if\ i \neq j\ and\ e\left(v_i, v_j\right) \in E \\ 0 & otherwise \end{cases} \quad (26)$$

$$\begin{aligned} P^{(2)}(v) &= P^{(1)}(v) + \sum_{u\in\mathcal{N}(v_i)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(1)}(u) \\ &= P^{(0)}(v) + \sum_{u\in\mathcal{N}(v_i)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(0)}(u) \\ &\quad + \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d(v)\times d(u)}} \left(P^{(0)}(u) + \sum_{s\in\mathcal{N}(u)} \frac{1}{\sqrt{d(u)\times d(s)}} P^{(0)}(s)\right) \\ &= P^{(0)}(v) + \sum_{u\in\mathcal{N}(v_i)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(0)}(u) \\ &\quad + \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d(v)\times d(u)}} P^{(0)}(u) + \sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d(v)\times d(u)}} \sum_{s\in\mathcal{N}(v)} \frac{1}{\sqrt{d(u)\times d(s)}} P^{(0)}(s) \end{aligned}$$

$$P^{(2)}(v) = P^{(0)}(v) + \sum\nolimits_{u\in\mathcal{N}(v)} \frac{2}{\sqrt{d(v)\times d(u)}} P^{(0)}(u) + \sum\nolimits_{s\in 2^{nd}-\text{depth }(v)} \frac{1}{\sqrt{d(v)\times d(s)\times d(u)}} P^{(0)}(s) \quad (27)$$

$$2 - order\ proximity\ e\left(v_i, v_j\right) = \begin{cases} \dfrac{1}{\sqrt{d\left(v_j\right)\times d\left(v_j\right)\times d\left(v_l\right)}} & when\ e\left(v_i, v_l\right) \in E\ and\ e\left(v_l, v_j\right) \in E \\ 0 & otherwise \end{cases} \quad (28)$$

$$\begin{aligned} P^{(K)}(v) &= P^{(0)}(v) + \sum_{u_1\in 1^{th-depth}(v)} \frac{\binom{K}{1}}{\sqrt{d(v)\times d(u_k)}} P^{(0)}(u_1) \\ &\quad + \ldots + \sum_{u_K\in K^{th-path}(v)} \frac{\binom{K}{K}}{\sqrt{d(v)\times d(u_k)}\times \prod_{u_l\in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \end{aligned} \quad (29)$$

$$\begin{aligned} P^{(K+1)}(v) &= P^{(0)}(v) + \sum_{u_1\in 1^{th-depth}(v)} \frac{\binom{K+1}{1}}{\sqrt{d(v)d(u_k)}} P^{(0)}(u_1) \\ &\quad + \cdots + \sum_{u_{K+1}\in K+1^{\text{th}}-depth(v)} \frac{\binom{K+1}{K+1}}{\sqrt{d(v)\times d(u_{k+1})}\times \prod_{u_l\in K+1-path(v,u_k)} d(u_l)} P^{(0)}(u_{k+1}) \end{aligned} \quad (31)$$

$$
\begin{aligned}
P^{(K+1)}(v) &= P^{(0)}(v) + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{1}}{\sqrt{d(v) \times d(u_k)}} P^{(0)}(u_1) \\
&\quad + \ldots + \sum_{u_K \in K^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{\mathrm{K}}}{\sqrt{d(v) \times d(u_k)} \times \prod_{u_l \in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \\
&\quad + \sum_{u_1 \in \mathcal{N}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} \Bigg[P^{(0)}(u_1) + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{1}}{\sqrt{d(u_1) \times d\left(u_{1_k}\right)}} P^{(0)}\left(u_{1_1}\right) + \ldots \\
&\quad + \sum_{u_{1_K} \in K-1^{\text{th-depth}}(u_1)} \frac{\binom{\mathrm{K}}{\mathrm{K}-1}}{\sqrt{d(u_1) \times d\left(u_{1_{k-1}}\right)} \times \prod_{u_{1_l} \in K-1-path(u_1,u_{k-1})} d(u_{1_l})} P^{(0)}\left(u_{1_{K-1}}\right) \\
&\quad + \sum_{u_{1_K} \in K^{\text{th-depth}}(u_1)} \frac{\binom{\mathrm{K}}{\mathrm{K}}}{\sqrt{d(u_1) \times d\left(u_{1_k}\right)} \times \prod_{u_{1_l} \in K-path(u_1,u_k)} d(u_{1_l})} P^{(0)}\left(u_{1_K}\right) \Bigg] \\
P^{(K+1)}(v) &= P^{(0)}(v) + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{1}}{\sqrt{d(v) \times d(u_k)}} P^{(0)}(u_1) \\
&\quad + \ldots + \sum_{u_K \in K^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{\mathrm{K}}}{\sqrt{d(v) \times d(u_k)} \times \prod_{u_l \in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \\
&\quad + \Bigg[\sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} P^{(0)}(u_1) + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{\binom{\mathrm{K}}{1}}{\sqrt{d(u_1) \times d\left(u_{1_k}\right)}} P^{(0)}\left(u_{1_1}\right) + \ldots \\
&\quad + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} \sum_{u_{1_{K-1}} \in K-1^{\text{th-depth}}(u_1)} \left(\frac{\binom{\mathrm{K}}{\mathrm{K}}}{\sqrt{d(u_1) \times d\left(u_{1_{k-1}}\right)} \times \prod_{u_{1_l} \in K-path(u_1,u_{k-1})} d(u_{1_l})} \right) P^{(0)}\left(u_{1_{K-1}}\right) \\
&\quad + \sum_{u_1 \in 1^{\text{th-depth}}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} \sum_{u_{1_K} \in K^{\text{th-depth}}(u_1)} \frac{\binom{\mathrm{K}}{\mathrm{K}}}{\sqrt{d(u_1) \times d\left(u_{1_k}\right)} \times \prod_{u_{1_l} \in K-path(u_1,u_k)} d(u_{1_l})} P^{(0)}\left(u_{1_K}\right) \Bigg] \\
P^{(K+1)}(v) &= P^{(0)}(v) + \sum\nolimits_{u_1 \in 1^{th-depth}(v)} \frac{\binom{K}{1}}{\sqrt{d(v) \times d(u_k)}} P^{(0)}(u_1) \\
&\quad + \ldots + \sum\nolimits_{u_K \in K^{th-depth}(v)} \frac{\binom{K}{K}}{\sqrt{d(v) \times d(u_k)} \times \prod_{u_l \in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \\
&\quad + \Bigg[\sum\nolimits_{u_1 \in 1^{th-depth}(v)} \frac{1}{\sqrt{d(v) \times d(u_1)}} P^{(0)}(u_1) + \sum\nolimits_{u_1 \in 1^{th-depth}(v)} \sum\nolimits_{u_1 \in 1^{th-depth}(v)} \frac{\binom{K}{1}}{\sqrt{d(v) \times d\left(u_{1_k}\right)} \times d(u_1)} P^{(0)}\left(u_{1_1}\right) \\
&\quad + \ldots + \sum\nolimits_{u_1 \in 1^{th-depth}(v)} \sum\nolimits_{u_{1_{K-1}} \in K-1^{th-depth}(u_1)} \frac{\binom{K}{K-1}}{\sqrt{d(v) \times d\left(u_{1_{k-1}}\right)} \times d(u_1) \times \prod_{u_{1_l} \in K-1-padth(u_1,u_{k-1})} d(u_{1_l})} \\
&\quad \times P^{(0)}\left(u_{1_{K-1}}\right) + \sum_{u_1 \in 1^{th-depth}(v)} \sum_{u_{1_K} \in K^{th-depth}(u_1)} \frac{\binom{K}{K}}{\sqrt{d(v) \times d\left(u_{1_k}\right)} \times d(u_1) \times \prod_{u_{1_l} \in K-path(u_1,u_k)} d(u_{1_l})} P^{(0)}\left(u_{1_K}\right) \Bigg]
\end{aligned}
\tag{32}
$$

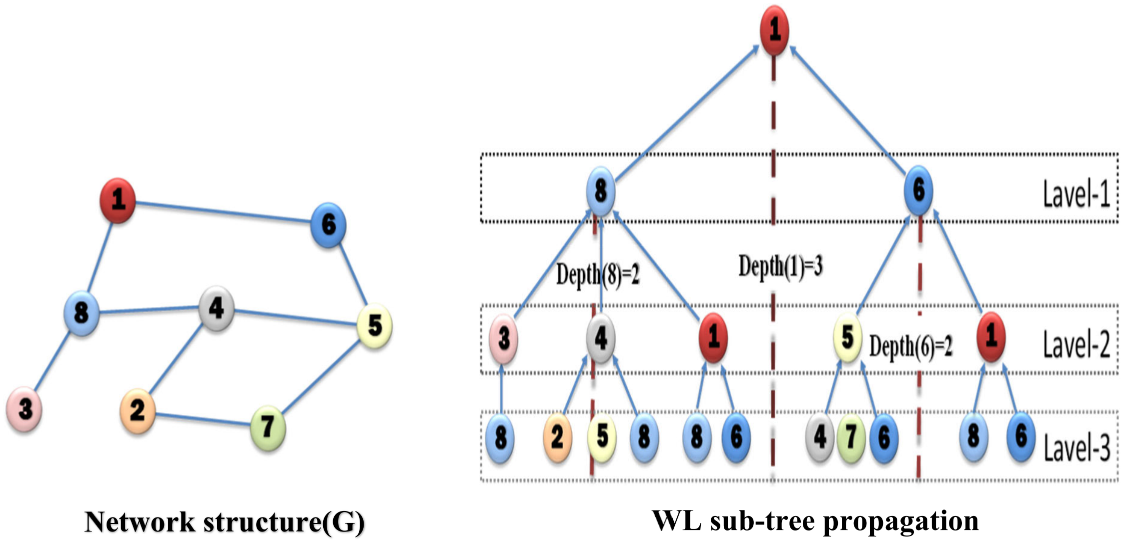


FIGURE 12. Illustrating the WL sub-tree propagation of node 1 according to network G, the nodes on the third level of the root node can be found on the second level of the nodes that are next to it.

Depending on the representation of $P^{(K)}(v) =$ in Equation 29, Equation 30 can be formulated as (32), shown at the previous page.

We can note that:

$$\sum_{u_1\in 1^{\text{th}-\text{depth}}(v)} \sum_{u_{1_k}\in k^{\text{th}-\text{depth}}(u_1)} \text{x} = \sum_{u_k\in k+1^{\text{th}-\text{depth}}(v)} \text{x}, P^{(0)}\left(u_{1_K}\right) = P^{(0)}\left(u_{k+1}\right)$$

It is noticeable that FIGURE 12 provides a visual representation of this observation.

So, equation (31) will be $P^{(K+1)}(v)$, as shown at the bottom of the previous page.

From Pascal's identity $(\binom{K}{k} + \binom{K}{k-1} = \binom{K+1}{k}$, *for* $0 < k \leq K)$ and the fact $\binom{K+1}{K+1} = \binom{K}{K} = \binom{K}{0} = 1$, equation 33 can be written as (34), as shown at the top of the next page.

$$\begin{aligned} P^{(K+1)}(v) = P^{(0)}(v) &+ \sum_{u_1\in 1^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{1}}{\sqrt{d(v)\times d(u_k)}} P^{(0)}(u_1) \\ &+ \ldots + \sum_{u_K\in K^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{K}}{\sqrt{d(v)\times d(u_k)}\times \prod_{u_l\in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \\ &+ \Bigg[\sum_{u_1\in 1^{\text{th}-\text{depth}}(v)} \frac{1}{\sqrt{d(v)\times d(u_1)}} P^{(0)}(u_1) + \sum_{u_1\in 2^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{1}}{\sqrt{d(v)\times d(u_2)}\times d(u_1)} P^{(0)}(u_2) \\ &+ \ldots + \sum_{u_K\in K^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{K-1}}{\sqrt{d(v)\times d(u_k)}\times \prod_{u_l\in K-path(v,u_k)} d(u_l)} P^{(0)}(u_k) \\ &+ \sum_{u_{K+1}\in K+1^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{K}}{\sqrt{d(v)\times d(u_{k+1})}\times \prod_{u_l\in K+1-path(v,u_k)} d(u_l)} P^{(0)}(u_{k+1}) \Bigg] \end{aligned}$$

$$\begin{aligned} P^{(K+1)}(v) = P^{(0)}(v) &+ \sum_{u_1\in 1^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{1}+\binom{K}{0}}{\sqrt{d(v)\times d(u_k)}} P^{(0)}(u_1) \\ &+ \ldots + \sum_{u_K\in K^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{K}+\binom{K}{k-1}}{\sqrt{d(v)\times d(u_k)}\times \prod_{u_l\in K-path(v,u_k)} d(u_l)} P^{(0)}(u_K) \\ &+ \sum_{u_{K+1}\in K+1^{\text{th}-\text{depth}}(v)} \frac{\binom{K}{K}}{\sqrt{d(v)\times d(u_{k+1})}\times \prod_{u_l\in K+1-path(v,u_k)} d(u_l)} P^{(0)}(u_{k+1}) \end{aligned} \quad (33)$$

$$
\begin{aligned}
P^{(K+1)}(v) = P^{(0)}(v) &+ \sum_{u_1 \in 1^{th-depth}(v)} \frac{\binom{K+1}{1}}{\sqrt{d(v) d\left(u_k\right)}} P^{(0)}\left(u_1\right) \\
&+ \cdots + \sum_{u_K \in K^{th-depth}(v)} \frac{\binom{K+1}{K}}{\sqrt{d(v) \times d\left(u_k\right)} \times \prod_{u_l \in K-\text{ path }\left(v, u_k\right)} d\left(u_l\right)} P^{(0)}\left(u_K\right) \\
&+ \sum_{u_{K+1} \in K+1^{th-depth}(v)} \frac{\binom{K+1}{K+1}}{\sqrt{d(v) \times d\left(u_{k+1}\right)} \times \prod_{u_l \in K+1-\text{ path }\left(v, u_k\right)} d\left(u_l\right)} P^{(0)}\left(u_{k+1}\right)
\end{aligned}
\tag{34}
$$

In accordance with equation 34, the hypothesis in equation 31 is confirmed.

Hence, Theorem1 is true for K+1, and in general true for each K.

REFERENCES

[1] A. M. Fout, *Protein Interface Prediction Using Graph Convolutional Networks*. Fort Collins, CO, USA: Colorado State Univ., 2017.

[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, ''Line: Large-scale information network embedding,'' in *Proc. ACM WWW*, vol. 2015, pp. 1067–1077.

[3] A. Bojchevski and S. Günnemann, ''Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking,'' 2017, *arXiv:1707.03815*.

[4] L. Liao, X. He, H. Zhang, and T.-S. Chua, ''Attributed social network embedding,'' *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2257–2270, Dec. 2018.

[5] W. Wang, H. Yin, X. Du, W. Hua, Y. Li, and Q. V. H. Nguyen, ''Online user representation learning across heterogeneous social networks,'' in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 545–554.

[6] U. Can and B. Alatas, ''A new direction in social network analysis: Online social network analysis problems and applications,'' *Phys. A, Stat. Mech. Appl.*, vol. 535, Dec. 2019, Art. no. 122372.

[7] J. Li and G. Yang, ''Network embedding enhanced intelligent recommendation for online social networks,'' *Future Gener. Comput. Syst.*, vol. 119, pp. 68–76, Jun. 2021.

[8] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, ''Graph learning: A survey,'' *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 109–127, Apr. 2021.

[9] J. Zhou, L. Liu, W. Wei, and J. Fan, ''Network representation learning: From preprocessing, feature extraction to node embedding,'' *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–35, Feb. 2023.

[10] C. Zhuang and Q. Ma, ''Dual graph convolutional networks for graph-based semi-supervised classification,'' in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 499–508.

[11] S. Fu, W. Liu, K. Zhang, Y. Zhou, and D. Tao, ''Semi-supervised classification by graph p-Laplacian convolutional networks,'' *Inf. Sci.*, vol. 560, pp. 92–106, Jun. 2021.

[12] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, ''Link prediction techniques, applications, and performance: A survey,'' *Phys. A, Stat. Mech. Appl.*, vol. 553, Sep. 2020, Art. no. 124289.

[13] L. Cai, J. Li, J. Wang, and S. Ji, ''Line graph neural networks for link prediction,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, Sep. 2022.

[14] H. Sun, F. He, J. Huang, Y. Sun, Y. Li, C. Wang, L. He, Z. Sun, and X. Jia, ''Network embedding for community detection in attributed networks,'' *ACM Trans. Knowl. Discovery From Data*, vol. 14, no. 3, pp. 1–25, Jun. 2020.

[15] X. Su et al., ''A comprehensive survey on community detection with deep learning,'' *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.

[16] H. Yang, L. Chen, S. Pan, H. Wang, and P. Zhang, ''Discrete embedding for attributed graphs,'' *Pattern Recognit.*, vol. 123, Mar. 2022, Art. no. 108368.

[17] Y. Xie et al., ''SketchNE: Embedding billion-scale networks accurately in one hour,'' *IEEE Trans. Knowl. Data Eng.*, 2023.

[18] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong, ''Dynamic network embedding survey,'' *Neurocomputing*, vol. 472, pp. 212–223, Feb. 2022.

[19] M. COSKUN, ''A high order proximity measure for linear network embedding,'' *Nigde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi*, vol. 11, no. 3, pp. 477–483, 2022.

[20] S. Schneider, J. H. Lee, and M. W. J. N. Mathis, ''Learnable latent embeddings for joint behavioural and neural analysis,'' *Nature*, vol. 11, no. 3, pp. 1–9, May 2023.

[21] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, ''Distributed large-scale natural graph factorization,'' in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 37–48.

[22] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, ''Asymmetric transitivity preserving graph embedding,'' in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1105–1114.

[23] N. M. Kriege, F. D. Johansson, and C. Morris, ''A survey on graph kernels,'' *Appl. Netw. Sci.*, vol. 5, no. 1, pp. 1–42, Dec. 2020.

[24] H. Chen and H. Koga, ''Gl2vec: Graph embedding enriched by line graphs with edge features,'' in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2019, pp. 3–14.

[25] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, ''Weisfeiler–Lehman graph kernels,'' *J. Mach. Learn. Res.*, vol. 12, no. 9, pp. 2539–2561, 2011.

[26] B. Perozzi, R. Al-Rfou, and S. Skiena, ''DeepWalk: Online learning of social representations,'' in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[27] A. Grover and J. Leskovec, ''node2vec: Scalable feature learning for networks,'' in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, vol. 2016, pp. 855–864.

[28] B. Perozzi, V. Kulkarni, H. Chen, S. Skiena, and S. Don't Walk, ''Online learning of multi-scale network embeddings,'' in *Proc. 2017 IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, vol. 2017, pp. 258–265.

[29] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, ''Distributed representations of words and phrases and their compositionality,'' 2013, *arXiv:1310.4546*.

[30] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, ''Network representation learning with rich text information,'' in *Proc. IJCAI*, 2015, pp. 2111–2117.

[31] B. Rozemberczki, C. Allen, R. Sarkar, and x. Thilo Gross, ''Multi-scale attributed node embedding,'' *J. Complex Netw.*, vol. 9, no. 1, pp. 1–22, Apr. 2021.

[32] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, ''A comprehensive survey on graph neural networks,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[33] M. Gori, G. Monfardini, and F. Scarselli, ''A new model for learning in graph domains,'' in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2005, pp. 729–734.

[34] M. Henaff, J. Bruna, and Y. LeCun, ''Deep convolutional networks on graph-structured data,'' 2015, *arXiv:1506.05163*.

[35] T. N. Kipf and M. Welling, ''Semi-supervised classification with graph convolutional networks,'' 2016, *arXiv:1609.02907*.

[36] F. Manessi, A. Rozza, and M. Manzo, ''Dynamic graph convolutional networks,'' *Pattern Recognit.*, vol. 97, Jan. 2020, Art. no. 107000.

[37] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[38] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Proc. NeurIPS*, 2018, pp. 9197–9207.
[39] H. Gao and H. Huang, "Deep attributed network embedding," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 3364–3370.
[40] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.
[41] M. Hou, L. Wang, J. Liu, X. Kong, and F. Xia, "A3Graph: Adversarial attributed autoencoder for graph representation learning," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, 2021, pp. 1697–1704.
[42] A. T. Al-Furas, M. F. Alrahmawy, W. M. Al-Adrousy, and S. Elmougy, "Deep attributed network embedding via Weisfeiler–Lehman and autoencoder," *IEEE Access*, vol. 10, pp. 61342–61353, 2022.
[43] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Ser.*, vol. 2, no. 9, pp. 12–16, 1968.
[44] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis, "Graph kernels: A survey," 2019, *arXiv:1904.12218*.
[45] H. Yang, L. Chen, M. Lei, L. Niu, C. Zhou, and P. Zhang, "Discrete embedding for latent networks," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1223–1229.
[46] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, "RiWalk: Fast structural node embedding via role identification," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 478–487.
[47] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 2, pp. 95–107, Apr. 2020.
[48] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1320–1329.
[49] J. Guo, L. Xu, and J. Liu, "SPINE: Structural identity preserved inductive network embedding," 2018, *arXiv:1802.03984*.
[50] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 385–394.
[51] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proc. IJCAI*, 2016, pp. 3889–3895.
[52] D. Jin, M. Ge, L. Yang, D. He, L. Wang, and W. Zhang, "Integrative network embedding via deep joint reconstruction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3407–3413.
[53] D. Jin, B. Li, P. Jiao, D. He, and W. Zhang, "Network-specific variational auto-encoder for embedding in attribute networks," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2663–2669.
[54] R. Hong, Y. He, L. Wu, Y. Ge, and X. Wu, "Deep attributed network embedding by preserving structure and attribute information," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 51, no. 3, pp. 1434–1445, Mar. 2021.
[55] W. Yu, W. Cheng, C. Aggarwal, B. Zong, H. Chen, and W. Wang, "Self-attentive attributed network embedding through adversarial learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2019, pp. 758–767.
[56] K. Wang, L. Xu, L. Huang, C.-D. Wang, Y. Tang, and C. Fu, "Inter-intra information preserving attributed network embedding," *IEEE Access*, vol. 7, pp. 79463–79476, 2019.
[57] D. Chen, M. Nie, H. Zhang, Z. Wang, and D. Wang, "Network embedding algorithm taking in variational graph AutoEncoder," *Mathematics*, vol. 10, no. 3, p. 485, Feb. 2022.
[58] Y. Pan, J. Zou, J. Qiu, S. Wang, G. Hu, and Z. Pan, "Joint network embedding of network structure and node attributes via deep autoencoder," *Neurocomputing*, vol. 468, pp. 198–210, Jan. 2022.
[59] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 3–28, Mar. 2020.
[60] S. Akbari, E. Ghorbani, J. H. Koolen, and M. R. Oboudi, "On sum of powers of the Laplacian and signless Laplacian eigenvalues of graphs," *Electron. J. Combinatorics*, vol. 17, no. 1, pp. 171–182, Aug. 2010.
[61] B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate club: An API oriented open-source Python framework for unsupervised learning on graphs," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 3125–3132.
[62] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, "Binarized attributed network embedding," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 1476–1481.
[63] S. Yang and B. Yang, "Enhanced network embedding with text information," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 326–331.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[65] A. A. Al-Furas, M. F. Alrahmawy, W. M. A. Al-Adrousy, and S. Elmougy, "Improving link prediction in network representation learning with feature fusion and local outlier factor," *Fusion, Pract. Appl.*, vol. 12, no. 2, pp. 120–131, 2023, doi: 10.54216/FPA.120210.
[66] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, Jun. 2000.

AMR AL-FURAS received the B.S. degree in mathematics and computer Sciences from Ibb University, Ibb, Yemen, in 2005, and the M.S. degree in computer sciences from the Faculty of Computers and Information, Mansoura University, Mansoura, Egypt, in 2017, where he is currently pursuing the Ph.D. degree in computer sciences. He was a Demonstrator with Ibb University, from 2009 to 2013. His current research interests are AI, data analytics, social network analysis, and recommender systems.

MOHAMMED F. ALRAHMAWY received the B.Eng. degree in electronics engineering from the University of Mansoura, Egypt, in 1997, the M.Sc. degree in automatic control engineering from Mansoura University, in 2001, and the Ph.D. degree in computer science from the University of York, U.K., in 2011. In 2005, he joined the Real-Time Systems Research Group, Computer Science Department, University of York, as a Ph.D. Research Student. In 2011, he joined as a Lecturer with the Department of Computer Science, Mansoura University, and became a Professor of computer science, in January 2023. Since January 2022, he has been the Acting-Head of the Computer Scientists Department, Mansoura University. His Ph.D. study was fully funded by the Egyptian Ministry of Higher Education. His current research interests include deep learning, network and graph analytics, real-time systems and languages, NLP, cloud computing, distributed and parallel computing, image processing, computer vision, the IoT, and big data. He received the Best M.Sc. Thesis Award from Mansoura University, in 2003.

ABDULAZIZ ALBLWI received the M.Sc. and Ph.D. degrees in information technology from Bournemouth University, in 2017 and 2020, respectively. In 2012, he joined as a Lecturer Assistant with the Computer Science Department, Faculty of Applied College, Taibah University, Saudi Arabia, where he is currently an Assistant Professor of information technology. His research has an interdisciplinary nature, with a focus on human–computer interaction, gamification, and data analytics.

WALEED MOHAMED AL-ADROUSY was born in Mecca, Saudi Arabia, in 1982. He received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the Faculty of Computer and Information Sciences, Mansoura University, in 2004, 2010, and 2015, respectively. He is an Egyptian researcher. He was a Demonstrator (2004–2010), a Teaching Assistant (2010–2015), and a Lecturer (since 2015) with the Faculty of Computer and Information Sciences, Mansoura University. He is interested in several scientific fields, such as social network analysis, recommender systems, distributed systems, game development, 3-D modeling, simulation, mobile development, security, and software engineering.

SAMIR ELMOUGY received the Ph.D. degree in computer science from the School of Electrical Engineering and Computer Science, Oregon State University, USA. He is a Professor of computer science with the Department of Computer Science, Faculty of Computers and Information, Mansoura University, Egypt. From 2008 to 2014, he was an Assistant Professor with the Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He has published over 70 articles in refereed IEEE Transactions/Springer journals, IEEE conferences, and book chapters. He has participated in the reviewing process of many international refereed journals and conferences. His current research interests include artificial intelligence, the IoT, information theory, and software engineering.