Deep Attributed Network Embedding via Weisfeiler-Lehman and Autoencoder

Amr Al-furas 1,2, Mohammed F. Alrahmawy 1, Waleed M. Al-Adrousy 1, and Samir Elmougy 1.

1 Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt, 35516
2 Computer Science Department, Ibb University, Ibb, Yemen

Corresponding author: Amr Al-furas (e-mail: amrmos783@gmail.com).

ABSTRACT Network embedding plays a critical role in many applications. Node classification, link prediction, and network visualization are examples of such applications. Attributed network embedding aims to learn the low-dimensional representation of network nodes by integrating network architecture and attribute information. The network architectures of many real-world applications are complex, and the relations between network architectures and their attributed nodes are opaque. Thus, shallow models fail to capture deep nonlinear information when an attributed network is embedded, leading to unreliable embedding. In the present paper, a Deep Attributed Network Embedding via Weisfeiler-Lehman and Autoencoder (DANE-WLA) is proposed in order to capture high nonlinearity and preserve the many proximities in the network attribute information of nodes and structures. Weisfeiler-Lehman proximity schema was used to capture the node dependency between both node edges and node attributes based on information sequences. Then, a deep autoencoder was applied to invest complex nonlinear information. Extensive experiments were conducted on benchmark datasets to verify that DANE-WLA is computationally efficient for various tasks requiring network embedding. The experimental results show that our model outperforms the state-of-the-art network embedding models.

INDEX TERMS Attributed Network Embedding; Autoencoder; link prediction; Node classification; visualization; Weisfeiler-Lehman proximity schema.

1. INTRODUCTION

Complex networks are used to describe objects (nodes) and dependencies (edges) [1], which makes them useful as a delimiter for many computing systems in the real world [2-4], for example, social networks [5, 6], citation networks [7, 8] and networks of protein-protein interactions [9]. Such network complexity is a major challenge to numerous machine-learning tasks. Node classification [8, 10], link prediction [11, 12], and community detection [13, 14] are examples of these tasks. To handle these problems, some network embedding methods [15-18] have been presented to encode the architecture and represent them as a vector in a low-dimensional space.

Traditional methods of network embedding, such as Laplacian (LE)[19] and IsoMAP[20] have been adopted to represent the network as a graph structure. These methods attempt to capture local proximity among nodes with common links using graph theory. Because of bifurcation and the expansion of real-world networks, many potential links between nodes can be lost and ignored. Consequently, building an embedding model based on local proximity between nodes is unsatisfactory because it neglects the general network structure.

In recent years, many researchers have shown the importance of considering the general structure of the network as well as the first-order local proximity [8, 15, 21, 22] to develop a network-embedding model. For example, Deepwalk[15] and node2vec[21] seek to hold a high degree of proximity between nodes through a random traffic node sequence in a graph. SDNE[16] and Line[8] considered first-order and second-order proximity between nodes in the network. This type of network-embedding process is called plain network embedding, whereby the links of the node independently rely upon using no ancillary data of the node attributes. Hence, it is impossible to capture the data of each node or to take advantage of both network structure and node attributes.

Attributed networks are widely used to represent networks in which links of nodes and attributes are important in the analysis process [23-25]. For example, in social networks, users associate with their friends and share their profiles as attributes. The embedding process becomes more challenging in attributed networks because it considers network structure and attributed information in the embedding process.

To overcome this challenge, several researchers have proposed network embedding models that learn jointly from node links and attributed information based on previous embedding models [2, 26-28]. For example, Yang et.al.[2] used textual features to observe random walks on networks and proposed the Text-associated DeepWalk (TADW) model. In Huang et.al.[28], links of the node and attributes were used interchangeably as labels to observe learning for each other.

All the embedding methods mentioned above are based on models of shallow network embedding. They involve the attributed data in the modeling process to reduce the data scattering problem. However, attributed network patterns are considered nonlinearly [29], which makes it difficult to capture complex latent information using shallow embedding models. So many studies have suggested deep learning models to capture this latent information[23, 29-31].

On the other hand, Weisfeiler-Lehman Graph Kernels [32] is a well-relied methodology to combine scalability with the capacity to accord with node attributes. For example H. Yang et.al.[33, 34] defined a Weisfeiler-Lehman matrix to represent the interrelationship between a node's attributes and its structure. Xuewei Ma et.al.[35] relied on the ability of the
We summarize the contributions of this work as follows:

• A deep model (DANE-WLA) was proposed for network embedding. The model consisted of two stages. In the first stage, a new Weisfeiler-Lehman schema was used to capture the integration of the links of the node and the attributes. In the second stage, a deep autoencoder was applied to learn the proximity matrix in a nonlinear complex pattern.

• The applicability of the proposed model to real-world networking tasks was evaluated by conducting experiments on four datasets used in common tasks, including node classification, link prediction, and visualization. The results showed that DANE-WLA significantly and consistently outperformed many other recent network embedding methods, even with variations in the various experimental variables.

This paper is organized as follows: The related work is explored in Section 2, and the technical details of DANE-WLA are exhibited in Section 3. Section 4 presents the experiments and results. Finally, in section 5, our conclusion is presented.

II. RELATED WORK

Below is a summary of the related work, which is categorized into three sections: 1) models of plain network embedding, 2) models of Attributed Network Embedding (ANE), and 3) models of deep neural networks (DNN). Table 1 shows a summary of these methods.

A. PLAIN NETWORK EMBEDDING

The earlier methods of network embedding, such as Laplacian Eigenmaps [19] and locality preserving projection [36] are related to the methods of graph embedding. These methods aim to preserve the local data structure and represent it with a lower dimension. Such methods may not be viable for large-scale network embedding, as all of them have a time complexity of $O(n^3)$, where $n$ is the number of nodes. Recently, interest has been focused on developing learning graph embedding with a method-inspired skip-gram model [41]. DeepWalk [15] depends on this task in the case of Ding. This method is based on memorizing short random walks of simultaneous frequencies of pairs of nodes. Node2Vec [21] is a generalized version of a deep walk. In this version, the depth and breadth of the random walks are tuned. The authors in [8] proposed an embedding model to benefit the first and second order of proximity, while node representations were learned. Another method was proposed in HOPE [37] to capture asymmetric transitivity and higher-order proximity in directed graphs. However, these previous methods utilized the network structure and ignored the helpful attributes of nodes.

B. ATTRIBUTED NETWORK EMBEDDING (ANE)

Recently, the ANE has attracted attention. The TADW model [2] is a typical model of ANE, which amalgamates DeepWalk and related text features into the matrix factorization framework. In another study, Hong Yang et al. suggested a binaryized ANE (BANE) [24] model. They defined the Weisfeiler-Lehman proximity matrix for data dependency to be captured between node links and attributes. Subsequently, they used the cyclic coordinate descent (CCD) task to learn the binary node representation [42]. The above works depend on shallow models.

C. DEEP NEURAL NETWORKS

DNNs are developed to learn complex, non-linear object characteristics automatically, and they have been successfully used in many fields, such as image classification [43] and natural language processing [44]. Of all models of deep learning, only the autoencoder and its variants present unsupervised methods for feature engineering and are embraced in several tasks of network analysis. Zhai and Zhang [45] proposed an approach based on an autoencoder to learn representations of nodes in sparse networks. DANE [23] was proposed with a separate double autoencoder that reconstructs the network structure and node attributes.
Richang et al. proposed DANE [29], which consists of three steps that deal with the data discrepancy, integration of graph topology and features, and nonlinearity patterns of ANE in an integrated framework. These frameworks may not be applicable to large-scale network embedding, as they have a time complexity of $O(n^3)$, where $n$ is the number of nodes. Cong Li et al. suggested that (DANRL-AEN) [39] method espouses the concept of the autoencoder. To catch the different order proximity, they developed the decoding part to three offshoots. DANRL-AEN is $O(n^2)$, where $n$ is the number of nodes.

### III. THE PROPOSED METHOD

#### A. PROBLEM DEFINITION
The attributed network $G(V, E, X)$ is a network with $|V|$ nodes and $|E|$ edges, and each node has $F$ features. In this network, $X = \{x_{i,j}\}_{i,j=1}^{\|V\|} \in \mathbb{R}^{|V| \times F}$ is the attributed network. ANE aims to build trained models to represent the nodes in a low-dimensional space on the basis of the matrix of links between node $E$ and attribute matrix $X$. Therefore, the learned representation is able to maintain the closeness existing in the network structure and node attributes.

**Definition 1:** Given an attributed network $G(V, E, X)$, the task of ANE, $f : G \rightarrow \mathbb{R}^{|V| \times d}$, $f(G) = H$ is a function that produces a vectorial low-dimensional representation of each node $v_i \in V$, where the structural proximity in $E$ and attribute proximity in $X$ are preserved, and the $i$-th row in $H$ is called the node embedding of $v_i$.

#### B. PROPOSED DANE-WLA FRAMEWORK
The architecture of the DANE-WLA consists of two main stages as shown in FIGURE 1.

- **Stage 1 (Information aggregation):** In this stage, The Weisfeiler-Lehman proximity schema is defined based on Weisfeiler-Lehman graph kernels to generate a new proximity matrix $P$. This scheme integrates the node attributes in $X$ and links in the network structure.

- **Stage 2 (Deep Embedding):** Because the hidden information in attributed networks is mostly complicated and not linear, a deep autoencoder model is used to embed the proximity matrix $P$ produced from the first stage. The above two stages are summarized in Algorithm 1, and their details are presented next.

### Algorithm 1: Framework of DANE-WLA

| **Input** | Attributed Network $G(V, E, X)$, parameters: number of WL iterations $K$, embedding dimension $d$. |
| | **output**: node vector representations $H \in \mathbb{R}^{|V| \times d}$, encoder parameters $\{W^{(1)}, b, \theta\}$ |

```plaintext
for $k = 1; K$
for $v \in V$
$X^{k+1}(v) = x^k(v)$
for $u \in N(v)$
$X^{k+1}(v) = x^k(v) + \frac{1}{d(v) + d(u)} x^k(u)$
end for
end for
end for
```

while *not getting convergence* do
Sampling a mini-batch of aggregated features $P$
Compute the gradient of $L$ based on Equation (9)
Update autoencoder module parameters $\theta$
end while

**End while**

Obtain representations $H = \theta(P)$ based on Equation (6)

1) **INFORMATION AGGREGATION**

Information aggregation from neighboring network nodes to a goal node is constructed from the graph kernels of Weisfeiler-Lehman [32]. It uses a parameter $k$ to control the neighboring node layers, which join the process of aggregating information.

The Weisfeiler-Lehman scheme is used to increase node labels using the labels of their associated nodes. Subsequently, the increased labels are compressed into new labels.

**Definition 2** (Weisfeiler-Lehman scheme): Let $G = (V, E, l)$ be a graph with initial node labels $l^0(v) = l(v)$ for each $v \in V$, and let $K$ be the number of WL iterations, for all $0 \leq k < K$ recursively computes the new label of $v$ in iteration $k + 1$ by

$$l^{k+1}(v) = f_\#(l^k(v), [l^k(u) : u \in N(v)])$$  \((1)\)
where \( \mathcal{N}(v) \) is the neighbors set of \( v \).

The basic idea of graphs with continuous attributes \( X \) is to construct a spread diagram that captures the features of the current node and creates them by combining them with the features of neighboring nodes according to balanced proportions. Suppose we have a continuous attribute \( x^B(v) = x(v) \in X \) for each node \( v \in \mathcal{V} \). Then, we recursively define

\[
x^{k+1}(v) = x^k(v) + \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d(v) \cdot d(u)}} x^k(u)
\]

where \( d(v) = |\mathcal{N}(v)| \) denotes the degree of node \( v \), and \( \frac{1}{\sqrt{d(v) \cdot d(u)}} \) is an appropriately selected normalizing constant for the edge \( (v, u) \) depending on the features offered by the labeling matrix to preserve the graph structure and its broad applications. This normalizing constant \( \frac{1}{\sqrt{d(v) \cdot d(u)}} \) is taken from the signless Laplacian matrix, which can be explained as follows:

**Definition 3 (Signless Laplacian matrix):** The Laplacian matrix of \( A \), where \( A = [a_{ij}]_{n \times n} \in \mathbb{R}^{n \times n} \) represents the adjacency matrix of an attributed network, and \( G \) is the matrix \( L = D - A \), where \( D = diag(d_1, d_2, \ldots, d_n) \) is a degree matrix of \( G \), with \( d_i = \sum_{j=1}^{n} a_{ij} \) is the degree of node \( i \). The matrix \( S = D + A \) is called the signless Laplacian matrix of \( G \). For undirected graph \( G \), the normalized signless Laplacian matrices are defined as [46]

\[
S^+ = D^{-1/2} S D^{-1/2} = I + D^{-1/2} A
\]

The element of \( S^+ \) are given by:

\[
S^+_{ij} = \begin{cases} 
\frac{1}{\sqrt{d_i d_j}} & \text{if } i = j \\
0 & \text{otherwise} 
\end{cases}
\]

Using the recursive procedure described above, we propose a WL-based feature extraction scheme that generates node features from the node attributes of the graphs.

**Definition 4 (WL features):** Let \( G(A, X) \) be an attributed network, and let \( K \) be the number of WL iterations. Then, for every \( k \in \{0, \ldots, K\} \), we define the WL features as

\[
X^k = \begin{bmatrix} x(v_1)^k, x(v_2)^k, \ldots, x(v_1)^k \end{bmatrix}^T
\]

So, we can define \( P \) as

\[
P = x^k, P(v_i) = x_i^k
\]

### 2. DEEP NETWORK EMBEDDING

In this stage, low-dimensional deep embedding vectors are generated from the proximity matrix \( P \) using deep learning models. We apply the autoencoder to capture the nonlinear complex patterns resulting from the previous stage because it is a dominant unadministered deep model of feature learning [23]. The model first learns the input's deep representations through nonlinear learning from the inputs. Next, the outputs are constructed from deep representations. In the DANE-WLA deep autoencoder process, given the proximity matrix \( P \), we aim to learn a mapping function \( \varphi: P \rightarrow H \in \mathbb{R}^{|V| \cdot d} \) as defined in equation 7. The autoencoder architecture contains an encoder and decoder, which can be defined as:

**Encoder:** This transforms the input vector \( p_i \) to a hidden layer representation \( h_i \in \mathbb{R}^d \), where \( p_i \in \mathbb{R}^m \) is the \( i \text{th} \) row of the improved matrix \( P \). It is defined as:

\[
h_i = \varphi(W^{(1)} p_i + b)
\]

where \( \{W^{(1)}, b\} \) are encoder parameters, \( W^{(1)} \) is a weight matrix while \( b \) is a bias vector, and \( \varphi(.) \) denotes the nonlinear activation function.

**Decoder:** The autoencoder maps \( h_i \) to the reconstruction \( \tilde{p}_i \) of the same shape as \( p_i \). It is formulated as:

\[
\tilde{p}_i = \varphi(W^{(2)} h_i + b')
\]

where \( \{W^{(2)}, b'\} \) are decoder parameters, \( W^{(2)} \) is a weight matrix while \( b' \) is a bias vector, and \( \varphi(.) \) denotes the nonlinear activation function.

A solution of the following optimization problem can help in learning all the parameters:

\[
\min \sum_{i=1}^{n} \mathcal{L}(p_i, \tilde{p}_i; W^{(1)}, b, W^{(2)}, b')
\]

By reconstruction error minimization, parameters of the model can be learned:

\[
\min \sum_{i=1}^{n} \left\| \tilde{p}_i - p_i \right\|_2^2
\]

To capture the high nonlinearity in the improved matrix, the deep autoencoder utilizes \( L \) layers in the encoder as follows:

\[
h_i^{(l)} = \varphi(W^{(l)} h_i^{(l-1)} + b^{(l)})
\]

Reciprocally, there are \( L \) layers in the decoder, where \( h_i^{(L)} \) is the required low-dimensional representation of the \( i \text{th} \) node.

**C. MODEL ANALYSIS AND DISCUSSION**

In this subsection, we focus on and evaluate the competitive characteristics and time complexity of the proposed method by analyzing them and then comparing them with other models.

**1) TIME COMPLEXITY:**

Our proposed DANE-WLA framework consists of two parts: information aggregation and deep network embedding. In the first part, we used a Weisfeiler-Lehman whose time-complexity is \( O(k|E|) \) [32], where \( k \) is a small value. This part has \( O(|E|) \) time-complexity. In the second part, we used an autoencoder whose time-complexity is \( O(|V|) \) where \( |V| \) represents the largest hidden layer dimension. Accordingly, the DANE-WLA time complexity is \( O(|V| + |E|) \) (See TABLE 2).

**2) COMPARISON BASED ON THE DESIDERATA**

A node embedding technique must have specific properties to produce callable node features and scalability [47, 48].

---

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
We summarized the evaluation of our proposed embedding model against previous embedding methods in TABLE 2 where concerns have (√) desired properties.

IV. EXPERIMENTS
We applied the proposed model to real-world networks to verify its efficiency and effectiveness in ANE. Then, we compared its performance with that of many well-known models in this field.

A. BENCHMARK DATASETS
To evaluate the proposed model, we used four social networks in our experiments, which are freely available on SNAP[49]. The descriptions of these datasets are as follows.

- **Facebook**: The nodes of this network represent Facebook pages, whereas likes are expressed by the edges linked to the nodes. Node attributes are taken from page represents, and the category is one of four classes (Politician, Government, Company, and TV Show).
- **Twitch social networks**: Twitch users represent the nodes of this network, while friendships between them are expressed by the edges linked to the nodes. Node attributes are taken depending on the streaming habits, and the binary category variable is described as whether the user produces evident content.
- **Wikipedia graphs**: In this network, Wikipedia pages represent the nodes and edges are the reciprocal links. The node attributed represents the existence of nouns in the essay, and the binary category variable articulates the volume of traffic on the page.
- **GitHub**: The nodes of this network represent the GitHub developers, and the links represent common follower relationships. Attributes are determined based on location, metadata, and biography. The categories are to classify each developer as either a machine learning developer or a web developer.

TABLE 3 summarizes the details of these dataset networks.

| dataset      | No. of Nodes | No. of Edges | No. of Attributes | No. of Target |
|--------------|--------------|--------------|------------------|--------------|
| Facebook     | 22,470       | 171,002      | 4,714            | 4            |
| Twitch       | 7,126        | 35,324       | 3,170            | 2            |
| Wikipedia    | 11,631       | 170,918      | 13,183           | 2            |
| GitHub       | 37,700       | 289,003      | 4,005            | 2            |

B. BASELINES
The proposed model DANE-WLA is compared with seven common models, and the details of these models can be listed as follows:

---

TABLE 2. A summary of existing node embedding techniques concerning having (√) desired properties.

| Model   | Generic features | Nonlinearity | Implicit | Higher-order | Handle missing data | Time Complexity |
|---------|-----------------|--------------|----------|--------------|---------------------|-----------------|
| HOPE    |                 |              |          |              |                     | O(|V| P).       |
| DeepWalk|                 |              |          |              |                     | O(|V| P).       |
| Node2Vec|                 |              |          |              |                     | O(|V| P).       |
| BANE    | √               |              |          |              |                     | O(|V| P).       |
| TENE    | √               |              |          |              |                     | O(|E| ).        |
| TADW    | √               |              |          |              |                     | O(|V| P).       |
| MUSAE   |                 |              |          |              |                     | O(|V|).         |
| DANE-WLA|                 |              |          |              |                     | O(|E| + |V| ).   |

---

Algorithm 2: Node representation using trained DANE-WLA

**Input**: Attributed Network: G(V, E, X), vertex v, number of WL iterations K
**encoder parameters** [W(1), b, ], φ.

**output**: node vector representations h ∈ R1×d

\[ p = \text{WL} \_ \text{aggregation} (v, K) \]

Obtain representations \( h = \phi (p) \) based on Equation (6)

**return** h

\[ \text{WL} \_ \text{aggregation} (v, k) \]

\[ \text{if} \quad k = 0 \quad \text{then} \]
\[ p = \text{X}(v) \]

\[ \text{else} \]
\[ p = \text{WL} \_ \text{aggregation} (v, k - 1) \]

\[ \text{for} \quad u \in N(v) \]
\[ p = p + \frac{1}{\sqrt{d(v) \times d(u)}} * \]
\[ \text{WL} \_ \text{aggregation} (u, k - 1) \]

**end if**

**return** p

- **Generic features**: Prevalent properties of nodes, such as social network user profiles, are encoded when learning to embed a node. Generic characteristics have been utilized for the universal contextualization of node locations in the integration space.
- **Non-linear**: A node-node closeness outcome in the goal model is a non-linear task of the vector of node embedding, which helps to encode superlinear and sublinear proximity outcome encoding.
- **Implicit**: The decomposed target matrix is inexplicitly computed, which saves time and the required complexity of space by applying the integration model in practical large-scale made-up settings.
- **Inductive**: The node representation technique assigns invisible nodes to the representation areas that are unconnected to the training network. In DANE-WLA, we can represent any node by applying Algorithm2.
- **Higher-order**: Representation encodes the information of nodes that is far from a node. For instance, in contextualization, which relies on WL, two nodes are similar in generation k+1 if they are similar in k generation, and all neighbors are similar.
- **Handling missing data**: When missing data are presented in node attributes and ignored, the results of the network analysis will be highly skewed.
• **HOPE[37]:** A plain network embedding method that attempts to capture asymmetric transitivity and higher-order proximity in directed graphs using generalized singular value decomposition instead of an adjacency matrix.

• **DeepWalk[15]:** It learns embedding through the prognosis of the local neighborhood of the nodes, sampled from random walks on the network, and uses skip-gram to analyze the walking tracks.

• **Node2vec[21]:** It is a generalized version of the deep walk, and there are two basic strategies to create a path between two nodes in the graph "Depth First Search" (DFS) and "Breadth-First Search". The authors proposed a method to construct a random walk synthesis of these two strategies.

• **TADW[2]:** TADW uses textual features to observe random walks on graphs and proposes the text-associated DeepWalk to incorporate node content features.

• **BANE[24]:** This model was proposed to embed the attributed network in a binary space. To capture the dependence of the network architecture on the data, a proximity matrix based on the Weisfeiler-Lehman kernel is defined. In addition, cyclic coordinate descent (CCD) is utilized to represent the binary node representation limitation.

• **TENE[50]:** This method subdigit the attributed network embedding issue as a mutual non-negative matrix factorization issue. In addition, the node embedding acquired from sub-tasks is utilized to organize them.

• **MUSAE[47]:** MUSAE learns attributed node representation, which expressly factorizes the production of an attributed matrix and a normalized adjacency matrix power.

C. PARAMETER SETTINGS

Considering the Weisfeiler Lehman Kernel (with parameter $K \in [5]$), the graphlet sampling (GS) kernel [51](with $K \in [3, 4, 5]$) and Relaxed Weisfeiler-Lehman Kernel[52] (with $K \in [5]$), we conducted an experiment on our Weisfeiler-Lehman kernel with a depth parameter $K \in [6]$. We found that the Weisfeiler-Lehman kernel gives the best results with depth parameter $K = 4$ for all datasets. In addition, after making the comparison of applying the employment of Sigmoid, Tanh, and ReLu in the proposed model, the ReLU function is used as the nonlinear activation function of the autoencoder. The details of the layer size and dimensions of the layers in the autoencoder are listed in Table 4. For all baselines, we used the public source code provided by Karate Club[49] and tuned the parameters to ensure that each model has achieved the optimal performance on all datasets and different experiments.

| Datasets       | Number of neurons in each layer |
|----------------|---------------------------------|
| FACEBOOK       | 4714-1000-500-128-500-1000-4714 |
| WIKI           | 5170-800-400-128-400-800-3170   |
| GITHUB         | 13183-2000-500-128-500-2000-13183|
| TWITCH         | 4005-1000-128-1000-4005         |

D. EXPERIMENTAL RESULTS

This section presents the results of comparative evaluation experiments of the proposed model against all the baselines. To compare the efficiency of the node embedding models, we used node classification, correlation prediction, and visualization tasks.

1) **NODE CLASSIFICATION RESULTS**

Classifying nodes is a significant task. It aims to predict the labels of the nodes using the same known information of the network. This is a widespread task to determine the efficiency of network embedding achievement. In the experiment, we first learned the node representation over different models. These node representations were used as features to classify each node into a set of classes. Then, some nodes were randomly sampled for training, while the others were sampled for testing. We randomly selected 90%, 70%, 50%, 30%, and 10% nodes as the testing set and the remaining as the training set, respectively. Classification implementation was evaluated by training the SVM as the classifier. Macro-F1 and Micro-F1 were employed as the evaluation metrics. If there are $C$ classes in the test set and $TP_i, FP_i, TN_i$ and $FN_i$ are the true positive, false positive, true negative, and true negative of class $i$, then the Micro – F1 and Macro – F1 might be as follows:

\[
\text{Macro - F1} = \frac{1}{C} \sum_{i=1}^{C} \frac{F1 - \text{score}_i}{C}.
\]

Where
\[
\text{precision} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FN_i)}, \quad \text{recall} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FP_i)}.
\]

The node classification results are presented in Table 5. For each tuning, the best and second-best models are highlighted in boldface and underlined, respectively. From these results, the following conclusions can be drawn:

• **Plain vs. Plain:** DeepWalk shows similar or better results than Node2vec in different datasets. Compared to HOPE, these methods based on random walks work better, mainly because these methods can explore diverse information about the structure of the network through biased random walks.

• **Plain vs. Attribute:** The results show that the methods that integrate the attributes of the nodes have better experimental results than those that focus only on the structure. This shows that the integration of structural information and the semantics of the attributes is advantageous for learning the vectors.

• **Attribute vs. Attribute:** DANE-WLA has achieved the best performance in terms of the Micro-F1 and Macro-F1 values of all methods for most configurations. The ranking results are higher than those of the remaining reference methods for a percentage of $3\%$ for the FACEBOOK dataset. The results show that DANE-WLA is efficient and robust in handling both structure and attribute information.

2) **LINK PREDICTION**

Link prediction is an outstanding implementation of network embedding that endeavors to infer the probability of the presence of edges between nodes that are not connected in the input network. We created the test set by combining 20% of the randomly selected edges with an equal number of nonexistent edges. Link prediction was measured by two prevalent evaluation metrics: AUC and precision [53].

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
AUC is the area under the receiver-operating characteristic curve. It is a stable measurement used to define the average accuracy over the entire range of test values. The trivial AUC of the random estimation method was 0.5. In short, when the value is larger, the performance improves.

**TABLE 5. THE RESULT OF NODE CLASSIFICATION ON FACEBOOK, WIKIPEDIA, GITHUB AND TWITCH DATASETS WITH DIFFERENT MODELS.**

| Dataset | Model  | Micro-F1 (%) | Macro-F1 (%) |
|---------|--------|--------------|--------------|
|         |        | 0.10  | 0.30  | 0.50  | 0.70  | 0.90  | 0.10  | 0.30  | 0.50  | 0.70  | 0.90  |
| FACEBOOK| HOPE   | 68.35 | 66.40 | 65.20 | 64.71 | 61.58 | 62.45 | 61.00 | 59.78 | 58.65 | 54.70 |
|         | DeepWalk | 70.72 | 69.26 | 69.77 | 68.87 | 63.64 | 67.62 | 67.02 | 67.14 | 66.83 | 60.80 |
|         | Node2Vec | 70.01 | 70.70 | 69.70 | 69.44 | 65.39 | 67.55 | 68.70 | 67.55 | 67.08 | 63.06 |
|         | BANE   | 63.73 | 64.74 | 64.03 | 64.02 | 60.39 | 57.61 | 58.42 | 57.13 | 57.42 | 48.96 |
|         | TADW   | 64.35 | 65.38 | 64.74 | 64.62 | 63.04 | 59.96 | 61.59 | 60.79 | 60.75 | 59.73 |
|         | MUSAE  | 90.74 | 90.35 | 89.75 | 89.49 | 88.48 | 89.86 | 89.63 | 88.87 | 88.56 | 87.40 |
|         | DANE-WLA | 94.42 | 93.92 | 93.86 | 93.29 | 90.13 | 94.35 | 93.65 | 93.46 | 92.89 | 89.28 |
| WIKI    | HOPE   | 80.58 | 81.35 | 80.62 | 79.23 | 78.31 | 70.70 | 72.18 | 70.42 | 68.19 | 64.59 |
|         | DeepWalk | 89.86 | 89.31 | 88.80 | 85.28 | 81.75 | 86.61 | 86.39 | 84.56 | 80.21 | 74.51 |
|         | Node2Vec | 89.06 | 88.73 | 88.17 | 85.97 | 82.13 | 88.54 | 85.32 | 84.47 | 81.31 | 76.06 |
|         | BANE   | 72.68 | 72.46 | 71.44 | 71.50 | 71.53 | 47.07 | 45.84 | 41.67 | 41.69 | 41.70 |
|         | TADW   | 86.85 | 85.67 | 83.85 | 83.92 | 80.60 | 82.25 | 80.38 | 77.85 | 77.76 | 72.00 |
|         | MUSAE  | 89.43 | 88.85 | 88.47 | 88.21 | 85.34 | 86.47 | 87.28 | 86.80 | 86.81 | 81.59 |
|         | DANE-WLA | 89.86 | 89.40 | 89.05 | 88.46 | 85.73 | 88.55 | 87.56 | 86.99 | 86.76 | 81.69 |
| GITHUB  | HOPE   | 79.95 | 80.51 | 79.72 | 79.42 | 78.70 | 6617 | 66.87 | 64.88 | 64.47 | 61.49 |
|         | DeepWalk | 84.48 | 84.94 | 84.94 | 84.49 | 84.11 | 77.70 | 78.18 | 78.20 | 77.36 | 77.16 |
|         | Node2Vec | 84.85 | 85.08 | 85.08 | 84.40 | 84.18 | 77.80 | 78.22 | 78.28 | 77.34 | 46.44 |
|         | BANE   | 73.34 | 73.95 | 74.15 | 74.14 | 74.21 | 42.31 | 42.51 | 42.58 | 42.58 | 42.60 |
|         | TADW   | 85.09 | 83.32 | 83.64 | 83.77 | 83.62 | 78.31 | 76.29 | 76.90 | 76.81 | 76.65 |
|         | MUSAE  | 85.52 | 86.42 | 86.31 | 86.13 | 86.16 | 79.23 | 80.51 | 80.52 | 80.22 | 80.15 |
|         | DANE-WLA | 86.65 | 86.63 | 86.74 | 86.22 | 85.94 | 81.43 | 80.53 | 80.63 | 80.21 | 79.60 |
| TWITCH | HOPE   | 59.82 | 56.83 | 57.11 | 56.69 | 54.82 | 49.52 | 45.43 | 47.80 | 47.14 | 46.22 |
|         | DeepWalk | 59.24 | 59.63 | 55.93 | 56.97 | 52.01 | 57.27 | 58.22 | 54.64 | 56.16 | 52.01 |
|         | Node2Vec | 57.81 | 57.81 | 59.19 | 57.60 | 54.43 | 56.63 | 56.55 | 58.03 | 56.82 | 53.79 |
|         | BANE   | 57.78 | 54.86 | 55.40 | 53.86 | 54.08 | 47.46 | 48.66 | 45.51 | 52.54 | 42.76 |
|         | TADW   | 59.19 | 56.88 | 57.48 | 58.31 | 55.34 | 57.13 | 53.84 | 54.36 | 55.53 | 53.63 |
|         | MUSAE  | 59.54 | 58.74 | 57.90 | 58.79 | 55.82 | 57.70 | 57.04 | 56.15 | 57.54 | 54.95 |
|         | DANE-WLA | 63.67 | 62.23 | 59.19 | 60.73 | 57.55 | 62.79 | 61.44 | 58.97 | 59.65 | 56.05 |
Visualization is an application of network analysis and is an effective way to measure the quality and efficiency of an embedding model. Visualization clarifies the global characteristics of embedding by understanding whether nodes of the same type are sufficiently close to each other in the embedding space. First, we represent nodes from the Facebook dataset in a low-dimensional embedding space using various models. This information is then used as an input for the t-SNE tool [54]. Therefore, each node is represented as a 2D vector, and all nodes are displayed in a 2D space. Each color represents a node with the same label. When the points of the same color are in close clusters, this indicates that the performance of the embedding model is good. The network visualization with different embedding methods is shown in FIGURE 2. It can be seen that the points are distributed in four different colors representing four separate clusters. It is clear that our model achieves better compactness and separation of clusters when it is compared to other reference models. Points of similar color resulting from DANE-WLA model appear together in the same space. While in other models the clusters did not appear separately from each other. Therefore, our model may work the best for both supervised and unsupervised tasks.

### 3) VISUALIZATION

Visualization of different approaches for the Facebook dataset.

### 4) PARAMETERS ANALYSIS

In this evaluation method, the impact of the depth parameter K on the performance of our DANE-WLA model was first analyzed, which varied from 1 to 6. We found that DANE-WLA provided the best results with a depth parameter K=4 for all datasets. FIGURE 3 shows the micro-F1 and macro-F1 results obtained by varying the K parameters when setting the training ratio=80% and embedding dimension =64. Second, as shown in FIGURE 4, the proposed model provided stable results when the training data percentage became equal to 30% of the total data. Finally, we tested the Micro-F1 and Macro-F1 scores of DANE-WLA with embedding dimensions varying from 32 to 256, as shown in FIGURE 5. From the figure, we can observe that the proposed model yields good results with different embedding sizes, and we note that the performance is directly proportional to the dimension of the embedding size.
A. Al-furas et al.: Deep Attributed Network Embedding via Weisfeiler-Lehman and Autoencode

FIGURE 3. Comparisons of Weisfeiler-Lehman kernel depth size.

FIGURE 4. Comparisons of training ratio.

FIGURE 5. Comparisons of embedding dimensions on the different datasets.
V. CONCLUSION

In this paper, a new deep attribute network embedding method called DANE-WLA was proposed. We argued that plain methods ignore the integration between the network structure and the attributed information. Most ANE methods are shallow and fail to capture nonlinear deep information. In the same context, existing ANE methods are mainly designed for static networks, and they work with nonlinear complexity. Accordingly, we proposed the DANE-WLA method with linear complexity that tackled the implicit data, invisible nodes, missing data, structure and attribute integration, and nonlinear patterns of ANE in an integrated framework. We found that Weisfeiler-Lehman embedding is efficient in capturing dependencies among node links and attributes. In addition, using an autoencoder is significant for investigating the nonlinear and complex patterns in the produced Weisfeiler-Lehman matrix. Experiments with link prediction, node classification, and visualization tasks showed that the DANE-WLA model outperformed the other network embedding models across the four real datasets. In addition, the model was evaluated using several advanced parameter sensitivity analyses and model contributions, and the results showed the superiority of DANE-WLA.

REFERENCES

[1] N. K. Ahmed et al., “A framework for generalizing graph-based representation learning methods,” arXiv preprint arXiv:1709.04596, 2017. [Online]. Available: https://arxiv.org/abs/1709.04596.
[2] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, “Network representation learning with rich text information,” in IJCAI, 2015, vol. 2015, pp. 2111-2117.
[3] I. Makarov et al., “Survey on graph embeddings and their applications to machine learning problems on graphs,” PeerJ Computer Science, 2021. [Online]. Available: https://peerj.com/articles/cs-357.pdf.
[4] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications.” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 9, pp. 1616-1637, 2018.
[5] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” arXiv preprint arXiv:1706.02216, 2017.
[6] W. Wang, H. Yin, X. Du, W. Hua, Y. Li, and Q. V. H. Nguyen, “Online user representation learning across heterogeneous social networks,” in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 545-554.
[7] A. Bojchevska and S. Giimmen, “Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking,” arXiv preprint arXiv:1707.03815, 2017. [Online]. Available: https://arxiv.org/abs/1707.03815.
[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in Proceedings of the 24th international conference on world wide web, 2015, pp. 1067-1077.
[9] A. M. Fout, “Protein interface prediction using graph convolutional networks,” Colorado State University, 2017.
[10] S. Zhu, K. Yu, Y. Chi, and Y. Gong, “Combining content and link for classification using matrix factorization,” in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007, pp. 487-494.
[11] S. Gao, L. Denoyer, and P. Gallinari, “Temporal link prediction by integrating content and structure information,” in Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pp. 1169-1174.
[12] L. Cui, J. Wu, D. P. Zhang, and P. Kennedy, “Dual implicit missing-based latent friend recommendation,” IEEE transactions on systems, man, and cybernetics: systems, vol. 50, no. 5, pp. 1663-1678, 2018.
[13] H. Sun et al., “Network embedding for community detection in attributed networks,” ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 14, no. 3, pp. 1-25, 2020.
[14] B. Hu, H. Wang, X. Yu, W. Yuan, and T. He, “Sparse network embedding for community detection and sign prediction in signed social networks,” Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 1, pp. 175-186, 2019.
[15] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701-710.
[16] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1225-1234.
[17] Y. Liu, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2015, vol. 29, no. 1.
[18] S. Zhu, J. Li, H. Peng, S. Wang, P. S. Yu, and L. He, “Adversarial directed graph embedding,” ed. 2020a, 2020.
[19] M. Belkin and P. Nyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” Neural computation, vol. 15, no. 6, pp. 1373-1396, 2003.
[20] J. B. Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” science, vol. 290, no. 5500, pp. 2319-2323, 2000.
[21] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855-864.
[22] Q. Wang, H. Yin, W. Wang, Z. Huang, G. Guo, and Q. V. H. Nguyen, “Multi-hop path queries over knowledge graphs with neural memory networks,” in International Conference on Database Systems for Advanced Applications, 2019: Springer, pp. 777-794.
[23] H. Gao and H. Huang, “Deep attributed network embedding,” in Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI), 2018.
[24] H. Yang, S. Pan, P. Zhang, L. Chen, D. Lian, and C. Zhang, “Binarized attributed network embedding,” in 2018 IEEE International Conference on Data Mining (ICDM), 2018: IEEE, pp. 1476-1481.
[25] C. Li et al., “PPNE: property preserving network embedding,” in International Conference on Database Systems for Advanced Applications, 2017: Springer, pp. 163-179.
[26] L. Liao, X. He, H. Zhang, and T.-S. Chua, “Attributed social network embedding,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 12, pp. 2257-2270, 2018.
[27] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, “Tri-party deep network representation,” Network, vol. 11, no. 9, p. 12, 2016.
[28] X. Huang, J. Li, and X. Hu, “Accelerated attributed network embedding,” in Proceedings of the 2017 SIAM international conference on data mining, 2017: SIAM, pp. 633-641.
[29] R. Hong, Y. He, L. Wu, Y. Ge, and X. Wu, “Deep attributed network embedding by preserving structure and attribute information,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019.
[30] W. Yu, W. Cheng, C. Aggarwal, B. Zong, H. Chen, and W. Wang, “Self-attributive attributed network embedding through adversarial learning,” in 2019 IEEE International Conference on Data Mining (ICDM), 2019: IEEE, pp. 758-767.
[31] W. Yu et al., “Learning deep network representations with adversarially regularized autoencoders,” in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 2663-2671.
[32] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, “Weisfeiler-lehman graph kernels,” Journal of Machine Learning Research, vol. 12, no. 9, 2011.
[33] H. Yang, L. Chen, M. Lei, L. Niu, C. Zhou, and P. Zhang, “Discrete Embedding for Latent Networks,” in IJCAI, 2020, pp. 1223-1229.
[34] H. Yang, L. Chen, S. Pan, H. Wang, and P. Zhang, “Discrete embedding for attributed graphs,” Pattern Recognition, vol. 123, p. 108368, 2022.
[35] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, “RiWalk: Fast structural node embedding via role identification,” in 2019 IEEE International Conference on Data Mining (ICDM), 2019: IEEE, pp. 478-487.
[36] X. He and P. Niyogi, “Locality preserving projections,” Advances in neural information processing systems, vol. 16, no. 16, pp. 153-160, 2004.
[37] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1105-1113.
[38] D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, “From properties to links: Deep network embedding on incomplete graphs,” in Proceedings of the
This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3181120

A. Al-furas et.al: Deep Attributed Network Embedding via Weisfeiler-Lehman and Autoencoder

Amr Al-furas received the B.S. degree in Mathematics & 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, Egypt., in 2017. He is currently pursuing a Ph.D. degree in Computer Sciences at Mansoura University, Mansoura, Egypt. He worked as a demonstrator from 2009 to 2013 in Ibb University. His current research interests are AI, Graph analytics, social network analysis, and recommender systems.

Mohammed alrahmawy Received both B.Eng, in Electronics Engineering and M.Sc. in Automatic Control Engineering from Mansoura University in 1997 and 2001 respectively. In 2003, he joined Computer Science Department at the Faculty of Computers and Information at Mansoura University as a teaching assistant. His M.Sc. thesis received the award of best M.Sc. thesis in Computer Science from Mansoura University in 2003. Then, in 2005 he was awarded a fully-funded Ph.D. scholarship from the Egyptian government to do a Ph.D. at the Realtime Systems Research Group, University of York, UK. He received his Ph.D. in 2011 in Computer Science from the University of York, U.K. Later, in 2011, he joined back the Department of Computer Science, Mansoura University, as a Lecturer. Since 2017, he has been working as an Associate Professor in the same department. His current research interests include Realtime Systems and Languages, Cloud Computing, Distributed and Parallel Computing, AI, Graph analytics, Soft Computing, Image Processing, Computer Vision, IoT, and Big Data.

Waleed Mohamed Al Adrousy: He was born in Maca, KSA in 1982. He is an Egyptian researcher. He received BSc, MSc and Ph.D. degrees of computer science from Faculty of Computer and Information sciences, Mansoura university in 2004, 2010 and 2015 respectively. He worked as a demonstrator from 2004 to 2010, and as a teaching assistant from 2010 to 2015, and as lecturer from 2015 till present in the Faculty of Computer and Information Sciences, Mansoura University. He is interested in several scientific fields like: Social Network Analysis, Recommender Systems, Distributed Systems, Game Development, 3D Modeling, Simulation, Mobile Development, Security and Software Engineering.

Samir Elmougy received a 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 at the Dept. of Computer Science, Faculty of Computers and Information, Mansoura University, Egypt. From 2008 to 2014, he was an Assistant Professor at the Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He published over 70 papers in refereed IEEE Transactions/Springer Journals, IEEE conferences, and book chapters. Also, he participated in the reviewing process of many international refereed journals and conferences. His current research interests include Artificial Intelligence, IoT, Information Theory, and Software Engineering.

References

1. C. Li, M. Shi, B. Qu, and X. Li, "DeepAttributedNetworkRepresentationLearningviaAttributeEnhancedNeighborhood," arXiv preprint arXiv:2104.05234, 2021. [Online]. Available: https://arxiv.org/abs/2104.05234.
2. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104-3112.
3. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111-3119.
4. F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 37-45.
5. A. Al-Furas, M. El-Dosuky, and T. Hamza, "Multi-column Deep Neural Network Based On Particle Swarm Optimization," in 2019 First International Conference of Intelligent Computing and Engineering (ICOICE), 2019; IEEE; pp. 1-5.
6. K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014. [Online]. Available: https://arxiv.org/abs/1406.1078.
7. S. Zhang and Z. Zhang, "Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs," in Proceedings of the 2015 SIAM international conference on data mining, 2015: SIAM, pp. 451-459.
8. S. B. Bozkurt Altindag, "Note on the sums of powers of normalized signedless Laplacian eigenvalues of graphs," Mathematics Interdisciplinary Research, vol. 4, no. 2, pp. 171-182, 2019.
9. B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," Journal of Complex Networks, vol. 9, no. 2, p. cnab014, 2021.
10. D. Zhang, J. Yin, X. Zhu, and C. Zhang, "SINE: scalable incomplete network embedding," in 2018 IEEE International Conference on Data Mining (ICDM), 2018: IEEE, pp. 737-746.
11. B. Rozemberczki, O. Kiss, and R. Sarkar, "Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs," in Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 3125-3132.
12. S. Yang and B. Yang, "Enhanced network embedding with text information," in 2018 24th International Conference on Pattern Recognition (ICPR), 2018; IEEE, pp. 326-331.
13. N. Sherfashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in Artificial intelligence and statistics, 2009: PMLR, pp. 488-495.
14. T. H. Schulz, T. Horváth, P. Welke, and S. Wrobel, "A generalized weisfeiler-lehman graph kernel," arXiv preprint arXiv:2101.08104, 2021. [Online]. Available: https://arxiv.org/abs/2101.08104.
15. H. Wang and Z. Le, "Seven-Layer Model in Complex Networks Link Prediction: A Survey," Sensors, vol. 20, no. 22, p. 6560, 2020.
16. L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of machine learning research, vol. 9, no. 11, 2008.