Shin *et al. Journal of Big Data* (2025) 12:219
https://doi.org/10.1186/s40537-025-01270-y

Journal of Big Data

RESEARCH Open Access

WalkGCN: a biased sampling strategy for GNNs on non-attributed graphs

Mincheol Shin[1], Taeyoung Choe[1], Yejong Ryu[1], Yanggon Kim[2] and Mucheol Kim[1*]

*Correspondence:
Mucheol Kim
kimm@cau.ac.kr
[1]Department of Computer Science and Engineering, Chung-Ang University, 84, Heukseok- ro, Dongjak-gu, Seoul 06974, Korea
[2]Department of Computer and Information Sciences, Towson University, 8000 York Rd, Towson, MD 21252, USA

Abstract
Graph Neural Networks (GNNs) typically assume the presence of node attributes to capture interactions in a graph structure. However, real-world graph data often has incomplete or completely-missing attribute information. GNN approaches to dealing with incomplete attributes are widely implemented, yet there is a paucity of research on graphs without attributes. DeepWalk and node2vec can be exploited to generate artificial attributes in Graph Convolutional Networks (GCN). However, the stochastic nature of random walks disrupts consistent performance. Furthermore, it introduces a degree bias, which causes the over-sampling of hub nodes and the under-representation of low-degree nodes. To address this limitation, we propose Walk Graph Convolutional Networks (WalkGCN) for generating artificial node attributes. WalkGCN employs a biased sampling strategy that mitigates degree-induced bias during node sequence generation. It increases the sampling frequency of leaf nodes to balance the training data. The node embeddings generated in the preceding phase are employed as artificial attributes for each node. The graph with artificial attributes is then fed to the vanilla graph convolutional networks model, which performs node classification. The results of extensive experiments on three graph datasets demonstrate that WalkGCN effectively generates artificial attributes that are independent of the hyperparameters or randomness of the random walk. Furthermore, the proposed model outperforms a variety of baseline models. The code is available at https://github.com/mincheol-shin-cau/WalkGCN.

Keywords Graph embedding, Attribute missing, Graph neural networks, Artificial attributes, Random walk

Introduction

Graph Neural Networks (GNNs) have become a paradigm-shifting class of models in the field of artificial intelligence, attracting considerable interest from both academic and industrial communities [1, 2]. Despite their robust capabilities, GNNs encounter considerable challenges when confronted with incomplete attribute information, a common scenario in real-world data [3, 4]. Most methods assume fully observed node attributes. However, this assumption is unrealistic in real-world scenarios [5–9]. For example, a significant portion of sensitive data is unreported in areas such as social networks [5, 6], where privacy concerns are prevalent [4, 9]. The significant decrease in accuracy when

Springer Open

© The Author(s) 2025. **Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

the missing rate of graph attributes is high indicates that incomplete data has a serious impact on the performance [7, 10]. This limitation requires the development of a novel GNN framework that can effectively handle the lack of complete attribute information.

Graph learning often relies on node attributes, which are frequently missing or unreliable in practice. Traditional techniques such as matrix factorization [11] and K-nearest neighbor methods [12] impute missing values based on observed attributes. Simpler strategies replace missing values with zeros or attribute-wise means [13]. Although these strategies are easy to apply in cases of missing attributes, they often introduce artificial distortions that hinder accurate prediction [3, 5]. Graph completion learning approaches reconstruct missing node attributes by leveraging both structural and attribute information [2, 9]. For instance, the Structure-Attribute Transformer (SAT) applies distributional matching techniques to align the distributions of node features and structural patterns [3]. The Structured Variational Graph Autoencoder (SVGA) models intra-graph correlations through structured variational inference [8]. SVGA helps stabilize latent representations by mitigating the risk of overfitting. However, they only work in environments where node attributes are available, making them unsuitable for non-attributed graph environments.

GNNs address various graph-related problems in scenarios where explicit node attributes are available (a, b in Fig. 1). Despite their effectiveness in solving problems, their ability to address scenarios with completely missing attributes remains limited (c in Fig. 1). Cui et al. [5] propose a method to generate artificial attributes using DeepWalk. However, this method has low robustness due to high variance while hyperparameter tuning is required for optimal performance. These limitations arise from the degree bias inherent in random walks. The random walk process disproportionately selects higher-degree nodes, leading to oversampling and increased risks of overfitting [14]. In contrast, low degree nodes suffer from under sampling and insufficient learning signals. This phenomenon is related to degree bias [15] and the friendship paradox [40]. Furthermore, random walk models struggle to capture local structural contexts accurately, whereas graph neural networks (GNNs) explicitly aggregate neighborhood information. It leads to biased embeddings and limited generalisation performance [15–17].

This paper proposes a biased sampling strategy that addresses the inherent randomness of random walks for GNNs on non-attributed graphs. The proposed model employs a graph embedding approach to generate artificial attributes in non-attributed graph environments. During node sequence generation, the sampling rate of leaf nodes (nodes with a degree of 1) is increased, with each sampling process rigorously controlled. This addresses degree-induced sampling imbalance during training. It also manages the

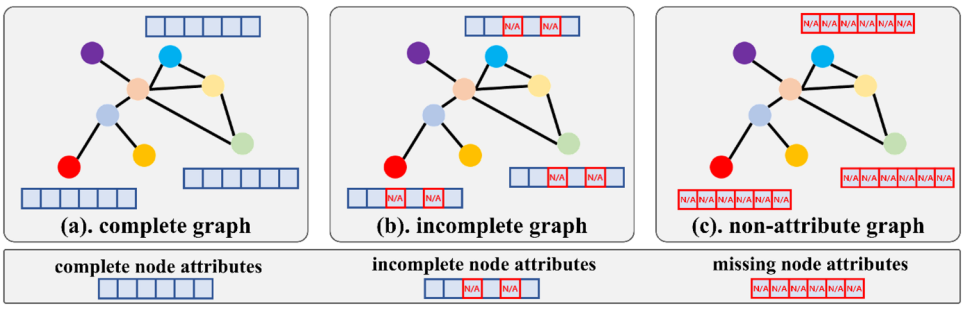


Fig. 1 Graphs are categorized into three types based on the completeness of the node properties

randomness that arises during the node sequence generation process. The generated node embeddings are fed into the Graph Convolutional Networks (GCN) [16] model as artificial attributes. The framework integrates with standard GCN layers without modification. The main contributions are as follows:

1. We introduce WalkGCN, a model that learns GCNs with only the topology information by adequately generating artificial attributes. The presented model facilitates the efficacious implementation of GCN, even when attribute information is not available.

2. Our method proposes a biased sampling strategy that adjusts the sampling rates of hub and leaf nodes and strictly controls the sampling process. By explicitly controlling the node-degree bias inherent in random walks, our approach mitigates the oversampling of hub nodes. It improves the efficiency of neighbor representation learning.

3. WalkGCN outperforms other non-attributed or artificially attributed baselines and even surpasses several attribute-using models. Furthermore, our model demonstrates the capability to be integrated with various GNN models.

Related work

Node embedding methods for incomplete graphs

Several methods improve the performance of graph embeddings on attribute-missing graphs. Deep Outlier aware attributed Network Embedding (DONE) [15] points out that the influence of outlier nodes has a significant impact on graph embeddings. They present two methods to reduce the influence of outliers. One approach is deep outlier DONE, which is based on unsupervised learning. The other is Adversarial ONE (AdONE) [15], which applies adversarial learning.

Graph completion learning

Alternatively, researchers apply graph completion learning methods to fill the missing attributes [2, 9]. GCNs for graphs containing missing attributes [17] incorporate Gaussian mixture models to predict the neuronal activation occurring in the first hidden layer of the network to enhance the representation of the missing data. SAT [3] adopts a unique approach where it treats graph structures and node attributes separately. It rigorously models the combined distribution of both elements by applying distribution matching techniques. The model also provides a methodology for the completion of missing attributes, allowing it to integrate seamlessly with existing GCN and Graph Attention Networks (GAT) [18] frameworks. SVGA [8] employs a structured variational inference approach to estimate missing node attributes in graphs. It integrates the Gaussian Markov random field to model the distribution of latent variables, taking advantage of the graph's structural information. PCFI [7] introduces a channel-wise confidence measure for each imputed feature. The model calculates confidence using pseudo-confidence derived from the shortest path distance to the nearest node with known attributes. WAGE [19] adopts a weight distribution encoder, which tightly integrates structural and attribute information to enable reliable attribute reconstruction in graphs with missing node features. Recent studies on heterogeneous graphs have also proposed various models. HAGNN-AC [20] completes missing node attributes by aggregating the attributes of neighboring nodes with available features in a weighted manner. This approach allows end-to-end integration with GNNs. HetReGAT-FC [21] introduces a residual graph attention network that performs attribute completion without pretraining. It improves

performance through the use of multiple edge types and does not rely on predefined meta-paths, which allows effective exploration of heterogeneous information.

Graph embedding methods for non-attributed graphs

The issue of entirely absent attributes in graphs remains a significant challenge in the field, despite the numerous studies on graphs with missing attributes. One possible approach is to utilize only the nodes and edges in the graph embedding. Both DeepWalk [22] and node2vec [23] leverage random walks and skip-grams to perform node pair sampling and node embedding learning. node2vec introduces a biased random walk strategy that integrates the concepts of depth-first and breadth-first search. This strategy overcomes the limitation of random exploration on DeepWalk. It enables researchers to control the exploration of the random walk using a tunable parameter. GraRep [24] leverages adjacency matrices to decompose k-dimensional adjacency matrices, which, in turn, extracts embeddings from each adjacency matrix. High-Order Proximity preserved Embedding (HOPE) [25] points out that existing graph embedding schemes have difficulty in preserving asymmetric transitivity adequately in directed graphs. It proposes a scalable embedding algorithm that exploits a variety of higher-order proximity measures. Hierarchical Representation Learning for Networks (HARP) [26] introduces a low-dimensional node embedding technique focused on preserving higher-order structural patterns. It avoids the common problems associated with non-convex objective functions found in traditional methods such as DeepWalk and node2vec by compressing the input graph. VERtex Similarity Embeddings (VERSE) [27] proposes a graph embedding method that explicitly learns all similarity measures between nodes. This method generates node embeddings by minimizing the Kullback–Leibler divergence from a personalized PageRank matrix to a low-dimensional approximation. Force2Vec [28], on the other hand, achieves high parallelization performance by shifting the core computation to a linear algebraic approach. GEMFP [29] enhances node embeddings by integrating both lower-order and higher-order (motif-based) structural features. It demonstrates strong performance in community detection tasks on real-world non-attributed graphs. Cui et al. [5] empirically demonstrate that node-centric artificial feature generation, particularly based on positional features such as DeepWalk and eigenvectors, achieves superior performance over structural or global graph-based methods. This advantage is especially pronounced in positional node classification tasks on non-attributed graphs. Recently, T2-GNN [30] proposed a dual-distillation framework that separates feature and structure completion via a teacher model. This model addresses the issue of interference amplification, which arises from the mismatch between features and structure in incomplete graph settings.

Methodology

In this paper, we propose the WalkGCN, a biased sampling strategy for GNNs on non-attributed graphs. Figure 2 illustrates the architecture of WalkGCN. The method begins by generating node sequences through biased sampling. Subsequently, embedding vectors are calculated for each node using the sequences. They are adopted as attributes, constructing an attribute-complete graph. Finally, the GCN model performs node classification using this reconstructed graph.

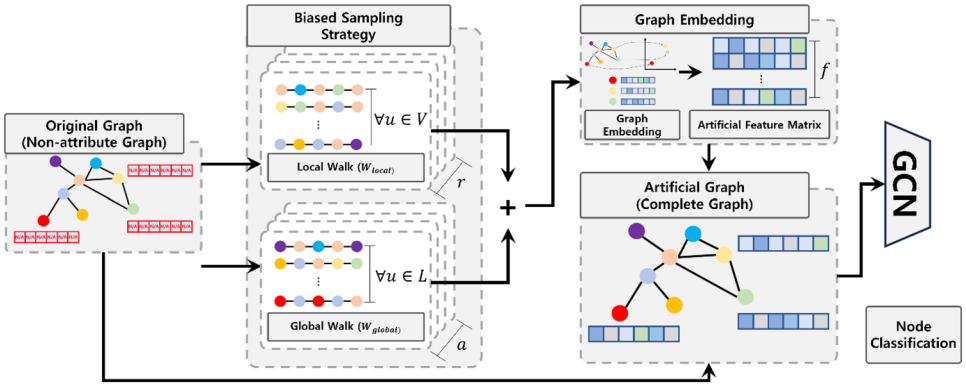


Fig. 2 Architecture of WalkGCN

Problem definition

Given a non-attributed graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges, the objective is to learn a node classification model. To this end, an artificial attribute matrix $T \in \mathbb{R}^{|V| \times d}$ is generated via biased random walks. This matrix is then fed into a GCN f_θ. The learning objective is to minimize the cross-entropy loss over the labeled nodes:

$$\mathcal{L} = -\sum_{v \in y_{train}} y_v \log\left(\hat{y}_v\right), \quad \hat{y}_v = \operatorname{softmax}\left(f_\theta\ (T)\right) \tag{1}$$

This formulation allows the model to learn node representations solely from graph structure, even in the complete absence of node attributes.

Biased sampling

We propose a novel biased sampling algorithm that incorporates strict sampling rules for the inclusion of leaf nodes. This approach resolves the issue of bias among nodes by reducing the proportion of hub nodes during the learning process, while increasing the proportion of leaf nodes. Furthermore, the sampling rules enhance the model's ability to learn from each node's neighboring node information. It achieves more effective graph embeddings in an intuitive and straightforward manner.

We first define the biased transition probability between nodes (Eqs. (2) and (3)) and then describe how these probabilities are incorporated into the random walk sampling procedure (Algorithm 1). In Eq. (2), $\alpha_{gs}(t, x)$ specifies unnormalized sampling weights based on node distance and the hyperparameters *g*(global) and *s*(local). This unnormalized sampling weight, $\alpha_{gs}(t, x)$, forms the basis for the biased transition probability used in Algorithm 1. In Eq. (3), c_i denotes the current node at the i-th step of the walk, and c_{i-1} denotes the previous node at the (i-1)-th step. v denotes the source node of the transition, and x denotes a candidate neighbor node. π_{vx} denotes the unnormalized transition probability from node v to node x. Z is the normalization constant that ensures the sum of all transition probabilities from a given node equals 1. This formulation of $\alpha_{gs}(t, x)$ and π_{vx} enables the biased random walk to sample nodes more effectively, thereby alleviating sequence bias and improving the efficiency of neighbor information learning.

Table 1 Comparison of node classification performance (Micro-F1 score) with existing models that learn on non-attribute graphs

Dataset	Model	Missing information (label)									
		99%	97%	95%	90%	80%	70%	60%	50%	40%	Rank avg
Cora	node2vec	58.86	69.15	72.67	74.12	77.70	80.00	83.10	81.70	82.80	3.11
	HOPE	44.16	46.28	47.02	53.85	57.91	60.97	63.63	64.62	63.74	7.11
	GraphWave	28.64	29.29	31.28	36.79	41.68	42.82	43.12	43.21	43.74	9.56
	SAT (GCN-X)	31.88	30.39	31.48	46.95	72.69	74.60	75.90	76.24	78.81	6.78
	SAT (GAT-X)	30.37	30.08	50.70	70.22	78.19	79.06	80.53	81.13	81.69	5.00
	HARP	53.23	63.49	70.19	70.88	73.60	76.37	76.62	77.18	78.60	4.89
	VERSE	33.79	41.34	49.71	63.08	69.82	71.31	75.14	78.54	79.89	6.11
	Force2Vec	**67.30**	*69.30*	*76.20*	*76.80*	*80.30*	*82.20*	**84.20**	*84.00*	*84.00*	*1.78*
	T2-GNN	29.01	29.25	30.06	33.52	41.10	47.46	53.28	58.91	62.23	9.44
	WalkGCN	*66.20*	**73.92**	**76.60**	**77.25**	**80.35**	**82.74**	*83.69*	**84.53**	**85.48**	**1.22**
Citeseer	node2vec	29.11	34.22	41.36	49.64	52.44	57.49	60.39	61.48	64.66	5.22
	HOPE	26.86	29.22	39.49	41.42	42.15	42.17	44.71	44.38	44.00	7.33
	GraphWave	33.53	37.21	38.62	39.59	40.71	41.36	42.02	42.27	42.54	7.33
	SAT (GCN-X)	26.86	34.40	28.21	50.28	39.39	57.86	62.98	*66.05*	66.13	5.11
	SAT (GAT-X)	30.97	31.07	45.43	53.09	*58.89*	*62.74*	*64.65*	66.04	66.65	3.44
	HARP	41.66	*48.12*	49.13	48.10	53.85	53.64	53.12	51.63	54.79	5.00
	VERSE	-	-	-	-	-	-	-	-	-	-
	Force2Vec	**48.43**	47.12	**53.77**	**56.76**	**60.42**	**64.17**	**66.35**	**69.02**	**70.79**	**1.22**
	T2-GNN	18.96	21.69	24.85	30.51	40.54	47.83	53.15	57.87	61.98	7.67
	WalkGCN	*45.33*	**51.46**	*52.03*	*53.81*	58.12	60.80	61.87	64.89	*69.35*	*2.56*
Pubmed	node2vec	71.72	75.98	76.44	77.36	77.42	78.11	79.00	78.94	79.10	4.11
	HOPE	47.81	63.05	66.80	68.60	68.70	69.88	70.89	69.53	69.81	4.11
	GraphWave	40.47	41.28	42.01	43.12	43.82	43.93	44.07	44.12	44.08	5.89
	SAT (GCN-X)	40.79	40.91	40.37	41.42	42.05	42.97	40.87	42.27	40.49	9.33
	SAT (GAT-X)	41.23	41.03	40.50	40.45	41.75	41.11	40.11	41.95	44.99	9.33
	HARP	70.14	*72.57*	75.26	75.89	78.50	79.11	80.07	80.03	80.22	4.11
	VERSE	49.33	52.85	54.45	59.20	62.32	64.06	65.43	65.62	66.44	6.89
	Force2Vec	**77.34**	*78.37*	*80.11*	**81.21**	*81.66*	82.26	82.30	82.50	83.05	*2.11*
	T2-GNN	62.19	58.87	58.87	77.32	80.33	*82.49*	*82.38*	*82.74*	**84.46**	3.44
	WalkGCN	*76.44*	**79.13**	**80.19**	*80.05*	**81.63**	**82.56**	**83.22**	**83.22**	*83.51*	**1.44**

$$\alpha_{gs}(t,x) = \begin{cases} \frac{1}{g} & if\ d_{tx} = 0 \\ 1 & if\ d_{tx} = 1 \\ \frac{1}{s} & if\ d_{tx} = 2 \end{cases} \tag{2}$$

$\alpha_{gs}(t, x)$ is the unnormalized probability of biased sampling. The hyperparameter global sampling (g) controls the probability of returning to the previous node, and local sampling(s) adjusts the probability of transition to a different community. d_{tx} represents the distance between the previous node t and the subsequent node x following the transition. d_{tx} denotes the shortest path distance between the previous node *t* and the candidate node *x*. For returning to the original node (d_{tx} = 0) the transition probability is $\frac{1}{g}$. For traversing to a completely new node (d_{tx} = 2) the transition probability is $\frac{1}{s}$. When moving to a node at a distance of 1 from the current node (d_{tx} = 1), the transition probability is set to 1.

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & if\ (v, x) \in E \\ 0 & otherwise \end{cases} \tag{3}$$

π_{vx} denotes the unnormalized transition probability that node v travels to node x. The initial state of this probability is unadjusted to ensure that the sum of all possible transition probabilities is 1. The normalization constant Z serves to scale each transition probability. We normalize by Z so that transition probabilities sum to 1. For W_{local}, g and s are set to 0.01 and 999, respectively, ensuring that each node predominantly learns from its neighbors. To enable the model to learn leaf nodes from global information, g and s are set to 999 and 0.01, respectively, for W_{global}. This configuration approximates the behavior of infinite values for practical purposes. It simplifies the implementation while preserving the near-deterministic transition behavior required for biased random walks, effectively reducing computational overhead and improving numerical stability. Our sampling strategy not only mitigates the bias of random walks towards hub nodes but also emphasizes the importance of neighboring nodes [16] in graph learning. The algorithm 1 shows the generation flow of biased sampling. For each node u in the graph V, the algorithm generates r local walk W_{local}, with u as the starting node and length l.

$$\forall u \in V, W_{local} = [(c_{0,1}, c_{1,1}, \ldots, c_{l-1,1}), \ldots, (c_{0,r}, c_{1,r}, \ldots, c_{l-1,r})] \quad (4)$$

Algorithm 1: Biased Sampling Strategy

Input: Graph *G*(*V*, *E*), Walk length *l*, Walks per node *r*, Additional walks per leaf node *a*
Output: Walks_total W_{total}
$\pi \leftarrow$ PreprocessModifiedWeights(*G*)
$G \leftarrow (V, E, \pi)$
Initialize W_{local}, W_{global} to Empty
for all nodes u ∈ *V* **do**
for *walk_iter* = 1 to *r* **do**
Initialize walk to [*u*]
for *inner_iter* = 1 to *l* **do**
curr ← walk[-1]
seq ← BiasedSampling(curr, π, 0.01, 999)
Append seq to walk
end for
Append walk to W_{local}
end for
if degree(u) = = 1 then
for walk iter = 1 to *a* **do**
Initialize walk to [*u*]
for inner iter = 1 to *l* **do**
curr ← walk[-1]
seq ← BiasedSampling(curr, π, 999, 0.01)
Append seq to walk
end for
Append walk to W_{global}
end for
end if
end for
$W_{total} \leftarrow W_{local} + W_{global}$
return walks_total

$c_{0,J}$ represents the initial node *u*, while $c_{1,J}$ through $c_{l-1,J}$ signifies the subsequent nodes visited following *u* in each *j*-th walk. The proposed approach generates a global walk to allow each leaf node to learn the structure of the graph in an extreme way. It addresses the problem that leaf nodes are overly focused on learning from neighboring nodes compared to other nodes [31, 32]. We define the set L of nodes with degree 1 in the graph V to generate the biased sampling for W_{global}. The overall space complexity

of Algorithm 1 is $\mathrm{O}\left((rn + aL)\, ld\right)$, which reflects the total number of sampled nodes stored in the walk sequences. r denotes the number of walks per node, a denotes the number of additional walks for each leaf node, l denotes the walk length, n denotes the total number of nodes, n_{leaf} denotes the number of leaf nodes, and d denotes the average degree of the graph.

$$V_{leaf} = \{u \in V \mid deg\,(u) = 1\} \tag{5}$$

For every u in the set V_{leaf} of leaf nodes, the model generates additional walks. The additional leaf node sampling W_{global} are represented as follows.

$$\forall\, u \in V, W_{global} = [(c_{0,1},\, c_{1,1},\, \ldots,\, c_{l-1,1}),\, \ldots,\, (c_{0,a},\, c_{1,a},\, \ldots,\, c_{l-1,a})] \tag{6}$$

Similar to W_{local}, W_{global} represents the order of node visits. W_{local} enhances neighbor information learning, while W_{global} compensates for the sparse connectivity of leaf nodes. The biased sampling combines W_{local} and W_{global}, generated for nodes with degree 1. It yields the comprehensive set of random walks, W_{total}.

$$W_{total} = W_{local} + W_{global} \tag{7}$$

Artificial attribute generation

The methodology for generating and learning artificial attributes is discussed below. The proposed algorithm initializes the artificial attribute vector u_t of each node u with the node2vec embedding vector f_u. It generates an artificial attribute graph $G_{artificial}$ = (V, E,T), which retains the structural information (edge, E) of the original graph while incorporating the newly generated artificial properties. E and T denote the structural information of the original graph and the newly generated attribute information, respectively.

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} f^{(l)}\Theta^{(l)}\right) \tag{8}$$

Algorithm 2: Create Artificial Node Attributes Based On node2vec.

Input: Graph G = (V, E), Dimensions d (default = 256), Context size k (default = 10), Walks total w (default = 40)
Output: Artificial node attribute graph $G_{artificial}$ = (V, E, T)
$f \leftarrow$ node2vec(k, d, w)
for all nodes $u \in V$ **do**
Initialize $u_t \leftarrow f_u$
end for
$G_{artificial} \leftarrow$ (V, E, T)
return $G_{artificial}$

Algorithm 2 generates artificial node attributes based on node2vec embeddings. Specifically, in the constructed graph $G_{artificial}$ = (V, E,T), the set T consists of node2vec embeddings f_u computed for each node u. These embeddings serve as the initial attribute matrix $f^{(0)}$ for the GCN. Node representations are then iteratively updated through the propagation process defined in Eq. (8). Here, $f^{(l)}$ denotes the node attribute matrix at the l-th layer, and $\Theta^{(l)}$ denotes the weight matrix of the corresponding GCN layer. The multiplication $f^{(l)}\Theta^{(l)}$ denotes the process of transforming the node embeddings of the l-th layer through the weights of that layer. It generates the input embeddings for the next layer. f_θ in Eq. (1) denotes the final prediction result after the entire GCN model has processed all layers. Therefore, $f^{(l)}\Theta^{(l)}$ denotes an intermediate

Table 2 Comparison of node classification performance (Micro-F1 score) with other artificial attribute generation strategy

Dataset	Model	Missing information (attribute, label)		
		99%	97%	95%
Cora	Random	30.92	29.88	30.86
	Eigen	39.31	53.90	68.01
	one-hot	45.66	62.35	69.22
	$\mathrm{WalkGCN_{cui}}$	62.17	70.56	74.50
	$\mathrm{WalkGCN_{p}}$	59.14	70.78	73.93
	$\mathrm{WalkGCN_{q}}$	59.16	69.77	71.10
	$\mathrm{WalkGCN_{random}}$	*65.23*	*72.56*	*75.10*
	$\mathrm{WalkGCN_{hub}}$	63.85	71.67	73.49
	$\mathrm{WalkGCN_{2}}$	63.71	70.36	74.09
	WalkGAT	62.89	68.60	73.11
	WalkGCN	**66.20**	**73.92**	**75.24**
Citeseer	Random	22.86	24.32	22.49
	Eigen	34.64	41.73	45.90
	one-hot	31.76	44.67	46.98
	$\mathrm{WalkGCN_{cui}}$	40.01	46.87	47.33
	$\mathrm{WalkGCN_{p}}$	41.16	47.01	49.55
	$\mathrm{WalkGCN_{q}}$	39.94	45.88	47.50
	$\mathrm{WalkGCN_{random}}$	*44.46*	*48.44*	*51.93*
	$\mathrm{WalkGCN_{hub}}$	44.11	48.29	49.20
	$\mathrm{WalkGCN_{2}}$	39.01	46.68	49.87
	WalkGAT	43.98	46.53	50.73
	WalkGCN	**45.33**	**51.46**	**52.03**

layer-wise transformation process, while f_θ denotes the final model prediction result. The GCN model updates node embeddings across multiple layers in a systematic manner. The final node representation for each node is obtained from the output of the last GCN layer.

In the subsequent stage, the node embedding vectors generated in the preceding step are employed for the purpose of performing node classification. These embedding vectors, which capture the structural and attribute information of the nodes, pass through a final layer to produce class predictions. The model optimizes the classification task by employing cross-entropy loss [33].

Experiments

We proceed to evaluate the node classification performance of the WalkGCN using three benchmark datasets. The experiments are designed to examine the following research questions:

RQ1. Does WalkGCN outperform state-of-the-art techniques in the node classification task on attribute-missing graphs?

RQ2. How effective are the artificial attributes generated using the proposed biased random walk for node classification?

RQ3. Does the proposed biased random walk mitigate the inherent randomness of random walk?

Experimental setup

Dataset. In order to prove the efficacy of our methodology, we adopt three widely used benchmark datasets for our experiments:

Cora [34]: Consists of 2,708 papers as nodes and 5,259 citation links as edges. Each paper is represented by a binary word vector, with classification into one of seven classes.

Citeseer [35]: Contains 3,327 papers as nodes and 4,732 citation links as edges. Each paper is represented by a binary word vector, with classification into one of 6 classes.

Pubmed [36]: Comprises 19,717 papers and 44,338 citation links related to diabetes. Papers are represented by TF/IDF-weighted word vectors from a 500-word dictionary, categorized into one of 3 classes.

To simulate attribute-missing scenarios, we apply random masking to the attribute matrix (X) by replacing selected entries with zeros to reflect absent attributes. "-X" refers to the masked version of the original attribute matrix in the context of SAT. In our experiments, we set the validation ratio to 20%. The training ratio is the complement of mi, and the test ratio is the complement of the sum of the training and validation ratios. WalkGCN has 2 to 4 GCN layers (The detailed settings are on the GitHub repository). A comparative analysis is conducted using the average results obtained after running all models for 10 iterations.

Baseline. In our experiments, we focus exclusively on baselines that operate without attributes. It includes node2vec [23], HOPE [25], GraphWave [37], SAT [3], HARP [26], VERSE [27], Force2Vec [28] and T2-GNN [30]. The mi ratio refers to labels.

Initialization of artificial attribute. We describe the artificial attribute initialization strategy for GCN in RQ 2. Models that require both random walks and additional walks are tested using the same number of walks.

Random. It generates an attribute vector for each node that follows a random distribution [5]. This strategy only provides information to distinguish each node.

One-hot. A unique one-hot vector is initialized for each node [38]. This method is also randomly generated based on a seed, similar to the random strategy.

Eigen: It generates a k-dimensional attribute vector for each node through normalization of the adjacency matrix and eigen-decomposition [38].

$\text{Walkgcn}_{\text{cui.}}$It employs the method proposed in [5] to conduct GCN while generating the artifical attribute.

$\text{WalkGCN}_{\text{p,q}}$ We perform artificial attribute generation from the perspective of node-2vec [22]. For WalkGCNp p and q are set to 2 and 0.5, respectively. For WalkGCNq they are set to 0.5 and 2, respectively.

$\text{WalkGCN}_{\text{random}}$,$\text{WalkGCN}_{\text{hub}}$ WalkGCN_{2}. These methods generate artificial attributes similar to our method. They select random nodes, hub nodes and nodes with a degree 2, respectively, for additional learning instead of leaf nodes.

$\text{WalkGCN}_{\text{GAT}}$: This strategy employs GAT layers in replacement of GCN layers within the methodology.

Node classification results (RQ 1)

We evaluate the performance of WalkGCN by comparing it with models designed for non-attributed graphs (Table 1). The experiments marked with a dash in Table 1 indicate implementations unavailable in our environment. In Tables 1 and 2, bold values indicate the best performance among the models, and underlined values represent the second-best performance for ease of comparison. Table 1 shows that our model is the most effective model in the absence of attributes. In particular, WalkGCN achieves significant improvements over existing random walk-based models. It shows performance

improvement of 0.59–7.34%, 1.48–17.24%, 2.69–4.72% compared to the best results of the node2vec models. WalkGCN also demonstrates superior results compared to attribute models. Our model consistently outperforms numerous alternative models in both the Cora and Pubmed datasets. It is noteworthy that while attribute models exhibit minimal performance improvement beyond an informativeness ratio of 20%, WalkGCN consistently improves as the informativeness ratio increases (Table 1). The results suggest that WalkGCN not only demonstrates robust generalizability but also achieves high performance without heavy reliance on attributes.

Despite its superior performance on most datasets, WalkGCN displays comparatively lower performance on the Citeseer dataset. This discrepancy is related to the structural characteristics of the dataset. When conducting random walks on each dataset, the proportion of hub nodes in the sampled data is 28.41% for Cora, 43.59% for Pubmed, and 23.13% for Citeseer. In contrast, the proportion of leaf nodes is 4.99% for Cora, 2.95% for Pubmed, and 7.12% for Citeseer. This structural difference indicates that random walks are less influenced by hub nodes in Citeseer, whereas they are more strongly biased in Cora and Pubmed. WalkGCN demonstrates strong performance on graphs such as Cora and Pubmed, where the imbalance between hub nodes and low-degree nodes is pronounced. In the dataset, low-level nodes are underrepresented because hub nodes dominate the sampled sequences. WalkGCN increases the learning opportunities of low-degree nodes through biased sampling, effectively mitigating the typical oversampling of hubs observed in random walk-based methods. On the other hand, Citeseer has low hub dominance during random walks due to its low average degree and high proportion of leaf nodes. A balanced structure reduces the severity of sampling bias, which reduces the advantage of WalkGCN's bias correction mechanism. Furthermore, under extreme conditions such as a 99% missing rate, the lack of strong neighborhood information exacerbates the limitations of biased sampling, as shown by the suboptimal results in Table 1. Force2Vec, which reconstructs node representations via matrix decomposition without heavy reliance on local structure, performs competitively on Citeseer. The results highlight the suitability of WalkGCN for graphs with severe degree heterogeneity, while Force2Vec is more effective on structurally balanced graphs.

Artificial attribute strategy results (RQ 2)

The following analysis focuses on WalkGCN's performance under diverse artificial attribute schemes. The experimental results are presented in Table 2. Our model achieves the highest performance across various settings and consistently outperforms baseline methods. WalkGCN exhibits superior performance in comparison to other strategies, demonstrating an improvement of 5.98–20.54% on Cora and 5.24–14.73% on Citeseer over methods such as random, eigen, and one-hot. This indicates that the generation of artificial attributes through network embedding techniques is more effective than traditional methods. Moreover, when compared with other strategies that similarly generate artificial attributes, our method yields superior results, with an improvement ranging from 0.14 to 7.04% on Cora and 0.29–7.48% on Citeseer. The underlined entries in the table correspond to models that rely on random node augmentation. While this strategy is simple and effective, it lacks consideration of structural roles and attribute distributions when selecting nodes. These limitations not only exacerbate data imbalance, but also lead to poor overall model performance. In contrast, WalkGCN emphasizes

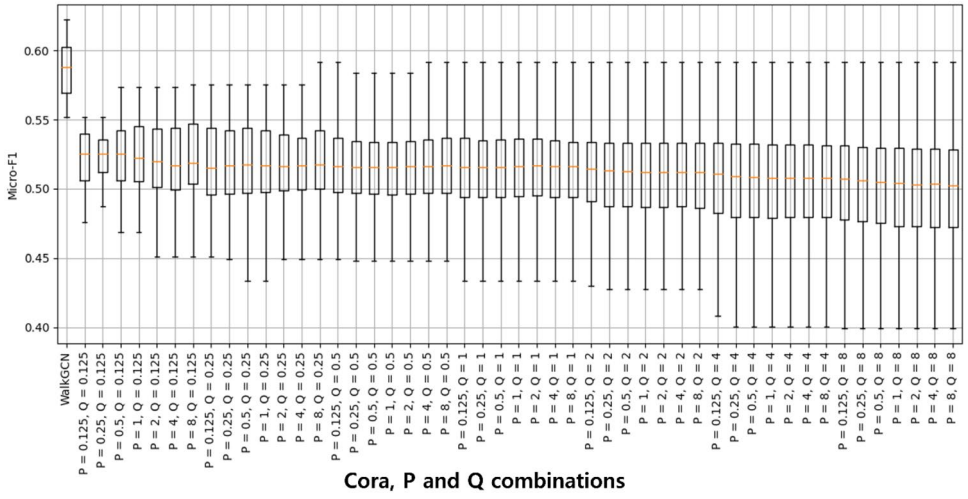


Fig. 3 Comparison of Micro-F1 Scores for WalkGCN's Biased Random Walk and Various Random Walk Strategies on the Cora Dataset (mi rate: 99%)

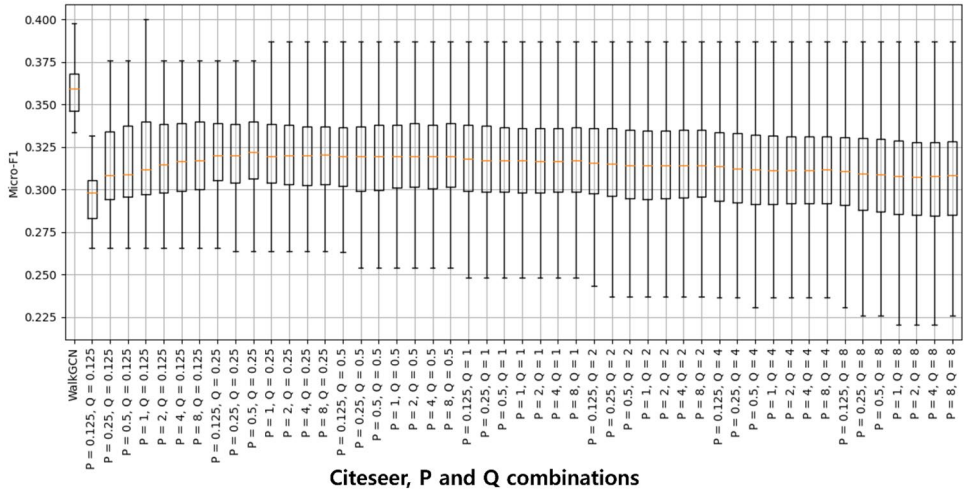


Fig. 4 Comparison of Micro-F1 Scores for WalkGCN's Biased Random Walk and Various Random Walk Strategies on the Citeseer Dataset (mi rate: 99%)

low-degree nodes by increasing their sampling frequency. It reduces the imbalance between hub and leaf nodes. The improved balance leads to better performance. This experimentally validates our solution to the learning imbalance between hub nodes and leaf nodes.

WalkGAT is designed to assess the feasibility of integrating the WalkGCN model with other frameworks. WalkGAT exhibits enhanced performance on the Citeseer dataset in comparison to the preceding WALKGCN model, which highlights the adaptability of our model across diverse GNN architectures.

Random walk randomness study (RQ 3)

In this study, we evaluate whether the biased random walk proposed in WalkGCN addresses the randomness issue of traditional random walks. Figures 3, 4 and 5 present the distribution of Micro-F1 scores obtained from 100 repeated experiments for each (p, q) configuration. Our biased random walk approach consistently shows the

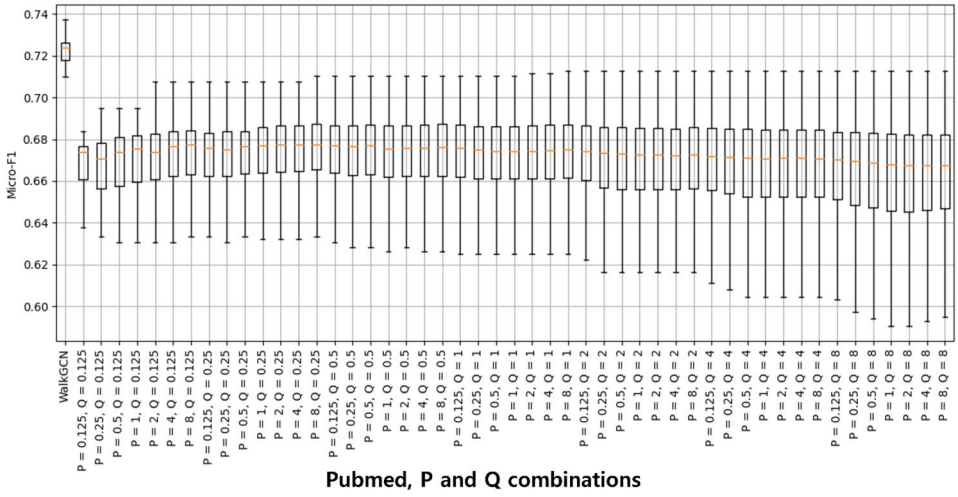


Fig. 5 Comparison of Micro-F1 Scores for WalkGCN's Biased Random Walk and Various Random Walk Strategies on the Pubmed Dataset (mi rate: 99%)

highest mean value in all cases. In particular, the box plots show that WalkGCN not only achieves the best average performance but also maintains a narrower interquartile range compared to other random walk strategies. This demonstrates that WalkGCN effectively mitigates the randomness inherent in traditional random walks. Moreover, across various combinations of p and q, WalkGCN consistently delivers stable and high-quality performance without the need for extensive hyperparameter tuning. These findings suggest that the proposed biased sampling strategy improves the model's robustness and its capacity for generalization.

Summary and limitations

Our experimental results demonstrate that WalkGCN offers a practical and robust solution for learning on non-attributed graphs. By constructing artificial node attributes through a dual-biased random walk strategy, the model effectively captures structural signals and consistently outperforms both random initialization and existing artificial feature generation methods across diverse datasets. WalkGCN is particularly effective on graphs with severe degree imbalance, such as Cora and Pubmed, where it mitigates hub-node oversampling by emphasizing low-degree nodes in its sampling process. It also shows strong robustness under extreme sparsity, maintaining low variance and stable classification accuracy even in the complete absence of node attributes.

However, WalkGCN faces limitations regarding the structural characteristics of the input graph. For example, in datasets like Citeseer, with its relatively high proportion of leaf nodes and more balanced node-degree distribution, the advantage gained from mitigating sampling bias is less pronounced. WalkGCN yields less pronounced performance improvements on structurally balanced graphs. This indicates a clear limitation that WalkGCN's effectiveness heavily relies on the imbalance between hub and leaf nodes. Future research should explore strategies to enhance its robustness and generalizability across graphs with various structural properties.

Conclusion

In this paper, we proposed WalkGCN, a biased sampling strategy designed for graph neural networks operating on graphs without node attributes. The method improves neighborhood information learning by reducing random walk variance and alleviating sequence bias caused by degree imbalance. WalkGCN generates synthetic node features by upsampling low-level nodes while tightly controlling the generation of node sequences. It allows us to effectively apply GCN in scenarios where attributes are missing. Experimental results on three benchmark datasets demonstrate that WalkGCN consistently outperforms other models designed for non-attributed graphs. We further validate its effectiveness through a systematic investigation of three research questions concerning performance, stability, and scalability. In future work, we plan to expand the experimental scope of WalkGCN by including heterogeneous graphs(i.e. BlogCatalog, Flickr, UAI2010), as well as non-attributed graphs such as TerroristRel. We aim to demonstrate the potential of WalkGCN as a general-purpose learning framework for non-attributed graphs by integrating it with recent domain-specific models.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2025-02217071) and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2025-02305436; Development of Digital Innovative Element Technologies for Rapid Prediction of Potential Complex Disasters and Continuous Disaster Prevention). This research was supported by the Chung-Ang University Graduate Research Scholarship in 2023.

Author contributions

Mincheol Shin and Mucheol Kim contributed to the conception of the idea and designed the complete study. Mincheol Shin performed the experiments. Mincheol Shin, Taeyoung Choe, Yejong Ryu, Yanggon Kim and Mucheol Kim analyzed the data, performed the statistical analysis, and completed the visualization. Mincheol Shin and Mucheol Kim wrote the first draft of the manuscript while Yanggon Kim and Mucheol Kim proofread the paper, and all authors approved this submitted version.

Data availability

No datasets were generated or analysed during the current study.

Declarations

Competing interests

The authors declare no competing interests.

Received: 28 December 2024 / Accepted: 24 August 2025

Published online: 26 September 2025

References

1. Li Y, Tarlow D, Brockschmidt M, Zemel R. Gated graph sequence neural networks. ArXiv Preprint arXiv:151105493. 2015.
2. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst. 2020;32:4–24.
3. Chen X, Chen S, Yao J, Zheng H, Zhang Y, Tsang IW. Learning on attribute-missing graphs. IEEE Trans Pattern Anal Mach Intell. 2020;44:740–57.
4. Jiang X, Qin Z, Xu J, Ao X. Incomplete graph learning via attribute-structure decoupled variational auto-encoder. Proceedings of the 17th ACM international conference on web search and data mining. 2024. pp. 304–12.
5. Cui H, Lu Z, Li P, Yang C. On positional and structural node features for graph neural networks on non-attributed graphs. Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022. pp. 3898–902.
6. Zhang X, Li M, Wang Y, Fei H, Amgcl. Feature imputation of attribute missing graph via self-supervised contrastive learning. ArXiv Preprint arXiv:230503741. 2023.
7. Um D, Park J, Park S, Choi JY. Confidence-based feature imputation for graphs with partially known features. ArXiv Preprint arXiv:230516618. 2023.
8. Yoo J, Jeon H, Jung J, Kang U. Accurate node feature estimation with structured variational graph autoencoder. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022. pp. 2336–46.
9. Fu S, Peng Q, He Y, Du B, You X. Towards unsupervised graph completion learning on graphs with features and structure missing. 2023 IEEE International Conference on Data Mining (ICDM). IEEE; 2023. pp. 1019–24.
10. Rossi E, Kenlay H, Gorinova MI, Chamberlain BP, Dong X, Bronstein MM. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. Learning on graphs conference. PMLR; 2022. p. 11.

11. Liu X, Zhu X, Li M, Wang L, Zhu E, Liu T, et al. Multiple kernel k k-means with incomplete kernels. IEEE Trans Pattern Anal Mach Intell. 2019;42:1191–204.
12. Batista GE, Monard MC. A study of K-nearest neighbour as an imputation method. His. 2002;87:48.
13. García-Laencina PJ, Sancho-Gómez J-L, Figueiras-Vidal AR. Pattern classification with missing data: a review. Neural Comput Appl. 2010;19:263–82.
14. Chen J, Wang Y, Zeng M, Xiang Z, Hou B, Tong Y, et al. Customizing graph neural networks using path reweighting. Inf Sci (N Y). 2024;674:120681.
15. Bandyopadhyay S, Vivek NL, Murty SV. MN. Outlier resistant unsupervised deep architectures for attributed network embedding. Proceedings of the 13th international conference on web search and data mining. 2020. pp. 25–33.
16. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. ArXiv Preprint arXiv:160902907. 2016.
17. Taguchi H, Liu X, Murata T. Graph convolutional networks for graphs containing missing features. Future Generation Comput Syst. 2021;117:155–68.
18. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. ArXiv Preprint arXiv:171010903. 2017.
19. Tu W, Zhou S, Liu X, Cai Z, Zhao Y, Liu Y et al. WAGE: Weight-Sharing Attribute-Missing graph autoencoder. IEEE Trans Pattern Anal Mach Intell. 2025.
20. Jin D, Huo C, Liang C, Yang L. Heterogeneous graph neural network via attribute completion. Proceedings of the web conference. 2021. 2021. pp. 391–400.
21. Li C, Yan Y, Fu J, Zhao Z, Zeng Q. HetReGAT-FC: heterogeneous residual graph attention network via feature completion. Inf Sci (N Y). 2023;632:424–38.
22. Perozzi B, Al-Rfou R, Skiena S, Deepwalk. Online learning of social representations. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 2014. pp. 701–10.
23. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016. pp. 855–64.
24. Cao S, Lu W, Xu Q, Grarep. Learning graph representations with global structural information. Proceedings of the 24th ACM international on conference on information and knowledge management. 2015. pp. 891–900.
25. Ou M, Cui P, Pei J, Zhang Z, Zhu W. Asymmetric transitivity preserving graph embedding. Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016. pp. 1105–14.
26. Chen H, Perozzi B, Hu Y, Skiena S. Harp: hierarchical representation learning for networks. Proceedings of the AAAI conference on artificial intelligence. 2018.
27. Tsitsulin A, Mottin D, Karras P, Müller E, Verse. Versatile graph embeddings from similarity measures. Proceedings of the 2018 world wide web conference. 2018. pp. 539–48.
28. Rahman MK, Sujon MH, Azad A. Force2Vec: Parallel force-directed graph embedding. 2020 IEEE International Conference on Data Mining (ICDM). IEEE; 2020. pp. 442–51.
29. Wu X, Zhang H, Quan Y, Miao Q, Sun PG. Graph embedding based on motif-aware feature propagation for community detection. Physica A. 2023;630:129205.
30. Huo C, Jin D, Li Y, He D, Yang Y-B, Wu L. T2-gnn: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation. Proceedings of the AAAI Conference on Artificial Intelligence. 2023. pp. 4339–46.
31. Liu Z, Nguyen T-K, Fang Y. Tail-gnn: Tail-node graph neural networks. Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021. pp. 1109–19.
32. Mohammed SN, Gündüç S. Degree-based random walk approach for graph embedding. Turkish J Electr Eng Comput Sci. 2022;30:1868–81.
33. Zhang Z, Sabuncu M. Generalized cross entropy loss for training deep neural networks with noisy labels. Adv Neural Inf Process Syst. 2018;31.
34. McCallum AK, Nigam K, Rennie J, Seymore K. Automating the construction of internet portals with machine learning. Inf Retr Boston. 2000;3:127–63.
35. Giles CL, Bollacker KD, Lawrence S, CiteSeer. An automatic citation indexing system. Proceedings of the third ACM conference on Digital libraries. 1998. pp. 89–98.
36. Namata G, London B, Getoor L, Huang B, Edu U. Query-driven active surveying for collective classification. 10th international workshop on mining and learning with graphs. 2012. p. 1.
37. Donnat C, Zitnik M, Hallac D, Leskovec J. Learning structural node embeddings via diffusion wavelets. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018. pp. 1320–9.
38. Errica F, Podda M, Bacciu D, Micheli A. A fair comparison of graph neural networks for graph classification. ArXiv Preprint arXiv:191209893. 2019.
39. Huang Q, He H, Singh A, Lim S-N, Benson AR. Combining label propagation and simple models out-performs graph neural networks. ArXiv Preprint arXiv:201013993. 2020.
40. Kojaku S, Yoon J, Constantino I, Ahn Y-Y. Residual2Vec: Debiasing graph embedding with random graphs. Adv Neural Inf Processing Syst. 2021;34:24150–63.

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.