University Library



A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Zhao, Y;Qi, J;Liu, Q;Zhang, R

Title:
WGCN: Graph Convolutional Networks with Weighted Structural Features

Date:
2021-07-11

Citation:
Zhao, Y., Qi, J., Liu, Q. & Zhang, R. (2021). WGCN: Graph Convolutional Networks with Weighted Structural Features. SIGIR 2021 Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.624-633. ASSOC COMPUTING MACHINERY. https://doi.org/10.1145/3404835.3462834.

Persistent Link:
https://hdl.handle.net/11343/281919

WGCN: Graph Convolutional Networks with Weighted Structural Features

Yunxiang Zhao[1], Jianzhong Qi[1], Qingwei Liu[1], Rui Zhang[2,*]
The University of Melbourne[1], www.ruizhang.info[2]
{yunxiangz,qingweil}@student.unimelb.edu.au,jianzhong.qi@unimelb.edu.au,rayteam@yeah.net

ABSTRACT

Graph structural information such as topologies or connectivities provides valuable guidance for graph convolutional networks (GCNs) to learn nodes' representations. Existing GCN models that capture nodes' structural information weight in- and out-neighbors equally or differentiate in- and out-neighbors globally without considering nodes' local topologies. We observe that in- and out-neighbors contribute differently for nodes with different local topologies. To explore the directional structural information for different nodes, we propose a GCN model with weighted structural features, named WGCN. WGCN first captures nodes' structural fingerprints via a direction and degree aware Random Walk with Restart algorithm, where the walk is guided by both edge direction and nodes' in- and out-degrees. Then, the interactions between nodes' structural fingerprints are used as the weighted *node structural features*. To further capture nodes' high-order dependencies and graph geometry, WGCN embeds graphs into a latent space to obtain nodes' latent neighbors and geometrical relationships. Based on nodes' geometrical relationships in the latent space, WGCN differentiates latent, in-, and out-neighbors with an attention-based geometrical aggregation. Experiments on transductive node classification tasks show that WGCN outperforms the baseline models consistently by up to 17.07% in terms of accuracy on five benchmark datasets.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**;

KEYWORDS

Directional Graph Convolutional Networks, Structural Information, Random Walk with Restart

ACM Reference Format:
Yunxiang Zhao[1], Jianzhong Qi[1], Qingwei Liu[1], Rui Zhang[2,*]. 2021. WGCN: Graph Convolutional Networks with Weighted Structural Features. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462834

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '21, July 11–15, 2021, Virtual Event, Canada
© 2021 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462834

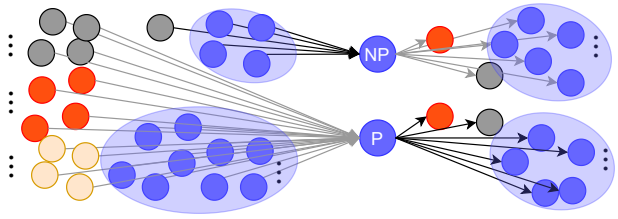


Figure 1: Citation networks for popular (P) and non-popular (NP) papers. Ellipses cover papers with the same class as the target paper. Different node colors denote different classes, such as DM, CV, and NLP (best view in color).

1 INTRODUCTION

Graphs are essential representations of many real-world data such as social networks, transportation networks, and e-commerce user-item graphs [19, 33, 37, 45, 50, 54]. Graph convolutional networks (GCNs), graph attention networks (GATs), and their variants have shown promising results in applications on graph datasets [2, 4–6, 20, 21, 29, 36, 47, 49]. However, most GCN and GAT based models focus on how to aggregate neighboring nodes' content features (e.g., keywords of a document in a document network) and take graph structures (edges/topology) for checking neighboring relationships only. For example, GATs learn the similarities between two nodes' content features as the attention coefficients while overlooking the relationship between nodes' local topologies (e.g., similarities).

Recent studies show that explicitly adding node structural information before the aggregation procedure of GCNs substantially improves models' performance [14, 46]. However, existing approaches treat directed edges as bi-directional, i.e., neighbors connecting to a node have the same weight irrespective of the direction when capturing node structural information [31, 48, 53]. Such approaches ignore directional structures such as irreversible time-series relationships, which may mislead the node structural information learning [40, 44]. A few studies [17, 22, 40, 41] differentiate in- and out-neighbors when capturing node's structural information, but they give the same weight to the in-neighbors of all nodes and another the same weight to the out-neighbors of all nodes.

We believe that *the weight difference between in- and out-neighbors when capturing nodes' structural information is highly related to nodes' local topologies*. Without loss of generality, we illustrate our insight via node classification tasks for directed graphs, as shown in Figure 1. Consider the difference between *popular* and *non-popular* nodes. A node is *popular* if it has much more in-neighbors than out-neighbors, e.g., a paper with many citations but cites few other papers or a celebrity social network user with many followers but follows few other users. A *non-popular* node, on the other hand, has

more out-neighbors than in-neighbors, e.g., a not-so-popular paper or an average social network user. In node classification, neighbors similar to the target node play an essential role, e.g., papers within the same class in citation networks and people with the same profession in social networks. In general, the more neighbors a node has in one direction (in or out), the higher proportion of dissimilar neighbors the node has in that direction. As shown in Figure 1, most citations of a not-so-popular paper (denoted as NP) are from the same class because, typically, only people studying in its area cite the paper. On the other hand, a popular paper (denoted as P) obtains many citations, which may come from other classes instead of the class to which the paper belongs. In summary, compared to the in-neighbors, the out-neighbors of a popular paper have a higher possibility of being similar neighbors. In this case, out-neighbors should carry higher weights when capturing node structural information. For non-popular papers, on the other hand, in-neighbors have a higher possibility of being similar neighbors, compared to the out-neighbors. In this case, in-neighbors should carry higher weights when capturing node structural information. To confirm this intuition, we analyze five real datasets and compute the possibility of a node reaching a neighbor of the same class via in- or out-edges. We find that up to 21% more nodes reach a neighbor of the same class via the direction with a smaller degree than the direction with a larger degree. This observation conforms to our intuition, which will be detailed in Section 3.1.

A straightforward strategy to differentiate the directional structural information for different nodes is to learn a parameter that weights one-hop in- and out-neighbors for each node according to its in- and out-degrees [9]. However, such a strategy cannot capture the nodes' density or differentiate nodes' geometrical relationships, and hence will lose structural information such as communities [48]. Besides, unlike continuous space such as images or videos, GCNs cannot differentiate non-isomorphic graphs due to their permutation-invariant aggregation and cannot capture long-range dependencies [18, 31]. Such information is critical in node representation learning. For example, capturing communities can help learn node structural information because nodes within the same community have closer relationships than those not within the same community [48]; capturing latent neighbors and hierarchical relationships [25, 28] in the latent space can further improve the performance of node representation learning [7, 24].

To address the above limitations, we propose a novel GCN variant named *WGCN*. It models nodes' directional structural information, where the weights of in- and out-neighbors are determined by nodes' local topologies. WGCN contains two components. (i) the first component is a direction and degree aware Random Walk with Restart algorithm (DDRWR), where the walk is guided by both edge direction and node in- and out-degrees. We take the proximity of reaching k-hop neighbors from a given node as the structural fingerprint of the node. The procedure of computing the proximity quantitatively weights in- and out-neighbors for different nodes and captures communities simultaneously. Based on the structural fingerprint of each node, we define the node level interactions between node structural fingerprints to obtain an $\mathcal{N}$-dimensional vector for each node as its structural features ($\mathcal{N}$ is the number of nodes in the graph). (ii) the second component embeds node topology into a latent space, and nodes far away from each other in the graph may become neighbors in the latent space. Moreover, the positions of different nodes in the latent space capture their latent geometrical relationships. Based on nodes' geometrical relationships in the latent space, in message passing, WGCN performs an attention-based geometrical aggregation with learnable parameters for latent, in-, and out-neighbors, respectively. We summarize our contributions as follows:

- We propose a novel GCN model named WGCN to capture nodes' structural information, which differentiates the weights of node in-neighbors from those of node out-neighbors according to nodes' local topologies.
- We propose to embed node topology into a latent space to obtain nodes' high-order dependencies and latent geometrical relationships. We then aggregate latent, in-, and out-neighbors separately with a geometrical attention-based message passing for node representation learning.
- We evaluate the proposed WGCN model with transductive node classification tasks on five public benchmark datasets, and WGCN achieves state-of-the-art performance consistently. We have made the implementation of WGCN public available[1].

2 RELATED WORK

GCNs apply a message-passing strategy where each node aggregates messages/features from its neighboring nodes and updates its feature vector until an equilibrium state is reached [11]. Instead of aggregating all neighboring nodes equally, Velickovic et al. [42] propose a graph attention model (GAT) with a trainable attention function, which learns implicit information to distinguish different neighbors during the aggregation. Based on GAT, various approaches have been proposed for aggregating neighbors' information based on neighbors' content features and structural information [16, 43]. For example, Wang et al. [43] propose to differentiate the weights of different neighbors for heterogeneous graphs and then aggregate content features from meta-path-based neighbors hierarchically. Recently, researchers found that omitting the edge direction information will lead to information loss for node representation learning [41], and explicitly adding rich structural features such as the topology or "shapes" of local edge connections can further boost the performance of GCNs [48]. Therefore, various GCN models have been proposed for directed graphs, and some also explicitly capture directional structural features. They are divided into spectral and spatial approaches.

To capture the edge direction information in the spectral domain. Ma et al. [22] propose a directed Laplacian matrix for directed graphs. However, they only consider the nodes' out-neighbors (out-degree matrix) and overlook the information from in-neighbors, which may cause information loss. To cope with the high computation cost of spectral-GCNs for directed graphs, Li et al. [17] propose a scalable graph convolutional neural network with fast localized convolution operators derived from directed graph Laplacian. To capture both edge direction information and nodes' local structural information, Tong et al. [40, 41] propose to learn first- and second-order proximities, which are combined via a signal fusion function. However, they apply the same weight for in-neighbors of all nodes

[1]The source code is public available at: https://github.com/ruizhang-ai/WGCN_Graph-Convolutional-Networks-with-Weighted-Structural-Features.

and another the same weight for out-neighbors of all nodes when computing the first- and second-order proximity.

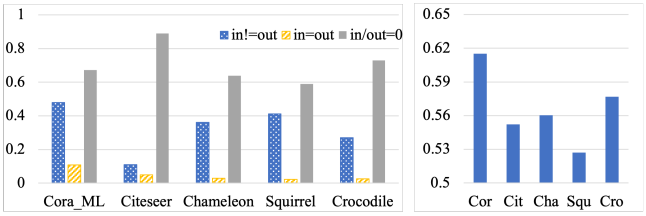
Spatial GCN models mainly focus on undirected graphs when learning nodes' representations and can be applied to directed graphs by following the edge directions during the message passing. For example, Hamilton et al. [6] propose a general inductive framework that can efficiently generate node embeddings for previously unseen data. Zhu et al. [52] propose a neighborhood-aware GCN model, which considers both neighbor-level and relation-level information. Mostafa et al. [23] propose a global attention mechanism where a node can selectively attend to and aggregate content features from any other node in the graph. To capture both edge direction information and nodes' local structural information, Zhang et al. [48] propose to use Random Walk with Restart to obtain node structural information according to node's local topology, and then train a GAT model that considers both the node content features and node structural features. Spatial GCN models can be applied to directed graphs by aggregating in- or out-neighbors only during the message passing. However, considering in- or out-neighbors only may lead to information loss because neighbors in the other direction are overlooked [41].

3 METHODS

We start by analyzing real data to show that in- and out-neighbors have different probabilities of being similar neighbors for nodes with different local topologies (Section 3.1). We then present an overview of our WGCN model that captures such data characteristics in embedding learning (Section 3.2). After that, we discuss how to adapt a classic algorithm – the Random Walk – to capture a neighbor's weight in representing a node's directional local structure information (Section 3.3), detail how to compute a node's structural features based on such an algorithm (Section 3.4), present an attention-based geometrical message passing to support our embedding leaning (Section 3.5), and show the time complexity of our model (Section 3.6).

3.1 Motivation and Insight

Our model is based on the insight that neighbors at the direction with a smaller degree have a higher possibility of being similar neighbors (e.g., being in the same class). We analyze five datasets (details in Section 4.1) and report the percentage of nodes that conform to our insight. In Figure 2a, the nodes in each dataset are divided into three categories: (i) nodes with in- or out-degree being zero (gray bars), i.e., nodes with only in- or out-neighbors. (ii) nodes with the same in- and out-degrees (yellow bars). (iii) nodes with different and non-zero in- and out-degrees (blue bars). Our model benefits the representation learning of the nodes of Category (iii), while it does not has a negative impact on nodes of the other categories. We see that the five datasets have between 52.7% and 61.5% of nodes of Category (iii). Improving the representations learned for such nodes is expected to bring substantial performance gains for downstream applications (e.g., node classifications, detailed in Section 4.2). Within the nodes in Category (iii), we further show the percentage of nodes that conform to our insight in Figure 2b. The y-axis denotes the percentage of nodes where neighbors at the direction with a smaller degree have a higher possibility of sharing



(a) (b)

Figure 2: (a) The percentage of nodes with different and non-zero in- and out-degrees (blue bars), the same in- and out-degrees (yellow bars), and only in- or out-neighbors (gray bars). (b) Among nodes with different and non-zero in- and out-degrees, the percentage of nodes where neighbors at the direction with a smaller degree have a higher possibility of sharing the same class with the nodes.

the same class with these nodes. We observe that over 50% of the nodes in Category (iii) of each dataset tested conform to our insight, confirming the necessity of weighing the in- and out-neighbors differently for the nodes. For the rest nodes in Category (iii), our model has a negative impact on their representation learning, but the overall performance of all nodes is increased since they are the minority. The theoretical analysis in Section 3.3 and the experimental results in Section 4.2 conform to this claim. Next, we present our WGCN model to learn such different weights.

3.2 Model Overview

The overall structure of WGCN is shown in Figure 3, which contains two components. Given a directed graph and node content features (e.g., keywords of a document in a document network, denoted as the vector next to each node), (i) we obtain nodes' structural fingerprints via a direction and degree aware Random Walk with Restart algorithm (DDRWR), where the walk is guided by edge direction and node in- and out-degrees. The interactions between nodes' structural fingerprints are taken as the nodes' structural features. We then concatenate structural features with node content features as the input to be passed to the message passing phase. (ii) meanwhile, we embed nodes into a latent space via existing embedding approaches that capture nodes' latent neighbors and the graph topology, i.e., the nodes' geometrical relationships. For each node, we differentiate its neighbors into different classes (folds) according to neighbors' geometrical relationships in the latent space. During the message passing, we first aggregate (e.g., sum or mean) latent, in- and out-neighbors separately in each sub-space to obtain virtual nodes denoted as ⓥ in Figure 3, and an attention mechanism is applied to enable nodes to discriminately attend to neighbors with different weights. We then update node representation by aggregating (e.g., concatenate) those virtual nodes to obtain the final representation of each node. The "concatenate" aggregation can retain the order of different virtual nodes and hence can differentiate neighbors with different geometrical relationships.

3.3 Neighbor Weight Modeling

In GCN models, Random Walk has been widely used to capture structural information in node representation learning [3, 13, 48],

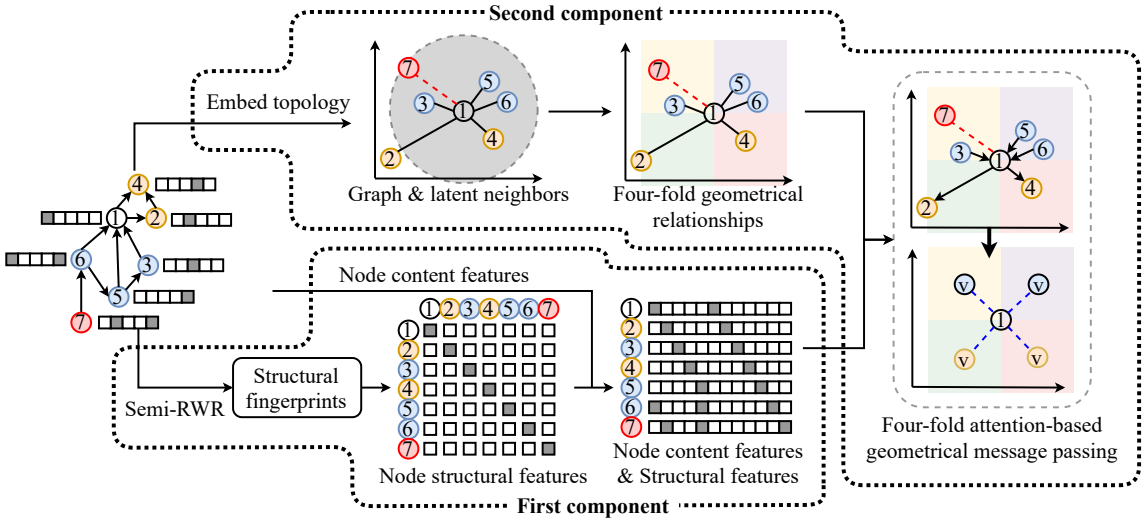


Figure 3: Overall structure of WGCN. We take 1-hop in-neighbors (blue) and out-neighbors (yellow) of node ① in a four-fold attention-based geometrical message passing as an example. Nodes with high-order dependencies to node ① in the latent space are red colored such as node ⑦ (best view in color).

where the possibility of reaching each neighbor during the exploring process denotes the weight of the neighbor in representing a node's structural information. In this subsection, we first show that traditional Random Walk algorithms do not serve our purpose on directed graphs. We then present our adapted Markov process and show its advantages.

Random Walk algorithms can be seen as time-homogeneous Markov processes. Let $\{X^{[n]},\ n \geq 0\}$ be a time-homogeneous Markov process with state/node space V and transition matrix $P = (p_{ij})_{i,j\in V}$. For any nodes $i, j \in V$:

$$p_{ij} := \mathbb{P}(X^{[n+1]} = j \mid X^{[n]} = i),\ \ \forall\, n \geq 0 \tag{1}$$

where $X^{[n+1]}$ denotes the next state generated from the current state $X^{[n]}$. Random Walk with Restart (RWR) further captures the intuitively decaying weight of neighbors of different hops, while maintaining the community information [48]. The RWR procedure forces the walk on a graph to always restart from the same node (or group of nodes). It can quantify the structural proximity between a given node and all other nodes in the graph [39, 51], which has been widely used for graph embedding [3, 12, 13, 30, 32].

Without loss of generality, we take graphs without self-loop and mutual connections as an example for illustration. For each node $i \in V$, we denote $A_i \subseteq V$ as the "neighbors" of node i, A_i^I and A_i^O as the in-neighbors and out-neighbors of node i, respectively, and $A_i = A_i^I \cup A_i^O$. Let d_i be the degree of node i (here and below degree means in-degree plus out-degree), i.e., $d_i := d_i^I + d_i^O$, where $d_i^I := |A_i^I|$ and $d_i^O := |A_i^O|$ are the in-degree and out-degree of node i, respectively. In the traditional RWR, the possibility of any node $i \in V$ reaching next node j during the exploring procedure is:

$$p_{ij} := \begin{cases} c, & \text{if } j = o \\ \frac{1-c}{d_i}, & \text{if } j \in A_i \\ 0, & \text{else} \end{cases} \tag{2}$$

where node o is the starting node, i.e., $\mathbb{P}(X^{[0]} = o) = 1$, and $c \in (0, 1)$ is a fixed constant, which indicates the possibility of restart.

Existing studies on RWR do not consider the asymmetric roles of vertices and are not robust for directed graphs [9]. Based on our insight in Section 3.1, for a node i, if there is a difference between its in-degree and out-degree, its neighbors in the direction with the smaller degree are more likely to be similar to node i. More precisely, given node i, if $|d_i^I - d_i^O|$ exceeds a threshold $thr \in \mathbb{N}$, for any node $j \in A_i^I$, and node $l \in A_i^O$, we have:

$$\begin{cases} \mathbb{P}(j \in \mathcal{I}_i) > \mathbb{P}(l \in \mathcal{I}_i), & \text{if } d_i^O - d_i^I > thr \\ \mathbb{P}(j \in \mathcal{I}_i) < \mathbb{P}(l \in \mathcal{I}_i), & \text{if } d_i^I - d_i^O > thr \end{cases} \tag{3}$$

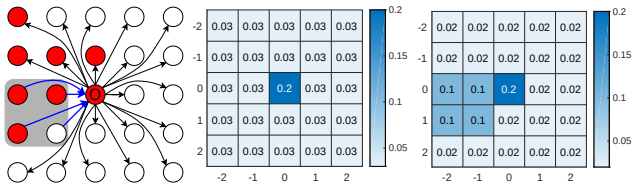
where $\mathcal{I}_i$ denotes all neighbors similar to node i, which is a random subset of V. In practice, we set the threshold thr as 0.

Based on this insight, we propose a Markov process $\{\tilde{X}^{[n]} : n \geq 0\}$ with transition matrix $\tilde{P} = (\tilde{p}_{ij})_{i,j\in V}$, where $\tilde{p}_{ij}$ for each node $i \in V$ is computed as follows:

$$\tilde{p}_{ij} = \begin{cases} c, & \text{if } j = o \\ w_i^I, & \text{if } j \in A_i^I \\ w_i^O, & \text{if } j \in A_i^O \\ 0, & \text{else} \end{cases}, \quad \begin{cases} w_i^I > w_i^O, & \text{if } d_i^O - d_i^I > thr \\ w_i^I < w_i^O, & \text{if } d_i^I - d_i^O > thr \\ w_i^I = w_i^O, & \text{else} \end{cases} \tag{4}$$

Here, $c + d_i^I w_i^I + d_i^O w_i^O = 1$. For each node i, the possibility of the Markov process reaching the set of similar neighbors $\mathcal{I}_i$ is $\sum_{n\geq 1} \tilde{\mathbb{P}}(\tilde{X}^{[n]} \in \mathcal{I}_i)$. Throughout this paper, we only consider such a set of neighbors w.r.t. the starting node o. Thus, for simplification, we use $\mathcal{I}$ instead of $\mathcal{I}_o$. Assume that the distribution of $\mathcal{I}$ is independent of the distribution of $\{\tilde{X}^{[n]} : n \geq 0\}$. Then the possibility of reaching set $\mathcal{I}$ in the typical RWR and the proposed Markov process in the 1-hop neighbors are:

$$\mathbb{P}(X^{[n+1]} \in \mathcal{I}) = \sum_{i\in V} \sum_{j\in A_i} \mathbb{P}(j \in \mathcal{I}) \frac{1-c}{d_i} \mathbb{P}(X^{[n]} = i) + c$$



(a) A simple graph. (b) RWR. (c) Ours.

Figure 4: Neighbor weighing via typical RWR and the proposed Markov process for node structural information learning on a simple graph.

$$\tilde{\mathbb{P}}(\tilde{X}^{[n+1]} \in \mathcal{I}) = \sum_{i \in S_1} \left\{ \sum_{j \in A_i^I} \mathbb{P}(j \in \mathcal{I}) w_i^I + \sum_{j \in A_i^O} \mathbb{P}(j \in \mathcal{I}) w_i^O \right\} \tilde{\mathbb{P}}(\tilde{X}^{[n]} = i)$$
$$+ \sum_{i \in S_2} \left\{ \sum_{j \in A_i^I} \mathbb{P}(j \in \mathcal{I}) w_i^I + \sum_{j \in A_i^O} \mathbb{P}(j \in \mathcal{I}) w_i^O \right\} \tilde{\mathbb{P}}(\tilde{X}^{[n]} = i)$$
$$+ \sum_{i \in S_3} \sum_{j \in A_i} \mathbb{P}(j \in \mathcal{I}) \frac{1-c}{d_i} \tilde{\mathbb{P}}(\tilde{X}^{[n]} = i) + c$$

where $n \geq 0$, S_1, S_2, and S_3 are defined as: $S_1 := \{i \in V : d_i^I - d_i^O > thr\}$, $S_2 := \{i \in V : d_i^O - d_i^I > thr\}$, $S_3 := \{i \in V : |d_i^O - d_i^I| \leq thr\}$. When $n = 0$, it is clear that $\mathbb{P}(X^{[1]} \in \mathcal{I}) \leq \tilde{\mathbb{P}}(\tilde{X}^{[1]} \in \mathcal{I})$. More precisely, given the initial distribution $\mathbb{P}(X^{[0]} = o) = \tilde{\mathbb{P}}(\tilde{X}^{[0]} = o) = 1$, node o must fall in one of S_i, since $\{S_1, S_2, S_3\}$ is a partition of V. If $o \in S_1$, we have $w_o^I < \frac{1-c}{d_o} < w_o^O$ and $\mathbb{P}(j \in \mathcal{I}) < \mathbb{P}(l \in \mathcal{I})$ for any $j \in A_o^I$, $l \in A_o^O$ from Equations 3 and 4. Therefore:

$$\sum_{j \in A_o^I} \mathbb{P}(j \in \mathcal{I}) w_o^I + \sum_{j \in A_o^O} \mathbb{P}(j \in \mathcal{I}) w_o^O > \sum_{j \in A_o} \mathbb{P}(j \in \mathcal{I}) \frac{1-c}{d_o} \quad (5)$$

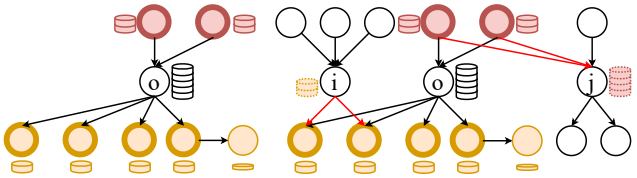
Similarly, the inequality (5) holds for $o \in S_2$. Because our model has the same possibility of reaching $\mathcal{I}$ as typical RWR for $o \in S_3$, then:

$$\mathbb{P}(X^{[1]} \in \mathcal{I}) \leq \tilde{\mathbb{P}}(\tilde{X}^{[1]} \in \mathcal{I}) \quad (6)$$

Compared with the neighbors of a dissimilar neighbor, those of a similar neighbor have a higher possibility to be similar high-order neighbors. Therefore, with the increased possibility of reaching $\mathcal{I}$ for node o, the possibility of reaching $\mathcal{I}$ in high-order neighbors will also become higher in our model.

Figure 4 shows an example of using typical RWR and our Markov process to capture the node structural information. In Figure 4a, circles denote nodes, and directed edges between nodes are the topology. Node o has 4 in-neighbors and 20 out-neighbors in the graph. According to our insight, the in-neighbors should have a higher possibility to be similar neighbors. In this graph, three out of four (i.e., a probability of 75%) in-neighbors are in the same class as node o. On the other hand, the out-neighbors of node o have a smaller possibility to contain similar neighbors of node o. In Figure 4a, there are four out of 20 (i.e., only a probability of 20%) out-neighbors in the same class as node o.

If we run the typical RWR algorithm with a restart rate of 0.2 to capture the node structural information of node o, the weight distribution will be like Figure 4b, where all neighbors connecting to node o have the same weight. On the other hand, if we take that "in-neighbors have a higher possibility to contain similar neighbors" as pre-knowledge, and assign the in-neighbors with a higher weight, then the weight distribution will be like Figure 4c, where the in-neighbors have a higher weight. Note that the similar neighbors of node o in the out-direction have lower weights, and the dissimilar in-neighbors of node o have higher weights in our method. However, the overall possibility of reaching similar nodes has been increased.



(a) Neighbor weighting. (b) Structural interactions.

Figure 5: Neighbor weighting and node-level structural interactions in WGCN. The in-neighbors of node o have higher weights due to the smaller in-degree compared to the out-degree. Node j has a higher interaction weight with node o because their interacted nodes have higher weights.

3.4 Directional Node Structural Features

In GCN models, node structural features can be represented as the similarities between the local topologies of pairwise nodes. To embed the directional information into node structural features, we first present our DDRWR module for learning nodes' local topologies (fingerprints), which is based on the proposed Markov process in Section 3.3. We then present the method for computing the node structural features.

When computing the structural fingerprint for each node, it is desirable to assign weights to neighbors adaptively based on nodes' local structures (e.g., communities). Figure 5a shows a spanning process on node o's k-hop (k=2 in this example) local subgraph G_o, and each neighbor in G_o contributes a weight, which denotes the importance of the neighbor when capturing the structural fingerprint of node o. Based on discussions above, in-neighbors A_o^I are more likely to be similar neighbors because the in-degree d_o^I of node o is smaller than its out-degree d_o^O, and hence they should have higher weights when capturing the node structural fingerprint. Besides, even though neighbors far away from node o should have smaller weights, mapping the node distance levels [1, 2, ..., k] to non-negative, monotonically decreasing weight levels $w_o = [w_1, w_2, ..., w_k]$ will overlook the importance of communities in evaluating node similarities.

To achieve the above objectives, we combine RWR with our proposed Markov process in Section 3.3, which yields an algorithm named *DDRWR*. DDRWR adjusts the walk according to the edge direction and the in/out-degrees of different nodes. Without abuse of notations, we define the transition matrix $\tilde{P} = (\tilde{p}_{ij})_{i,j \in V}$ as:

$$\tilde{p}_{ij} = 1 + b \cdot M_{ij} \left(\frac{d_i^I - d_i^O}{d_i^I + d_i^O} \right)^{\epsilon} \quad (7)$$

where b controls the maximum weight difference between in- and out-neighbors, M_{ij} indicates the relationship between nodes i and

node j in the graph. M_{ij} equals -1 if node i reaches node j via the in-edges of node i following the BFS, M_{ij} equals 1 if node i reaches node j via the out-edges of node i following the breadth first search (BFS), and M_{ij} is 0 if node j is out of reach from node i. Recall that d_i^I and d_i^O are the in- and out-degrees of node i, respectively. ϵ is zero or a positive odd number that controls the in- and out-neighbor weight variation trend regarding the in- and out-degrees ratio. For example, $\epsilon = 0$ denotes that in- and out-neighbors have the same weight, $\epsilon = 1$ denotes that the weight of in- and out-neighbors has a linear relationship to the in- and out-degrees' difference, and $\epsilon = 3$ denotes a cubic relationship. Accordingly, we formulate the iterations of DDRWR taking node o as the origin as:

$$\mathbf{w}_o{}^{[n+1]} = (1-c) \cdot \tilde{P}' \mathbf{w}_o{}^{[n]} + c \cdot \mathbf{e}_o \tag{8}$$

where $c \in [0, 1]$ determines the ratio of the restart, $\mathbf{e}_o$ is a vector with all components equal to 0, except the entry corresponding to node o, which is 1, and $\tilde{P}'$ is the normalized version of the transition matrix $\tilde{P}$ restricted on G_o.

In general, if d_i^I and d_i^O are close to each other, $\tilde{p}_{ij}$ approaches to the p_{ij} in typical RWR, which means that in- and out-neighbors obtain similar weights and closer neighbors obtain a higher weight. If d_i^I is larger than d_i^O, then M_{ij} is negative when node i reaches node j via its in-edges. Therefore, the weight $\tilde{p}_{ij}$ is larger for nodes reached via out-edges and smaller for nodes reached via in-edges. On the contrary, if d_i^I is smaller than d_i^O, weight $\tilde{p}_{ij}$ is smaller for nodes reached via out-edges and larger for nodes reached via in-edges. For mutually connected neighbors of node i, we set their probabilities as the sum of being both in- and out-neighbor. The converged solution of Equation 8 is:

$$\mathbf{w}_o = (I - (1-c) \cdot \tilde{P}')^{-1} \mathbf{e}_o \tag{9}$$

where $\mathbf{w}_o$ quantifies the proximity of node o reaching its k-hop neighbors G_o, and we take $\mathbf{w}_o$ as the structural fingerprint of node o, c controls the decay rate of the fingerprint. $\mathbf{w}_o$ is zeros except for position o if c is 1, and $\mathbf{w}_o$ has the same distribution as that produced by a Random Walk without restart if c is 0.

After computing the structural fingerprint $\mathbf{w}_o$ of each node o, we only keep those values of neighbors within G_o (neighbors within k-hops) and normalize $\mathbf{w}_o$ to compute the node level structural interactions. We use weighted Jaccard similarity to evaluate the structural interactions $S = (s_{ij})_{i,j \in V}$ as follows:

$$s_{ij} = \frac{\sum_{g \in (G_i \cup G_j)} min(w_{ig}, w_{jg})}{\sum_{g \in (G_i \cup G_j)} max(w_{ig}, w_{jg})} \tag{10}$$

where w_{ig} and w_{jg} are the gth values of the normalized $\mathbf{w}_i$ and $\mathbf{w}_j$, respectively. Each s_{ij} in S is a value denoting the structural interaction between node i and j. Figure 5b illustrates Equation 10. For node o, out-neighbors have a higher weight because the out-degree is smaller than the in-degree. Nodes i and j each has two intersected neighbors with node o, but node o has a higher structural interaction with node j because of the higher weights of their intersected neighbors.

3.5 Geometrical Message Passing

To explore node latent geometrical relationships, we embed each node i in the graph by $f : i \rightarrow \mathbf{z}_i$, where $\mathbf{z}_i$ denotes the position of node i in the latent space, function f is an embedding function that

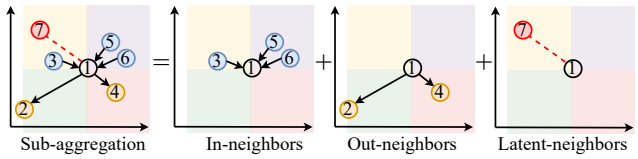


Figure 6: An example of four-fold sub-aggregation.

preserves node latent geometrical relationships. We apply Isomap embedding later in the experiments, which can be replaced by other approaches that preserve node latent geometrical relationships. In the latent space, complex topology patterns can be preserved and presented as intuitive geometry, such as subgraph, community, and hierarchy [25–27, 31]. We apply a distance threshold to determine the latent neighbors of each node, as the dashed circle shown in Figure 3. The way to obtain the threshold is the same as that in Geom-GCN [31]. To capture nodes' geometric relationships, for node i's neighbors N_i in the latent space ($G_i \subseteq N_i$ because N_i may contain latent neighbors), WGCN computes their relative positions:

$$\tau(\mathbf{z}_i, \mathbf{z}_j) \rightarrow r \in R, \quad j \in N_i \tag{11}$$

where τ is a function defined in the latent space, and R is a set of relationships. For example, in 2D embedding space, a four-fold relationship set R includes 13 relationships as shown in Figure 6: upper-left, upper-right, lower-left, lower-right for latent, in- and out-neighbors, respectively, and one last relationship is self-loop. Given the positions of two nodes in 2D embedding space, the function τ in Equation 11 will return one of the 13 relationships.

After obtaining the geometrical relationships for all neighbors N_i of a given node i, WGCN either performs attention-based or pure sub-aggregation. For attention-based sub-aggregation, WGCN first concatenates node structural features to content features as the new features of each node i via $\mathbf{h}_i = (\tilde{\mathbf{x}}_i || \tilde{\mathbf{s}}_i) \in \mathbb{R}^d$, where $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{s}}$ are two vectors denoting the row-normalized (described in GCN) content features $\mathbf{x}_i$ and structural features $\mathbf{s}_i$ of node i, $||$ denotes the concatenate operator. For neighbors with each type of relationship r, WGCN learns a projection matrix $W_r \in \mathbb{R}^{d \times d}$ parameterizing a particular attention function $\mathbf{A} \in \mathcal{A}$ to enable the sub-aggregation to discriminate the neighbors with relationship r to node i. The unnormalized attention coefficient between node i and its neighbors in one of the sub-aggregation spaces r is:

$$e_{ij} = \mathbf{A}(W_r \mathbf{h}_i, W_r \mathbf{h}_j), \; j \in N_{i,r} \tag{12}$$

where e_{ij} denotes the importance of node j to node i, $N_{i,r}$ denotes the set of neighbors with relationship r to node i only, i.e.,

$$N_{r,i} := \{j \in N_i : \; j \xrightarrow{r} i\}.$$

We then apply a Softmax function to e_{ij}, and the attention coefficient between each pair of nodes i and its neighbor j is:

$$\alpha_{ij} = \frac{exp(LeakyRelu(e_{ij}))}{\sum_{g \in N_{i,r}} exp(LeakyRelu(e_{ig}))} \tag{13}$$

where α_{ij} denotes the coefficient between node i and node j. With the attention coefficients α, we summarize the sub-aggregation of node i under relationship r in WGCN as:

$$\mathbf{v}_{i,r} = ||_{\mathbf{A} \in \mathcal{A}} \sum_{g \in N_{i,r}} \alpha_{ig} W_r \mathbf{h}_g \tag{14}$$

where $\mathbf{h}_g$ denotes the features of node g. The sum operator can be replaced by other permutation-invariant operators such as the mean operator. The attention function $\mathbf{A}$ (with α_{ig} and W_r) is learned for each relationship r. $||_{\mathbf{A}\in\mathcal{A}}$ concatenates the output from all attention heads $\mathcal{A}$ to obtain the features of virtual node $\mathbf{v}_{i,r}$. The corresponding equation to obtain features of virtual node $\mathbf{v}_{i,r}$ in pure sub-aggregation can be easily inferred from Equation 14. Based on the virtual nodes from neighbors with different relationships, WGCN aggregates the features of all virtual nodes as follows:

$$\mathbf{h}'_i = \sigma(\hat{W}Q([\mathbf{v}_{i,r_1}, ...\mathbf{v}_{i,r_n}])) \tag{15}$$

where $\hat{W}$ denotes a learnable weight matrix for the overall aggregation, σ denotes a non-linear activation function, function Q is a permutation-variant function (e.g., concatenate) that summarizes the features from all virtual nodes, so as to explore the neighbors' geometry relationships. Equations 14 and 15 work together as one message passing layer in WGCN, and multi-layer message passing can be implemented by aggregating Equations 14 and 15 iteratively.

3.6 Complexity Analysis

We analyze the complexity of WGCN as follows. To compute the structural fingerprint of each node, we reach k-hop neighbors G_i via BFS, where k is two for assortative datasets and one for disassortative datasets in our experiments. The complexity is $O(\mathcal{N} \cdot \mathcal{K})$, where $\mathcal{N}$ denotes the number of nodes in the graph, and $\mathcal{K}$ denotes the average number of k-hop neighbors. For each node, we compute the interactions with its $2k$-hop neighbors via Jaccard similarity, and the complexity is $O(\mathcal{K})$. Therefore, the overall complexity for learning node structural features is still $O(\mathcal{N} \cdot \mathcal{K})$. The computation of node structural features is a one-off pre-processing step before the training process. During the message passing, WGCN combines node structural features with node content features and then performs an attention-based geometrical aggregation. Take WGCN without attention mechanism as an example (adding attention mechanism does not change the time complexity [42]), the time complexity is $O((\mathcal{N}+\mathcal{F}) \cdot \mathcal{E})$, where $\mathcal{F}$ denotes the dimensionality of the original node content features, and $\mathcal{E}$ denotes the number of edges in the graph. An important future work is to develop accelerating technology for improving the scalability of WGCN.

4 EXPERIMENTS

We compare WGCN with state-of-the-art models on graph datasets for transductive node classification tasks.

4.1 Datasets and Experimental Setup

Datasets: We use both assortative and disassortative graph datasets. Assortative graphs [18] refer to those with high node homophily, and we use two directed citation graphs, Cora-ML [1] and Citeseer [1]. The graph nodes represent articles, while the edges represent citations between articles. Both datasets also include bag-of-words feature vectors for each article. Disassortative graphs [18] contain more nodes that share the same class labels but are distant from each other. We use three directed Wikipedia page link graphs Chameleon, Squirrel, and Crocodile. These are page-to-page link networks on three different topics. In these datasets, the nodes represent web pages, and the edges are directed links from one page to another. The nodes' content features correspond to informative nouns in the Wikipedia pages. The nodes are evenly divided into five classes according to their average monthly traffic (number of visits) [35]. Table 1 summarizes these datasets.

Table 1: Datasets statistics.

Dataset	Cora-ML	Citeseer	Chameleon	Squirrel	Crocodile
Nodes	2,995	4,320	2,277	5,201	11,631
Edges	8,416	5,358	36,101	217,073	180,021
features	2,879	602	2,325	2,089	13,183
Classes	7	6	5	5	5
Assortative	Yes	Yes	No	No	No

Experimental setup: We implement a two-layer WGCN. For each layer, we first apply a four-fold geometrical aggregation to obtain virtual nodes that capture the latent geometry relationships. To capture node latent geometrical relationships, we apply the Isomap embedding [38], where the distance patterns (lengths of shortest paths) and geometrical information are preserved in the latent space. We then concatenate the output of different virtual nodes as the output. During the structural fingerprint generation, we set the hops of neighbors to consider as two during the DDRWR exploring for assortative graphs and one for disassortative graphs, and ϵ as three. Parameter b and c are dataset and embedding algorithm related. We use the grid search to find optimal values for them in the range of [0,1] with a step length of 0.1. In general, setting $b \in [0.2, 0.4]$ and $c \in [0.4, 0.6]$ yields a satisfactory performance. We set the parameter b as 0.3 and c as 0.5 for all datasets. We use Adam optimizer [10], and ReLU as the activation function for the overall aggregation. We run our model for 500 epochs, use a dropout ratio of 0.5, a learning rate of 0.05, and a weight decay of 5e-6. We set the number of attention head as one, and the number of hidden units as 48 for both layers.

Baselines: We compare our model with nine state-of-the-art models divided into three groups: 1) spectral-based GNNs including GCN [11], DGCN [53], and H-GCN [8]; 2) spatial-based GNNs containing GAT [42], ADSF [48], Geom-GCN [31] (Geom-GCN$_{\text{I}}$, Geom-GCN$_{\text{S}}$, and Geom-GCN$_{\text{P}}$ are its variants with different embedding methods), and PH-GCN [23] (PH-GCN_ea replaces the original GAT attention mechanism in PH-GCN by Euclidean attention mechanism); 3) DiGCN [40], which is the SOTA GNN model for directed graphs. The source codes of other GNN models for directed graphs [17, 22, 41] are not publicly available. For GCN and GAT, we follow their implementations in the Geom-GCN paper. For all other baselines, we set their parameters the same as those in the original paper. We randomly split the datasets and perform 10 experiments for each model to obtain more reliable results. For all baselines and our WGCN model, we randomly split nodes of each class into 60%, 20%, and 20% for training, validation, and testing, respectively. This data splitting has been used in existing works such as Geom-GCN and PH-GCN. We run experiments with NVIDIA Tesla P100 [15].

4.2 Results

The experimental results are summarized in Table 2, where the reported values denote the mean classification accuracy in percentage. WGCN achieves state-of-the-art performance on both assortative and disassortative datasets. Specifically, (i) on assortative datasets,

Table 2: Mean Classification Accuracy (Percent) and Training Time for 100 Epochs (Second).

Models	Assortative				Disassortative					
	Cora-ML		Citeseer		Chameleon		Squirrel		Crocodile	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
GCN [11]	76.23±1.3	4.69	84.40±0.9	4.00	32.21±2.1	13.51	25.69±1.6	46.27	55.87±0.8	41.15
DGCN [53]	30.07±5.4	15.80	95.22±0.8	5.60	61.84±1.4	11.26	20.27±0.7	40.65	19.82±0.9	135.99
H-GCN [8]	72.60±4.4	9.97	91.59±1.2	8.32	42.32±0.1	8.12	23.54±0.1	38.47	44.01±0.1	46.52
PH-GCN [23]	59.25±3.4	2.56	80.47±2.4	2.05	30.70±3.6	3.94	21.13±0.9	17.70	55.09±0.9	42.96
PH-GCN_ea [23]	64.95±2.5	51.70	83.65±2.1	87.17	35.53±3.4	44.42	24.11±1.1	58.06	57.41±1.1	115.68
DiGCN [40]	81.22±0.5	13.14	85.14±0.5	12.04	51.92±1.5	14.71	34.32±1.1	38.23	65.47±0.3	75.77
DiGCN_ib [40]	83.55±0.2	22.28	89.33±0.5	21.60	50.59±1.9	25.86	33.67±0.6	92.57	63.91±0.3	170.59
GAT [42]	69.87±2.2	30.25	79.90±1.3	25.03	38.09±1.5	96.35	27.89±1.4	358.00	56.67±1.3	315.84
ADSF-RWR [48]	70.92±2.6	30.02	80.06±1.2	25.11	40.65±1.8	95.71	29.01±1.2	357.53	57.37±1.3	304.64
Geom-GCN$_I$ [31]	80.94±1.4	28.23	92.77±0.8	22.08	60.41±3.0	133.79	32.23±2.2	793.29	62.37±0.8	441.84
Geom-GCN$_S$ [31]	78.59±1.2	37.84	91.50±0.8	31.37	59.54±3.1	138.17	34.45±1.3	482.75	57.63±0.4	427.52
Geom-GCN$_P$ [31]	82.06±1.3	31.30	94.96±0.7	27.27	53.46±2.0	202.74	35.98±1.3	428.69	47.47±1.1	419.32
WGCN	**87.31±2.1**	27.42	**96.45±0.8**	22.45	**67.25±2.9**	74.97	**53.05±3.8**	425.56	**75.57±0.6**	580.13
WGCN$_S$	86.83±2.0	29.88	95.75±1.0	22.25	66.38±7.9	76.40	50.75±2.6	419.82	75.49±0.5	578.45
WGCN$_P$	86.99±1.2	27.04	95.84±1.1	22.50	62.59±3.3	74.94	45.81±2.2	425.96	73.12±0.9	594.12

WGCN achieves state-of-the-art performance on the Cora-ML and Citeseer datasets with 3.76% and 1.22% accuracy improvements, respectively; (ii) on disassortative datasets, WGCN outperforms baselines by up to 5.41% on the Chameleon dataset, 17.07% on the Squirrel dataset, and 10.10% on the Crocodile dataset.

To analyze the importance of the embedding methods for capturing nodes' latent neighbors, we implement two more variants of WGCN with Struc2vec [34] (WGCN$_S$) and Poincare [27] (WGCN$_P$) embedding methods, respectively. Struc2vec embedding that capturing node structure (k-hop neighbors' degree) similarities has a competitive performance on all datasets except the Squirrel dataset, where the accuracy is 2.3% lower than that of WGCN with Isomap embedding [38]. Its inferior performance may be due to the lack of capturing nodes' geometrical relationships, which does not suit our WGCN model. The performance of Poincare embedding achieves competitive performance on citation datasets. However, it achieves less satisfactory results on Wikipedia page link datasets with up to 7.24% drops in the accuracy. This is because Poincare embedding assumes that nodes have latent hierarchical relationships (e.g., mammal and rodent), which may not hold on the Wikipedia page link datasets on a specific topic. Therefore, we recommend using Isomap embedding (or other embedding methods) that can capture the nodes' geometrical relationships for our WGCN model.

4.3 Efficiency

To evaluate the time costs of WGCN, we run each model with 100 epochs on different datasets and summarize the running times in Figure 7. On assortative datasets, the running time of WGCN is on par with 5 out of 8 baselines, except for the GCN, DGCN, and H-GCN models. For disassortative graphs, WGCN still achieves comparable running times with DiGCN_ib, GAT, ADSF, and Geom-GCN. Considering the performance gain of WGCN, we argue that its extra time costs are worthwhile.

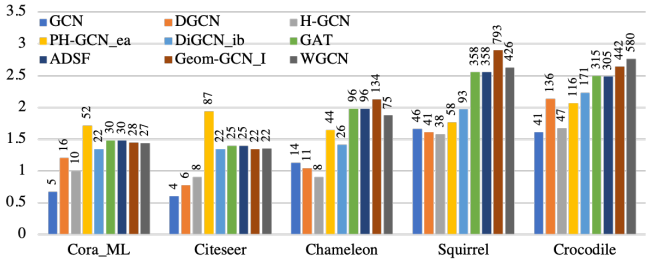


Figure 7: The time costs of different models on different datasets in 100 epochs. The y axis is the running time in the Log scale, and the labels are the running times in seconds.

4.4 Ablation Study

WGCN contains two main components. The first component DDRWR applies a direction and degree aware RWR to capture the rich directional structural information. To study its impact, we implement a variant of WGCN named WGCN/E that takes all neighbors with the same weights when capturing the node structural information. In other words, we replace our proposed DDRWR with typical RWR to capture the node structural information. We skip variants using vanilla Random Walk [32] as RWR has been shown to capture node structural features better [48]. The second component of WGCN captures nodes' geometrical relationships and latent neighbors for node structural information learning. To study its impact, we implement two variants of WGCN named WGCN/G and WGCN/L. WGCN/G does not differentiate the geometrical relationships of different neighbors. In WGCN/L, the latent neighbors are removed during the geometrical aggregation (e.g., node ⑦ in Figure 3).

The results of different variants are summarized in Table 3. In general, WGCN and WGCN/G achieve similar accuracy with at most 0.18% difference. Meanwhile, WGCN achieves lower variances on the disassortative datasets, and WGCN/G achieves lower variances

Table 3: Mean Classification Accuracy (Percent).

Models	Assortative		Disassortative	
	Cora-ML	Citeseer	Chameleon	Squirrel
WGCN	87.31±2.09	**96.45±0.8**	67.25±2.98	**53.05±3.83**
WGCN/G	**87.49±0.86**	96.41±0.3	**67.43±5.42**	52.95±4.26
WGCN/L	87.01±1.56	96.41±0.6	67.19±3.70	52.63±1.28
WGCN/E	86.78±0.92	96.22±0.8	66.71±5.48	51.97±1.76

on the assortative datasets. Therefore, we recommend using WGCN for disassortative datasets and WGCN/G for assortative datasets. The geometrical relationships among assortative datasets are less important than that of disassortative datasets. For WGCN/E and WGCN/L, we see that they are consistently worse than or equal to WGCN and WGCN/G. This confirms that differentiating the in- and out-neighbors and capturing the long-range dependencies are important when capturing node structural information.

4.5 Parameter Study

We examine the hyper-parameters ϵ, the number of hops of neighbors k during the DDRWR process, and the number of network layers of WGCN. We set the default values of ϵ as 3, k as 2 for assortative datasets and 1 for disassortative datasets, and the number of network layers as two.

Impact of ϵ: Parameter ϵ determines the weight variation trend of in- and out-neighbors under a given ratio between the in- and out-degrees. A smaller value of ϵ denotes that the weight is more sensitive to the ratio (e.g., linear relationship to the ratio when ϵ is 1). A larger value of ϵ denotes that the weight is less sensitive to the ratio but changes sharply if neighbors are mostly in- or out-neighbors (e.g., the variation trend of quintic function). We report the performance of WGCN with $k \in [1, 9]$ and a step length of 2. The results in Table 4 show that WGCN performs the best when the value of ϵ is 3 for all datasets. When $\epsilon > 3$, the model's performance decreases on all datasets slightly. Therefore, we take $\epsilon = 3$ as a choice for yielding a good performance.

Table 4: Accuracy with different ϵ.

ϵ	Assortative		Disassortative	
	Cora-ML	Citeseer	Chameleon	Squirrel
1	86.49±1.63	96.22±0.33	67.14±5.20	51.89±2.26
3	**87.31±2.09**	**96.45±0.80**	**67.25±2.98**	**53.05±3.83**
5	85.60±4.29	96.18±0.40	66.78±2.78	52.21±1.25
7	86.41±1.13	96.16±0.49	66.03±4.09	52.93±1.43
9	86.46±3.05	96.16±0.45	66.97±4.49	51.49±2.23

Impact of k: The neighbors' range determines the neighbors that WGCN considers when generating node structural fingerprint. We report the performance of WGCN with k-hop neighbors, where $k \in [1, 3]$. The results in Table 5 show that WGCN achieves the highest accuracy using 2-hop local neighbors for assortative graphs and 1-hop local neighbors only for disassortative graphs. This possibly is because, in assortative graphs, similar nodes tend to be close to each other, where 2-hop neighbors still maintain a strong relationship with the target node, while this relationship is weaker on disassortative graphs. Notice that the k determines the hops of graph neighbors to consider during the message passing, our model will also involve neighbors that close in the latent space but beyond k-hop in the graph topology.

Table 5: Accuracy with different k.

k-hops	Assortative		Disassortative	
	Cora-ML	Citeseer	Chameleon	Squirrel
1	85.85±1.96	95.90±0.78	**67.25±2.98**	**53.05±3.83**
2	**87.31±2.09**	**96.45±0.80**	66.07±3.47	48.99±1.56
3	86.03±0.63	95.86±0.55	65.15±1.89	47.98±5.44

Impact of the number of network layers: As shown in Section 3.5, our model first aggregates the nodes with different relationships to the target node into different virtual nodes and then aggregates all the virtual nodes to obtain the final representation of the target nodes. We add multiple layers (each layer contains a local and then a global aggregation) to evaluate the impact of the number of network layers on the model performance. As summarized in Table 6, when there are more layers, our model's performance decreases in general. We see that using two layers yields the best performance on both assortative and disassortative graphs. More specifically, on the assortative graphs, the performance of a 3-layer WGCN yields good performance, but the performance drops when the number of the network layers increases to four. On disassortative graphs, the performance of a 3-layer WGCN decreases a lot compared with a 2-layer WGCN, and the performance is even worse when the number of network layers increases to four.

Table 6: Accuracy with different network layers.

Layers	Assortative		Disassortative	
	Cora-ML	Citeseer	Chameleon	Squirrel
2	**87.31±2.09**	**96.45±0.8**	**67.25±2.98**	**53.05±3.83**
3	83.79±3.56	95.84±0.45	58.53±3.95	33.79±2.86
4	47.81±6.44	90.32±1.63	41.62±10.05	20.91±4.26

5 CONCLUSION

We proposed a graph convolutional network model named WGCN that captures structural features according to nodes' local topologies. WGCN first obtains nodes' structural fingerprints via a direction and degree aware Random Walk with Restart algorithm, where the walk is guided by both edge direction and in- and out-degrees of nodes. WGCN then takes the interactions between nodes' structural fingerprints as nodes' structural features. WGCN also embeds nodes into a latent space to capture nodes' high-order dependencies and latent geometrical relationships. During the message passing, WGCN contextualizes the content and structural features of each node with learnable parameters to navigate the attention-based geometrical aggregation. Experiments show that WGCN outperforms the baselines by up to 17.07% in terms of accuracy in transductive node classification on five benchmark datasets.

6 ACKNOWLEDGMENTS

This work is partially supported by Australian Research Council (ARC) Discovery Projects DP180102050. Yunxiang Zhao is supported by the Chinese Scholarship Council (CSC).

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR*. 1–13.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*. 1725–1735.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*. 3844–3852.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.

[7] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. 2002. Latent space approaches to social network analysis. *J. Amer. Statist. Assoc.* 97, 460 (2002), 1090–1098.

[8] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. 2019. Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification. In *IJCAI*. 4532–4539.

[9] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. 2019. Node representation learning for directed graphs. In *ECML&PKDD*. 395–411.

[10] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[11] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

[13] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *NIPS*. 13354–13366.

[14] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. 2018. Covariant compositional networks for learning graphs. In *ICLR (Workshop)*.

[15] Lev Lafayette, Greg Sauter, Linh Vu, and Bernard Meade. 2016. Spartan performance and flexibility: An hpc-cloud chimera. *OpenStack Summit* (2016).

[16] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data* 13, 6 (2019), 1–25.

[17] Chensheng Li, Xiaowei Qin, Xiaodong Xu, Dujia Yang, and Guo Wei. 2020. Scalable Graph Convolutional Networks With Fast Localized Spectral Filter for Directed Graphs. *IEEE Access* 8 (2020), 105634–105644.

[18] Meng Liu, Zhengyang Wang, and Shuiwang Ji. 2020. Non-Local Graph Neural Networks. *arXiv preprint arXiv:2005.14612* (2020).

[19] Ramon Lopes, Renato Assunção, and Rodrygo LT Santos. 2016. Efficient Bayesian methods for graph-based recommendation. In *RecSys*. 333–340.

[20] Zhibin Lu, Pan Du, and Jian-Yun Nie. 2020. VGCN-BERT: augmenting BERT with graph embedding for text classification. In *ECIR*. 369–382.

[21] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *SIGIR*. 719–728.

[22] Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. 2019. Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990* (2019).

[23] Hesham Mostafa and Marcel Nassar. 2020. Permutohedral-GCN: Graph Convolutional Networks with Global Attention. *arXiv preprint arXiv:2003.00635* (2020).

[24] Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. 2017. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature Communications* 8, 1 (2017), 1–19.

[25] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. 2016. Subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928* (2016).

[26] Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. 2019. Community detection on networks with ricci flow. *Scientific Reports* 9, 1 (2019), 1–12.

[27] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NIPS*. 6338–6347.

[28] Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*. 3776–3785.

[29] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

[30] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *KDD*. 1105–1114.

[31] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[33] Yongli Ren, Martin Tomko, Flora Dilys Salim, Jeffrey Chan, Charles LA Clarke, and Mark Sanderson. 2017. A location-query-browse graph for contextual recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 2 (2017), 204–218.

[34] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *KDD*. 385–394.

[35] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2019. Multi-scale Attributed Node Embedding. *arXiv preprint arXiv:1909.13021* (2019).

[36] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.

[37] Yixin Su, Rui Zhang, Sarah Erfani, and Zhenghua Xu. 2021. Detecting Beneficial Feature Interactions for Recommender Systems. In *AAAI*.

[38] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319–2323.

[39] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *ICDM*. 613–622.

[40] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. 2020. Digraph Inception Convolutional Networks. In *NIPS*.

[41] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. 2020. Directed Graph Convolutional Network. *arXiv preprint arXiv:2004.13970* (2020).

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[43] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[44] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. Nodeaug: Semi-supervised node classification with data augmentation. In *KDD*. 207–217.

[45] Natalie Widmann and Suzan Verberne. 2017. Graph-based semi-supervised learning for text classification. In *SIGIR*. 59–66.

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks?. In *ICLR*.

[47] Ying Xu and Dongsheng Li. 2019. Incorporating graph attention and recurrent architectures for city-wide taxi demand prediction. *ISPRS International Journal of Geo-Information* 8, 9 (2019), 414.

[48] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2020. Adaptive structural fingerprints for graph attention networks. In *ICLR*.

[49] Yunxiang Zhao, Qiuhong Ke, Flip Korn, Jianzhong Qi, and Rui Zhang. 2020. HexCNN: A Framework for Native Hexagonal Convolutional Neural Networks. In *ICDM*. 1424–1429.

[50] Yunxiang Zhao, Jianzhong Qi, and Rui Zhang. 2019. Cbhe: Corner-based building height estimation for complex street scene images. In *WWW*. 2436–2447.

[51] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *AAAI*. 2942–2948.

[52] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. 2019. Neighborhood-aware attentional representation for multilingual knowledge graphs. In *IJCAI*. 10–16.

[53] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *WWW*. 499–508.

[54] Chenyi Zhuang, Nicholas Jing Yuan, Ruihua Song, Xing Xie, and Qiang Ma. 2017. Understanding People Lifestyles: Construction of Urban Movement Knowledge Graph from GPS Trajectory.. In *IJCAI*. 3616–3623.