Curvature Graph Neural Network

Haifeng Li[a], Jun Cao[a], Jiawei Zhu[a], Yu Liu[b], Qing Zhu[c] and Guohua Wu[d,*]

[a]School of Geosciences and Info-Physics, Central South University, Changsha 410083, China
[b]School of Earth and Space Sciences, Peking University, Beijing, China
[c]Faculty of Geosciences and Environmental Engineering, Southwest Jiaotong University, Chengdu, China
[d]School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China

ARTICLE INFO

Keywords:
Deep learning neural network
Graph neural network (GNN)
Ricci curvature
Topology structure
Graph-based task

ABSTRACT

Graph neural networks (GNNs) have achieved great success in many graph-based tasks. Much work is dedicated to empowering GNNs with adaptive locality ability, which enables the measurement of the importance of neighboring nodes to the target node by a node-specific mechanism. However, the current node-specific mechanisms are deficient in distinguishing the importance of nodes in the topology structure. We believe that the structural importance of neighboring nodes is closely related to their importance in aggregation. In this paper, we introduce discrete graph curvature (the Ricci curvature) to quantify the strength of the structural connection of pairwise nodes. We propose a curvature graph neural network (CGNN), which effectively improves the adaptive locality ability of GNNs by leveraging the structural properties of graph curvature. To improve the adaptability of curvature on various datasets, we explicitly transform curvature into the weights of neighboring nodes by the necessary negative curvature processing module and curvature normalization module. Then, we conduct numerous experiments on various synthetic and real-world datasets. The experimental results on synthetic datasets show that CGNN effectively exploits the topology structure information and that the performance is significantly improved. CGNN outperforms the baselines on 5 dense node classification benchmark datasets. This study provides a deeepened understanding of how to utilize advanced topology information and assign the importance of neighboring nodes from the perspective of graph curvature and encourages bridging the gap between graph theory and neural networks. The source code is available at https://github.com/GeoX-Lab/CGNN.

1. Introduction

Inspired by the great success of deep learning in the Euclidean domain, GNNs attempt to generalize neural networks to non-Euclidean domains, e.g., graphs. With the ability to automatically extract low-dimensional representations of nodes or graphs, GNNs have been widely used in recommender systems [1], intelligent transportation system [2, 3], stock prediction [4] and bot detection [5].

The core idea of GNNs is how to aggregate the features of neighboring nodes to enrich the representations of target nodes. Mainly based on the homophily assumption [6, 7], GNNs strengthen the similarity of node representations belonging to the same class and the difference of node representations between different classes. GNNs also smooth the representation of nodes on local structures according to connection. Smoothness is an intrinsic property of GNNs [8], which makes neighboring node representations similar and is beneficial for neighboring nodes classified into the same class [9]. Smoothing is a double-edged sword for GNNs: Proper smoothing can generate high-quality, low-dimensional node representations that are beneficial to downstream tasks, while oversmoothing makes node representations indistinguishable and is therefore harmful to downstream tasks. Oversmoothing is mainly caused by the edges connecting different classes of nodes in real-world datasets, which dilute the useful information of the target nodes by aggregation and cause the different classes of node representations to be too similar [10]. To alleviate oversmoothing, we need to assign greater importance to the same class of nodes and smaller importance to nodes of other classes.

GNNs measure the importance of neighboring nodes in three ways. The simplest way considers the neighboring nodes to be equally important, such as GraphSAGE [11] and GIN [12]. The second way assesses the weights of neighboring nodes by utilizing node degrees. A typical GNN adopting this method is GCN [13], which assigns the weights of neighboring nodes to be inversely proportional to the degree of neighboring nodes. The GCN-derived SGC

*Corresponding author
ORCID(s):

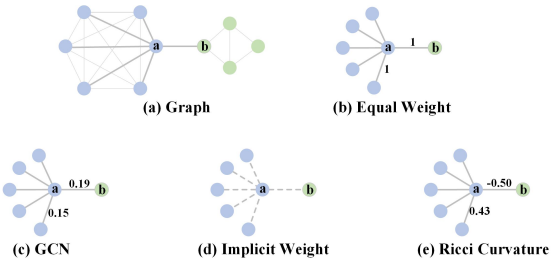


Figure 1: Illustration of different ways to compute the weights of neighbors of node a. (a) is the original graph which can be structurally divided into two classes: blue and green. For (b) and (c), the numbers of edges indicate the weights of neighbors in aggregation. The dashed lines of (d) represent the unknown weights. The numbers in (e) represent the Ricci curvature of the edges.

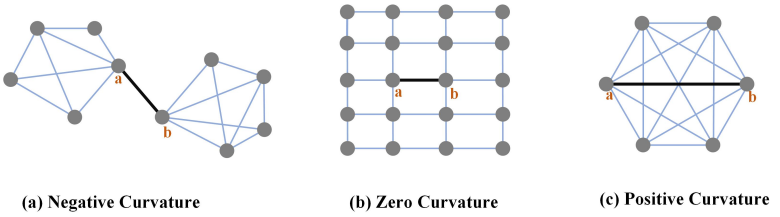


Figure 2: Illustration of structural information expressed by different values of the Ricci curvatures. The weights of all edges are 1.

[14] also leverage node degrees. Both methods explicitly compute the weights of neighboring nodes, while some other GNNs implicitly generate the weights to adapt to datasets in a data-driven manner. For example, GAT [15] calculates the weights through a self-attention mechanism.

However, the above three methods cannot properly characterize the structural importance of neighboring nodes. Many real-world datasets have an aggregating tendency for nodes, forming locally tight-connected groups with relatively sparse intergroup connections [16], named communities. In general, nodes in the same community have strong similarities, while nodes in different communities have strong differences. We argue that the denser the connection of neighboring nodes of pairwise nodes is, the stronger the structural relationship between them is, and vice versa. The nodes in Figure 1(a) can be obviously divided into two classes based on the topology structure. The key to aggregating the neighboring nodes for node a is to weaken the importance of node b. Equal importance treats node b and blue nodes as equivalents. Since the node degree of node b is smaller than blue nodes, GCN considers it more important than blue nodes. Implicit methods cannot measure the importance of node b before finishing training. Due to utilizing limited structural information such as node degree or connection, all of these ways cannot directly or explicitly weaken the importance of node b. The above three ways are shown in Figures 1(b), (c), and (d).

In this paper, we aim to improve the local structural adaptability of GNNs by exploring graph curvature. Graph curvature measures how the neighboring nodes of pairwise nodes connect with each other [17]. Analogous to curvature quantifying the deviation of a curve from a straight line in Euclidean space, discrete graph curvature measures the geometric deviation of the neighboring nodes of two nodes on an edge from a "flat" shape, e.g., the shape of a grid graph in which all nodes are structurally equivalent. There are several definitions of graph curvature, among which Ollivier's Ricci curvature [18] is the most attractive. Ollivier's Ricci curvature can quantify the strength of interaction or overlap between the neighboring nodes of pairwise nodes. In an infinite grid graph, the Ricci curvature of all edges is zero, as shown in Figure 2(b). When the connection of nodes in Figure 2(c) is denser, the Ricci curvature is positive. The edge (a, b) in Figure 2(a) connecting two independent groups which is similar to a "bridge", has a negative Ricci curvature. Naturally, the Ricci curvature diminishes the effect of node b on node a in Figure 1(a). Figure 1(e) shows that the Ricci curvature of (a, b) is -0.50, which is much smaller than the curvature between the blue node and node a. We argue that the Ricci curvature characterizes the relationship of the pairwise structure connections and should be exploited by GNNs.

Here, we propose a curvature graph neural network (CGNN), which improves the discrimination ability for local structures by leveraging the Ricci curvature. We note that CurvGN [19] has applied the Ricci curvature to GNN by implicitly converting the curvature to the weights of the neighboring nodes via the multi-layer perceptron (MLP). Nevertheless, the success of CurvGN is due to the learning ability of MLP, and the role of Ricci curvature is equivalent to the random initialization input of MLP [20]. To illustrate this, the performance of CurvGN does not change significantly even if we replace the Ricci curvature with random values randomly sampled from a uniform distribution of 0 to 1, as shown in Table 1. This means that CurvGN does not exploit the nature of the Ricci curvature. Unlike CurvGN, which implicitly maps the Ricci curvature into the weights by MLP, CGNN explicitly transforms the Ricci curvature into the weights of neighboring nodes to guarantee that the structural information is retained and utilized effectively. Since directly using the Ricci curvature as the weights makes CGNN difficult to train and degrades the performance, we propose the negative curvature transformation module (NCTM) and the curvature normalization module (CNM), both of which do not destroy the relative magnitude of curvature, i.e., edges with large curvature that have relatively large weights.

Table 1
The mean accuracies of CurvGN with different information on Cora and PubMed. The column of Ricci Curvature means the performance of CurvGN using the Ricci curvature as the input of MLP, while the other columns indicate replacing the Ricci curvature with the random values by sampling from a 0-1 uniform distribution with random seeds of 0, 10, and 100.

	Ricci Curvature	Seed=0	Seed=10	Seed=100
Cora	82.3%	82.2%	82.3%	82.1%
PubMed	78.9%	78.8%	78.9%	79.1%

We conduct extensive experiments on various synthetic and real-world datasets to illustrate that the Ricci curvature improves the local structural adaptability of GNNs. The experimental results indicate that the Ricci curvature is beneficial for measuring the strength of pairwise structural connections and that the performance of CGNN significantly outperforms the baselines on five large and dense datasets. Furthermore, we visually express that CGNN can better weaken the importance of node features between different classes. Finally, ablation experiments prove the necessity of NCTM and CNM, which guarantees the outstanding performance of CGNN. Our contributions are mainly as follows:

- We propose CGNN, which significantly improves the local structural adaptability and the quality of node representations by transforming the Ricci curvature into the weights of neighbors;
- We propose NCTM to process negative Ricci curvature and CNM to utilize the structural information of nodes in different order hops, and both of them effectively help the Ricci curvature to adapt to various datasets;
- CGNN outperforms the baselines on 5 large and dense node classification benchmark datasets

The paper is organized as follows: In Section 2, we briefly review the related work on GNNs and the discrete Ricci curvature. Section 3 describes the network architecture of CGNN and the pipeline of transforming the Ricci curvature. Section 4 evaluates the performance of CGNN on various synthetic and real-world datasets. Section 5 concludes this paper.

2. Related work

Graph Neural Network. We can classify graph neural networks into two categories: spectral GNNs and spatial GNNs [21]. Spectral GNNs define convolution through the eigendecomposition of the graph Laplacian matrix in the Fourier domain. Bruna et al. explored the generalization of CNNs to graph-structured data by the decomposition of the graph Laplacian matrix [22]. Defferrard et al. [23] approximate convolution with truncated Chebyshev polynomials, avoiding the computation of the eigenvectors of the graph Laplacian matrix. Kipf et al. proposed a graph convolution network (GCN) [13], which further simplifies convolution by setting the order of the truncated Chebyshev polynomials to 1 and implementing a renormalization trick. GCN can still be simplified to a linear model [14] by successively removing the nonlinear activation function and collapsing weight matrices between consecutive layers, which fits large-scale datasets and improves interpretability. HesGCN [24] utilizes the intrinsic local geometry structure of graph to

improve the discrimination ability of spectral GNNs by optimizing the one-order spectral graph Hessian convolutions. Since the spectrum is inherently graph-specific, a serious problem for spectral GNNs is that the trained model cannot be generalized to other datasets.

Spatial GNNs directly define convolution on the local structure of the graph to improve the generalization. The key of spatial GNNs is how to handle neighbors of different sizes and maintain the local invariance of convolution. A straightforward idea is to use different weight matrices for neighbors with different degrees [25], but this approach cannot be applied to large-scale graphs with rich node degrees. Monti et al. proposed MoNet [26], which directly generalizes CNNs to graphs and can automatically extract local, stationary, and compositional node representations. To aggregate the features of neighbors, Hamilton et al. proposed GraphSAGE [11], which introduced a fixed-size neighbor sampling method and three different ways to aggregate neighbors. GAT [15] introduced a self-attention mechanism into GNNs that implicitly assigns weights to neighbors. CurvGN [19] used the Ricci curvature as additional structural knowledge and implicitly assigned specific weights to channels of neighboring node features by MLP. Wang et al. [27] introduce the concept of capsule network into GNNs to extract distinctive features and obtains promising performance on graph classification.

Discrete Ricci Curvature. The Ricci curvature is defined on the manifold through geodesics and can be generalized to discrete space. Many works have been devoted to extending the Ricci curvature on graphs, such as Forman curvature based on the graph Laplacian [28] and Ollivier's Ricci curvature based on optimal transport theory [18]. Forman curvature is widely used in large-scale network analysis [29] and image processing [30], and Ollivier's Ricci curvature is widely applied to detect network backbone and congestion [31], study financial market fragility [32], and detect the community structure of networks [33]. Forman curvature tends to describe the interaction patterns of pairwise nodes, while Ollivier's Ricci curvature focuses on depicting the connectivity of the neighboring nodes of pairwise nodes [34]. Although computing Ollivier's Ricci curvature requires solving a linear programming problem that consumes more computational resources, it can better measure the sparsity/denseness of connections of neighboring nodes of pairwise nodes. Therefore, we select Ollivier's Ricci curvature as the advanced graph information to improve the adaptive locality ability of GNNs. Ni et al. [33] systematically compared and analysed the difference between these two types of Ricci curvature. Note that CurvGN has introduced Ricci curvature into GNNs, but the way it is introduced erases the unique property of the Ricci curvature and is completely different from ours.

3. Methodology

In this section, we elaborate on the architecture of CGNN and account for how to transform the Ricci curvature into the weights of neighbors in the aggregation. We first reasonably provide a brief discussion of the concept of Ollivier's Ricci Curvature. Then, we introduce the node classification task and formulate the forward propagation of CGNN according to the message passing neural network (MPNN) framework [35]. Finally, we propose the negative curvature transformation module and curvature normalization module to enhance the local structural adaptability of CGNN.

3.1. Ollivier's Ricci Curvature

Curvature can quantitatively measure the degree of deviation in space. In Euclidean space, curvature measures the degree to which a curve deviates from a straight line or a surface deviates from a plane. In Riemannian geometry, curvature measures the degree to which the manifold deviates from the Euclidean space, and the Ricci curvature quantifies its deviation in the orthogonal direction. The Ricci curvature determines the rate at which the volume of a ball grows, expressed as a function of the radius. It also determines the overlapping volume of two balls and the distance between their centres. If the overlapping volume is larger, the cost of the transfer is lower, which indicates that the Ricci curvature and optimal transport theory are closely related. Ollivier bridges the gap between them and generalizes the Ricci curvature to discrete space by optimal transport theory.

We now briefly describe a key distance function based on optimal transport theory to evaluate the difference between two probability distributions m_x, m_y: Wasserstein distance. The probability distribution can be viewed as an object with mass 1, and Wasserstein distance $D\left(m_x, m_y\right)$ means the minimum average mass-preserving transportation plan between m_x and m_y. For the graph, we define a probability distribution m_i for each node i, which represents the local structure of the node. $D\left(m_i, m_j\right)$ characterizes the connectivity or overlap of the local structure between node i and node j. $D\left(m_i, m_j\right)$ is small when the local structure of node i and node j shares a number of nodes, and vice versa. Then, Ollivier's Ricci curvature [18] generalizes the Ricci curvature by the Wasserstein distance from continuous space

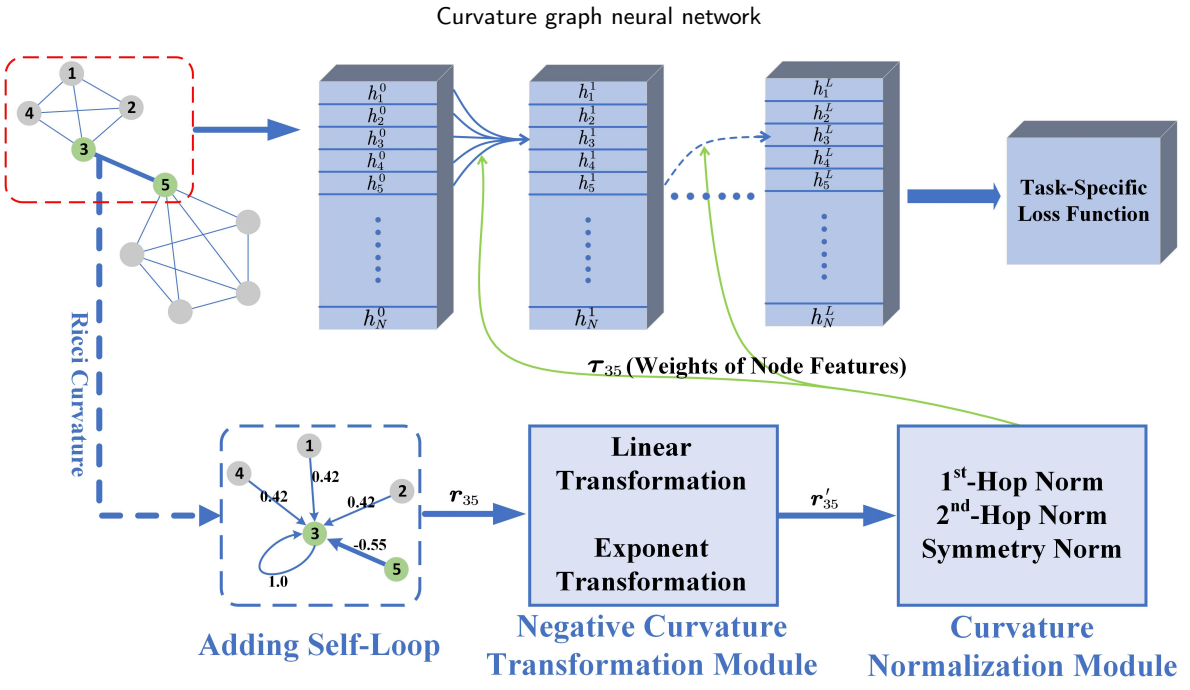


Figure 3: The architecture of the curvature graph neural network. The upper part shows message aggregation, and the lower part indicates the processing flow of the Ricci curvature.

to graphs. For simplicity, we still use the Ricci curvature to denote Ollivier's Ricci curvature. The Ricci curvature of an edge r_{ij} can be formulated as Eq. 1:

$$r_{ij} = 1 - \frac{D\left(m_i, m_j\right)}{d(i,j)} \tag{1}$$

where $d(i,j)$ indicates the shortest distance between nodes i and j. We choose a simple and effective probability distribution with a hyperparameter α regulating the relationship between the target node and the neighboring nodes [17]. For an undirected and unweighted graph, the probability distribution of node i with node degree k is Eq. 2:

$$m_i = \begin{cases} \alpha & \text{if } j = i \\ (1-\alpha)/k & \text{if } j \in \mathcal{N}(i) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Following the existing work [31], we set $\alpha = 0.5$. In addition, the Ricci curvature can be easily generalized to directed or weighted graphs. See [17] for more details. We set the Ricci curvature of the self-loop to 1.

The Ricci curvature incorporates rich topology structure information in terms of graph theory. If the curvature is positive, i.e., $D\left(m_i, m_j\right)$ being smaller than $d(i,j)$, then it indicates that the neighboring nodes of two nodes trend towards aggregation. It also implies that these two nodes are relatively closely related in structure. If the curvature of most of the edges is positive in the local structure, it can usually be referred to as community [33]. In contrast, the neighboring nodes of two nodes trend towards separation when the curvature is negative.

3.2. Curvature Graph Neural Network

We evaluate the performance of different GNNs by node classification tasks. First, the task is formulated. Suppose we have a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with N nodes $i \in \mathcal{V}$, E edges $e_{ij} = (i,j) \in \mathcal{E}$ and node features $H = \left(h_1, h_2, \ldots, h_N\right)^T \in \mathbb{R}^{N \times F}$. Here, h_i indicates the feature of node i, and F is the dimension of h_i.Given the labels of some of the nodes,

we aim to predict the labels of the remaining nodes as accurately as possible by utilizing node features and topology structure information.

The MPNN framework is currently a mainstream framework of GNNs, which is both flexible and efficient. It consists of a message-passing part and a readout part. Since the readout is for the whole graph, we focus only on the message-passing part for node classification. Message is an alternative expression of the node feature. To maintain consistency of expression, we still use node features to represent messages. The message-passing part extracts localized information by iteratively transforming, aggregating, and updating node features, as shown in the upper part of Figure 3. For some GNNs, the message-passing part can be summarized as Eq. 3:

$$h_i^{l+1} = \sigma\left(\square_{j\in\overline{\mathcal{N}}(i)}\left(\tau_{ij}^{l+1} W^{l+1} h_j^l\right)\right) \tag{3}$$

where $\overline{\mathcal{N}}(i) = \mathcal{N}(i) \cup \{i\}$ indicates the neighboring nodes of node i after adding the self-loop, h_j^l indicates the output node feature of node j in the l layer, h_j^0 means the node feature h_j, τ_{ij}^{l+1} indicates the weight of neighboring node j for node i, W^{l+1} indicates the weight matrix of the $l+1$ layer, σ indicates the activity function, and $\square$ represents a differentiable, permutation invariant aggregation function, e.g. mean, sum or max. We implement the curvature graph neural network (CGNN) under the MPNN framework, as shown in Figure 3.

Next, we describe the architecture of CGNN and the pipeline of transforming the curvature of edges into the weights of node features. Figure 3 illustrates the network architecture of CGNN, which includes a layer-stacked aggregation part and generation of node feature weights. To aggregate the features of neighboring nodes, we choose summation as the aggregation function, which is also widely adopted by other GNNs. Similar to Eq. 3, the forward propagation of CGNN is Eq. 4:

$$h_i^{l+1} = \sigma\left(\Sigma_{j\in\overline{\mathcal{N}}(i)}\left(\tau_{ij} W^{l+1} h_j^l\right)\right) \tag{4}$$

where τ denotes the weights of neighboring nodes based on the curvature which is fixed in any layer. Here, we choose Ollivier's Ricci curvature utilized in CGNN because of its distinct properties of curving the local topology structure. To generate the weights, we first need to add self-loops to nodes, then ensure that the curvature is positive by the negative curvature processing module, and finally normalize it. The workflow of the Ricci curvature is shown in the lower part of Figure 3. Similar to GCN, we set layers of CGNN to 2.

3.2.1. Negative Curvature Transformation Module

Negative curvature seriously degrades the performance of CGNN. There are always some edges in graphs in which the Ricci curvature is negative, such as e_{ab} in Figure 2(a). Note that most of the edges have a negative Ricci curvature when nodes are sparsely connected in the local structure. If we directly use the Ricci curvature with negative curvature as the weights of node features in aggregation, the negative curvature would make CGNN difficult to train. In addition, the negative curvature also has a serious impact on symmetric normalization in the curvature normalization module. Node degrees $d_i = \sum_{j\in\overline{\mathcal{N}}(i)} r_{ij}$ (the Ricci curvature being the weights of the edges) may be negative, while symmetric normalization requires calculating the root of node degrees and may result in imaginary weights.

To solve the unsatisfactory adverse effect of negative curvature, we propose the negative curvature transformation module (NCTM) to convert the negative curvature into positive numbers. We first explore the linear transformation, in which all of the Ricci curvatures simply subtract the minimum curvature and add a positive number. We formulate the linear transformation as Eq. 5:

$$r'_{ij} = r_{ij} - \min_{m,n}\left(r_{mn}\right) + \epsilon \tag{5}$$

where $\epsilon \geqslant 0$ denotes the weight of the edge with the minimum curvature after the linear transformation. The linear transformation is intuitive and efficient and ensures that the difference in curvature between edges is constant.

We also try to augment the difference between positive and negative curvature by exponential transformation, since positive and negative curvature indicates distinctly different signatures. We choose the sigmoid function to implement the transformation. The formula of the sigmoid function is Eq. 6:

$$r'_{ij} = \frac{1}{1 + e^{-r_{ij}}} \tag{6}$$

where e indicates Euler's number. The sigmoid function, as a common single increasing function, can not only transform negative curvature into positive numbers, but also enlarge the difference between positive and negative curvature by concentrating positive curvature at approximately 1 and negative curvature at approximately 0.

The NCTM ensures that there is no negative curvature after data processing.Neither the linear transformation nor the exponential transformation destroys the relative magnitude relationship of curvature and effectively preserves the property of the Ricci curvature.

3.2.2. Curvature Normalization Module

GCN smooths the features of neighboring nodes by normalizing node degrees. Li et al. [8] consider the convolution of GCN as a special kind of Laplacian smoothing. The Laplacian smoothing can be formulated as Eq. 7:

$$Y = \left(I - \gamma \tilde{D}^{-1}\tilde{L}\right) X \tag{7}$$

where Y indicates the node representation for downstream tasks, X indicates the hidden node features $X = HW$, $\tilde{A}$ indicates the adjacency matrix with the self-loop $\tilde{A}_{ii} = 1$, $\tilde{D}$ indicates the degree matrix with the self-loop $\tilde{D}_{ii} = \sum_{j=1}^{N} \tilde{A}_{ij}$, $\tilde{L} = \tilde{D} - \tilde{A}$, and $0 < \gamma \ll 1$ is a parameter balancing the target node and neighboring nodes. By setting $\gamma = 1$, we obtain $Y = \tilde{D}^{-1}\tilde{A}X$, which is the standard form of Laplacian smoothing. Then, we obtain $Y = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X$ by replacing the regularized Laplacian matrix $\tilde{D}^{-1}\tilde{L}$ with the symmetric regularized Laplacian matrix $\tilde{D}^{-1/2}\tilde{L}\tilde{D}^{-1/2}$, which is the graph convolution of GCN.

We rethink Laplacian smoothing from a structural perspective. Regularized Laplacian smoothing $\tilde{D}^{-1}\tilde{A}X$ can be considered as aggregating node features according to normalizing the degrees of 1^{st}-hop nodes, and is called 1^{st}-hop normalization. In addition, we assess the effect of 2^{nd}-hop nodes for normalizing the degrees, which can be formulated as $\tilde{A}\tilde{D}^{-1}X$. The 2^{nd}-hop normalization views the importance of neighboring nodes to the target node as inversely proportional to the degree of the neighboring nodes. This means that the more neighbors a neighboring node has, the less important it is to the target node, and vice versa. For example, a person with many friends in a social network is inevitably distracted by the energy required to maintain relationships and thus weakens his influence on friends, while a person with only a few friends will focus more on managing friendships and therefore increase his influence. Symmetric regularized Laplacian smoothing $\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}X$ seems to balance the structural information of 1^{st}-hop and 2^{nd}-hop nodes.

Here, we utilize the structural information of different orders of hops to normalize the Ricci curvature. We take the curvature processed by NCTM as the weights of the edges, and obtain the adjacency matrix $\boldsymbol{R'}$ and the corresponding degree matrix $\boldsymbol{D'}$ by adding the self-loop. We further utilize the above three normalization methods to calculate the weights of node features in aggregation. To be consistent with the MPNN framework, we reformulate the three normalization methods to the node level. The formula of the 1^{st}-hop normalization is Eq. 8:

$$\tau_{ij} = \frac{R'_{ij}}{D'_{ii}} \tag{8}$$

The formula of the 2^{nd}-hop normalization is Eq. 9:

$$\tau_{ij} = \frac{R'_{ij}}{D'_{jj}} \tag{9}$$

The formula of the symmetric normalization is Eq. 10:

$$\tau_{ij} = \frac{R'_{ij}}{\sqrt{D'_{ii} \cdot D'_{jj}}} \tag{10}$$

The results in Section 4.5 show that 1^{st}-hop normalization is appropriate on some datasets, while 2^{nd}-hop normalization is more suitable for some other datasets. The performance of symmetric normalization always seems to be a trade-off between 1^{st}-hop normalization and 2^{nd}-hop normalization.

4. Experiments and Results

4.1. Data Description

To illustrate the importance of the Ricci curvature for message passing, we conduct extensive experiments on synthetic and real-world datasets with diverse structures. We utilize the stochastic block model (SBM) [36] to generate synthetic graphs with community structure. SBM can customize the intra-community/inter-community probability for random-sampling edges. In contrast, we select the Erdős–Rényi model [37] as a null model where all of the nodes are randomly connected with the same probability. Besides, we also explore the hub structure in which a few nodes have significantly larger degrees through the Barabási–Albert model [38]. For the real-world dataset, we analyze undirected and directed graphs in detail, and Table 2 summarizes statistical details of the datasets. Cora, Citeseer, PubMed, Coauthor CS, and Coauthor Physics are citation networks, while Amazon Computers and Amazon Photo are e-commerce networks [39]. WikiCS [40] is a directed graph from Wikipedia pages on the topic of computer science.

Table 2
Statistical details of all datasets.

	Nodes	Edges	Features	Classes	Training	Undirected	Avg Degree
Cora	2,078	5,429	1433	7	140	True	3.9
Citeseer	3,327	4,732	3703	6	120	True	2.7
PubMed	19,717	44,338	500	3	60	True	4.5
Coauthor CS	18,333	100,227	6805	15	300	True	8.9
Coauthor Physics	34,493	282,455	8415	5	100	True	14.4
Amazon Computers	13,381	259,159	757	10	200	True	35.8
Amazon Photo	7,487	126,530	745	8	160	True	31.1
WikiCS	11,701	216,123	300	10	580	False	36.9

4.2. Baselines

4.2.1. Baselines for Synthetic Datasets

We select four models for comparison on synthetic datasets, including three state-of-the-art models: GCN [13], GAT with concatenation [15] and CurvGN [19]. GCN uses the node degrees to explicitly compute the weights of node features, while GAT and CurvGN implicitly generate the weights of node features in a data-driven manner. GAT introduces a self-attentive mechanism whose input is the hidden node features. CurvGN transforms the Ricci curvature into the multi-channel weights of node features through MLP. The remaining model is MLP, a particular kind of special GNN that is incapable of leveraging any advantage of the topology structure.

4.2.2. Baselines for Real-world Datasets

In addition to the above four models, we also compare Node2Vec [7] based on random walk; Chebyshev [23], GWNN [41] and ARMA [42] based on spectral graph theory; and MoNet [26], SAGE with mean aggregation [11], SGC [14], APPNP [43], PEGN [44] and GATv2 [45] for real-world datasets. These models are chosen because they utilize different graph structure information.

4.3. Experiments Setup

4.3.1. Dataset Division

Synthetic Datasets. Each dataset consists of 1000 nodes, which are equally divided into 5 classes. We randomly select 20 nodes from each class as the training set, 300 nodes as the validation set, and the remaining 600 nodes as the test set. Then, we randomly generate a 20-dimensional feature for each node as node features. For SBM, we first generate 100 graphs, of which intra-community probabilities p of randomly sampled edges range in $\{0.05, 0.07, \ldots, 0.23\}$ and inter-community probabilities q range in $\{0.0, 0.005, \ldots, 0.045\}$. Nodes within the same community are in the same class. For the random graph generated by Erdős–Rényi, the probability of randomly sampled edges is 0.01.

Real-world Datasets. For Cora, Citeseer, and PubMed, the datasets are divided in the same way as [46]. For Coauthor CS, Coauthor Physics, Amazon Computers, and Amazon Photo, the division is the same as [39]. Note that the division of the four datasets for CurvGN is different from [39]. WikiCS has a total of 20 kinds of divisions, each of which contains validation sets for hyperparameter selection and validation sets for early stopping. We choose the

first division for WikiCS, utilizing the validation set only for early stopping, and we discard the validation set for hyperparameter selection.

4.3.2. Hyperparameter Setting

To ensure the reproducibility of the paper, we choose 2020 as the random seed for any randomization operation. For the synthetic datasets, we set the dimension of the hidden layer in the models to 8 and set the head of GAT to 1. For the real-world dataset, we adjust the hidden layer dimension of CGNNs to 64. The architectures of baselines are consistent with their corresponding papers. We use the Adam SGD optimizer with a learning rate of 0.005 and L2 regularization of 0.0005, and we use the cross-entropy function as the loss function to train the models. In this paper, the weight matrix is initialized with Glorot initialization. We use an early stopping strategy based on the validation set's accuracy with a patience of 100 epochs. All models are trained on a single NVIDIA 2080Ti and are implemented via PyTorch_geometric [47]. The PyTorch implementation of CGNN refers to https://github.com/GeoX-Lab/CGNN.

4.3.3. Types of CGNN

For CGNN, we compare different configurations of NCTM and CNM. For example, CGNN_Linear_Sym means that we adopt the linear transformation and symmetric normalization operations, and CGNN_Exp_1^{st} is the combination of exponential transformation and 1^{st}-hop normalization. CGNN_***_2^{nd} indicates CGNN with 1^{st}-hop normalization. The remaining configurations are denoted similarly.

4.3.4. Evaluation Metrics

The evaluation metrics of the paper are the mean and standard deviation of classification accuracy on test nodes. We report the statistical results of 10 runs and 100 runs on synthetic datasets and real-world datasets respectively. To ensure fairness, we reuse the metrics reported by [26, 39, 15] for some baselines on some real-world datasets. In addition, we note that since classes are imbalanced in these datasets, micro-f1 is adopted to fairly compare the performance of different GNNs on the datasets with imbalanced classes.

4.4. Effect of Different Topology Structures on GNNs

To measure the effect of community structure on node feature aggregation, we investigate the classification accuracy of different GNNs on 100 random graphs generated by SBM, as shown in Figure 4(a), (b), (d), and (e). The colour indicates the classification accuracy and each grid represents a random graph generated by SBM with different intra-community and inter-community probabilities (p, q). For all heatmaps, the classification accuracy tends to decrease from the top left to the bottom right. This trend indicates that the performance of GNNs is always better when the intra-community probability is larger than the inter-community probability. Note that the smoothing of GNN enables it to distinguish nodes based on community structure, even if node features are randomly sampled. If the intra-community probability and the inter-community probability are close, SBM gradually degrades into the null model Erdős–Rényi. Then, the classification accuracies of GNNs are approximately 20%, which indicates that GNNs can no longer classify nodes. This result illustrates that appropriate smoothing is the key for GNNs, but too many inter-community edges dilute the useful information received by the target nodes, resulting in indistinguishable node features.

CGNN can better alleviate the effect caused by increasing inter-community edges. We subtract the accuracies of CGNN on 100 random graphs from GCN and CurvGN, as shown in Figure 4(c), (f). The colour indicates the difference between CGNN and GNNs. We find that CGNN outperforms GCN and CurvGN in the area of relatively large intra-community probabilities p and relatively small inter-community probabilities q, i.e., the red ellipses. The structure of real-world datasets is often similar to the red ellipses with community structure. For such datasets, the Ricci curvature can appropriately increase the weights of intra-community nodes and decrease the weights of inter-community nodes according to the topology structure, improving the quality of aggregation.

We further explored the effect of different types of structures on GNNs. We use the random graph of SBM $(p = 0.15, q = 0.025)$ as a complement to the real-world dataset with community structure, as shown in Figure 5(a). Figure 5(b) and (c) show the random graphs generated by Erdős–Rényi and Barabási–Albert respectively. Table 3 summarizes the classification accuracies of different models on these three datasets. The classification accuracies of MLP are always approximately 20%, which indicates that GNNs cannot discriminate nodes when the structure information is discarded. The accuracies of CGNN are significantly higher than those of the other models on SBM. This illustrates that the Ricci curvature remarkably enhances the adaptive locality ability of GNNs. For Erdős–Rényi, the classification accuracies of the models are approximately 20%, which implies that the topology structure has a

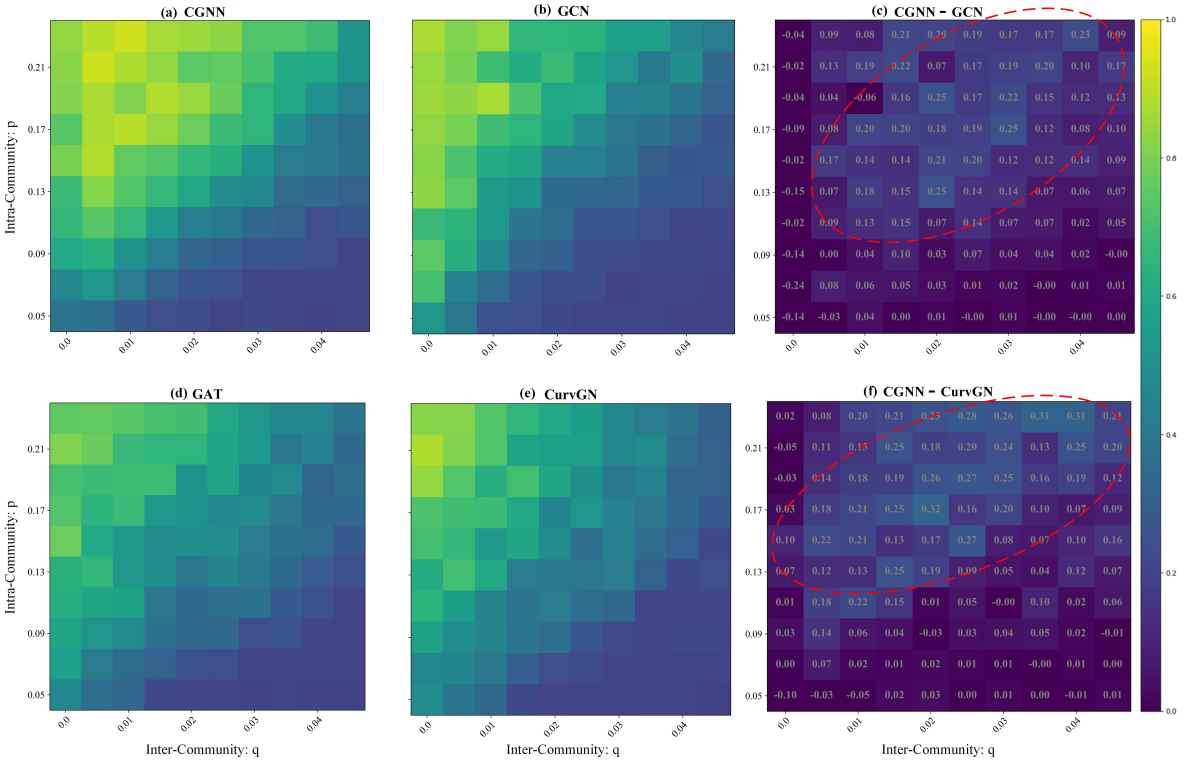


Figure 4: Heatmaps of accuracies on synthetic datasets for the stochastic block model. (a), (b), (d) and (e) represent the performance of CGNN, GCN, GAT and CurvGN respectively. (c) indicates the difference between CGNN and GCN, and (f) is between CGNN and CurvGN.

significant impact on the performance of GNNs. The accuracy of CGNN on Barabási–Albert is slightly higher than 20% but lower than that of GCN.The reason is that GCN can reduce the impact of hub nodes. However, the hub structure makes it possible for the curvature of the edges connecting different classes of nodes to be positive, incorrectly assign weights, and make node features indistinguishable, as indicated in the TexasChristian and UtahState nodes in Figure 6.

Table 3
Summary of statistical results in terms of classification accuracies (in percent) on different random graphs.

	MLP	GCN	GAT	CurvGN	CGNN
SBM	19.3±2.1	43.1±11.0	48.6±6.5	35.9±7.5	**62.1±8.2**
Erdös-Rényi	19.7±1.2	19.9±1.5	**19.9±1.2**	19.6±1.4	19.7±1.2
Barábasi-Albert	19.5±1.6	**34.4±2.5**	20.5±1.6	20.5±1.5	22.9± 2.3

4.5. Node Classification on Benchmark Datasets

In this section, we assess the performance of CGNN on the node classification benchmark datasets.The classification accuracies of CGNN and baselines are shown in Table 4. The accuracies of CurvGN on Coauthor CS, Coauthor Physics, Amazon Computers, and Amazon Photo differ from those of the original paper because CurvGN divides these four datasets differently from [39]. Therefore, we retest the performance of CurvGN on all datasets. Note that the best results of CGNN are always comparable to or even better than baselines. For the datasets with a small average node degree, the curvature of edges is mostly negative, which results in the curvature not characterizing the importance of neighboring nodes very well. APPNP achieves the best accuracies in these datasets by exploiting a large and adjustable

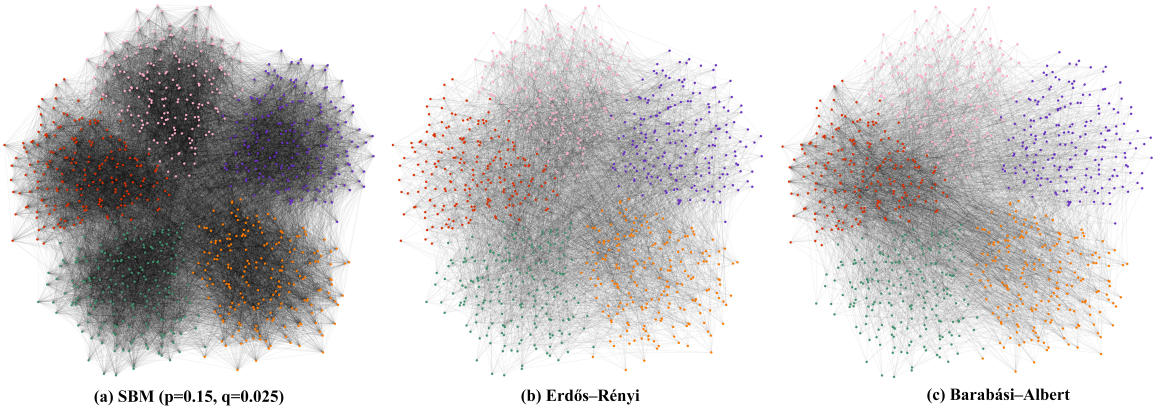


Figure 5: Visualization of three random graphs with different structures. The colours of nodes indicate the classes. In (a), the graph presents noticeable community structure. In (b), arbitrary pairs of nodes seem to be randomly connected. In (c), some of the red nodes are hubs.

neighborhood by personalized PageRank. However, the performance of CGNN is always superior for baselines on datasets with relatively higher average node degrees, and the second-best results are mostly distributed in CGNN. Even on the directed dataset WikiCS, CGNN also achieves the best accuracy. These datasets with higher average node degrees have more nodes and edges, and their local structures are more diverse and complex. For datasets with heterogeneous topology, the Ricci curvature can delicately and precisely measure the connection strength of pairwise nodes in terms of local structure, and help CGNN to better aggregate the features of neighboring nodes.

We further explore the impact of different combinations of NCTM and CNM on CGNN. For datasets with a small average node degree, exponential transformation can widen the gap between positive and negative curvature, thus helping to assist the aggregation of some nodes. In this case, the exponential transformation may be slightly better than the linear transformation. However, the exponential transformation for heterogeneous datasets with a larger average node degree would concentrate the curvature around 0 and 1 to weaken its hierarchy. The linear transformation is isometric and preserves the structural information of the curvature. The linear transformation can outperform the exponential transformation under this condition. In addition, we observe that the best results of CGNN are always distributed on 1^{st}-hop normalization or 2^{nd}-hop normalization, while symmetric normalization seems to be the trade-off. The result implies that we can try to normalize the curvature with different orders of hub nodes to pursue better accuracy. It is worthwhile to investigate what kind of normalization is suitable for which property of datasets.

We also evaluate the performance of CGNN on datasets with imbalanced classes. Micro-f1 of CGNN and baselines on different datasets are shown in Table 5. Micro-f1 of CGNN is lower than the best results of baselines on some datasets, but CGNN still outperforms baselines on larger and denser datasets, on which the Ricci curvature can better represent the local structural dependence of pairwise nodes. This indicates that the structural properties of the Ricci curvature are instrumental in alleviating the effect of imbalanced classes.

4.6. Visual Analysis of Node Classification

In this section, we select a simple but community-structured dataset, the Football dataset, to explore how the structural information affects the classification results of different GNNs in detail. The dataset is the schedule of Division I American college football games for the 2000 season: nodes indicate the teams of colleges and edges indicate regular-season games between two teams. The 12 college football conferences form the communities, and the nodes within the same community are considered to be of the same class. According to football college conference memberships, nodes are grouped together and colour-coded, as shown in Figure 6(a). The intra-community edges are relatively tight, while the inter-community edges are relatively sparse. Then, we compute the Ricci curvature of the edges and represent the Ricci curvature by colour, as shown in Figure 6(c), (d). The curvature of the intra-community edges is almost always positive, while that of the inter-community edges is mostly negative. We note that the positive curvature of inter-community edges is mainly located at the nodes TexasChristian and UtahState, because most of the neighboring nodes of these two nodes belong to the communities Western Athletic and Sun Belt, respectively.

Table 4
Summary of statistical results in terms of the mean test set classification accuracies (in percent) and standard deviation on eight node classification benchmark datasets. **Red** numbers indicate the best performance, and **bold** numbers mean the second-best performance. OOM means out of memory.

	Cora	Citeseer	PubMed	CS	Physics	Computers	Photo	WikiCS
MLP	59.0±0.9	58.9±0.7	67.1±0.5	88.9±0.6	87.5±0.8	67.7±1.2	81.8±0.8	72.5±0.2
Node2Vec	71.5±1.0	62.3±0.9	73.1±1.3	82.1±0.8	75.8±1.2	86.8±0.6	71.8±0.4	71.6±0.4
Chebyshev	81.2±1.0	69.8±1.3	74.1±2.5	OOM	OOM	OOM	OOM	OOM
GCN	81.5±1.3	71.9±0.9	77.8±2.9	91.1±0.5	92.8±1.0	82.6±2.4	91.2±1.2	72.4±0.3
MoNet	81.3±1.3	71.2±2.0	78.6±2.3	90.8±0.6	92.5±0.9	83.5±2.2	91.2±1.3	OOM
GraphSAGE	79.2±7.7	71.6±1.9	77.4±2.2	91.3±2.8	93.0±0.8	82.4±1.8	91.4±1.4	78.0±0.2
GAT	81.8±1.3	71.4±1.9	78.7±2.3	90.5±0.5	92.5±0.9	78.0±19.0	85.7±20.3	77.3±0.3
SGC	81.0±0.1	71.7±0.3	77.9±0.5	OOM	OOM	82.0±0.4	90.9±0.2	71.3±0.2
GWNN	81.9±0.7	71.5±0.4	78.1±0.8	92.0±0.3	OOM	OOM	OOM	OOM
APPNP	**83.3±0.7**	**72.3±0.4**	**80.2±0.2**	92.5±0.2	93.3±0.2	83.2±0.7	91.7±0.7	77.6±0.3
CurvGN	82.3±0.5	71.9±0.6	78.9±0.4	92.5±0.3	93.4±0.2	83.5±0.5	91.3±0.5	75.4±0.3
PEGN	82.9±0.9	71.7±0.6	79.0±0.3	92.9±0.4	93.0±0.5	82.4±1.0	91.6±0.9	76.8±0.3
ARMA	82.3±0.5	71.7±0.6	78.3±0.8	93.0±0.5	92.9±0.6	81.1±1.1	91.7±0.8	OOM
GATv2	**83.1±0.7**	71.5±0.9	78.9±0.5	91.0±0.8	OOM	83.0±1.2	91.3±0.9	77.9±0.3
CGNN_Linear_Sym	81.6±0.6	71.6±0.6	78.2±0.4	93.0±0.3	93.5±0.4	83.5±0.6	91.6±0.4	77.3±0.3
CGNN_Linear_1^{st}	81.6±0.6	71.5±0.5	78.0±0.3	92.5±0.3	93.4±0.3	83.5±0.6	**91.8±0.5**	**78.5±0.2**
CGNN_Linear_2^{nd}	81.5±0.5	71.8±0.6	78.1±0.4	**93.5±0.3**	**93.8±0.4**	**84.0±0.7**	91.5±0.7	76.3±0.4
CGNN_Exp_Sym	82.5±0.6	71.8±0.7	**79.4±0.3**	92.9±0.3	93.5±0.4	83.5±0.5	91.5±0.5	76.7±0.3
CGNN_Exp_1^{st}	82.8±0.7	71.4±1.0	78.4±0.4	92.1±0.2	93.5±0.4	83.5±0.6	**91.9±0.4**	**78.3±0.2**
CGNN_Exp_2^{nd}	82.5±0.6	**72.1±0.7**	78.9±0.5	**93.2±0.3**	**93.7±0.3**	**83.8±0.7**	91.4±0.6	75.7±0.4

Table 5
Summary of statistical results in terms of the mean test set micro-f1 (in percent) and standard deviation on eight node classification benchmark datasets. **Red** numbers indicate the best performance, and **bold** numbers mean the second-best performance. OOM means out of memory.

	Cora	Citeseer	PubMed	CS	Physics	Computers	Photo	WikiCS
GCN	81.2±0.4	71.4±0.3	77.6±0.2	91.8±0.2	92.1±0.5	81.5±1.1	90.6±0.6	72.1±0.2
GAT	81.1±0.7	71.1±0.8	77.4±0.5	91.1±0.3	92.0±0.7	78.7±12.0	89.7±21.3	77.1±0.3
SGC	79.4±0.1	70.5±0.1	75.8±0.1	OOM	OOM	81.3±0.2	88.9±0.1	75.3±0.1
APPNP	**82.7±0.6**	**72.0±0.4**	**79.9±0.4**	92.2±0.2	92.6±0.3	81.0±1.2	89.2±1.7	77.2±0.3
CurvGN	81.6±0.6	71.3±0.7	79.1±0.4	92.0±0.4	92.3±1.4	**82.3±1.4**	90.4±0.5	75.2±0.3
PEGN	81.1±1.0	**71.5±0.2**	77.5±0.5	92.6±0.2	92.5±0.4	81.2±1.1	90.8±0.8	76.6±0.3
GATv2	**82.3±0.9**	69.1±1.3	77.0±1.1	90.9±0.7	OOM	82.0±1.9	90.6±1.4	77.5±0.4
CGNN_Linear_Sym	80.9±0.6	71.1±0.6	78.5±0.3	**92.9±0.2**	92.1±0.3	82.1±1.5	**91.0±0.7**	77.1±0.2
CGNN_Linear_1^{st}	91.0±0.5	70.7±0.6	78.0±0.3	92.2±0.3	92.0±0.3	81.8±1.1	90.6±0.9	**78.6±0.2**
CGNN_Linear_2^{nd}	80.7±0.8	71.1±0.5	78.2±0.4	**93.3±0.4**	92.4±0.3	**82.4±1.1**	90.4±1.2	75.9±0.4
CGNN_Exp_Sym	81.8±0.7	71.3±0.6	**79.3±0.4**	92.5±0.3	**92.8±0.4**	82.2±1.2	**91.1±0.7**	76.8±0.2
CGNN_Exp_1^{st}	81.9±0.5	70.7±0.7	78.0±0.4	91.6±0.4	92.7±0.3	81.5±1.7	90.4±1.0	**78.4±0.2**
CGNN_Exp_2^{nd}	81.8±0.6	71.4±0.4	78.7±0.6	92.8±0.4	**92.9±0.2**	82.2±1.2	90.5±0.6	75.0±0.5

We use the adjacency matrix of the dataset as node features, randomly select one node for each class as the label, and the rest of the nodes as the test set. The colours of the nodes in Figure 6(b), (c), and (d) represent the prediction results of GCN, CGNN, and CurvGN, respectively. The classification accuracy of CGNN (85%) outperforms GCN (82%) and CGNN (81%), which indicates that CGNN makes better use of the structural information. Furthermore, we illustrate that the Ricci curvature efficiently improves the adaptive locality ability of GNNs by analysing the nodes in the light green shaded area. Both GCN and CurvGN misclassified ArkansasState and NewMexicoState in Sun Belt as

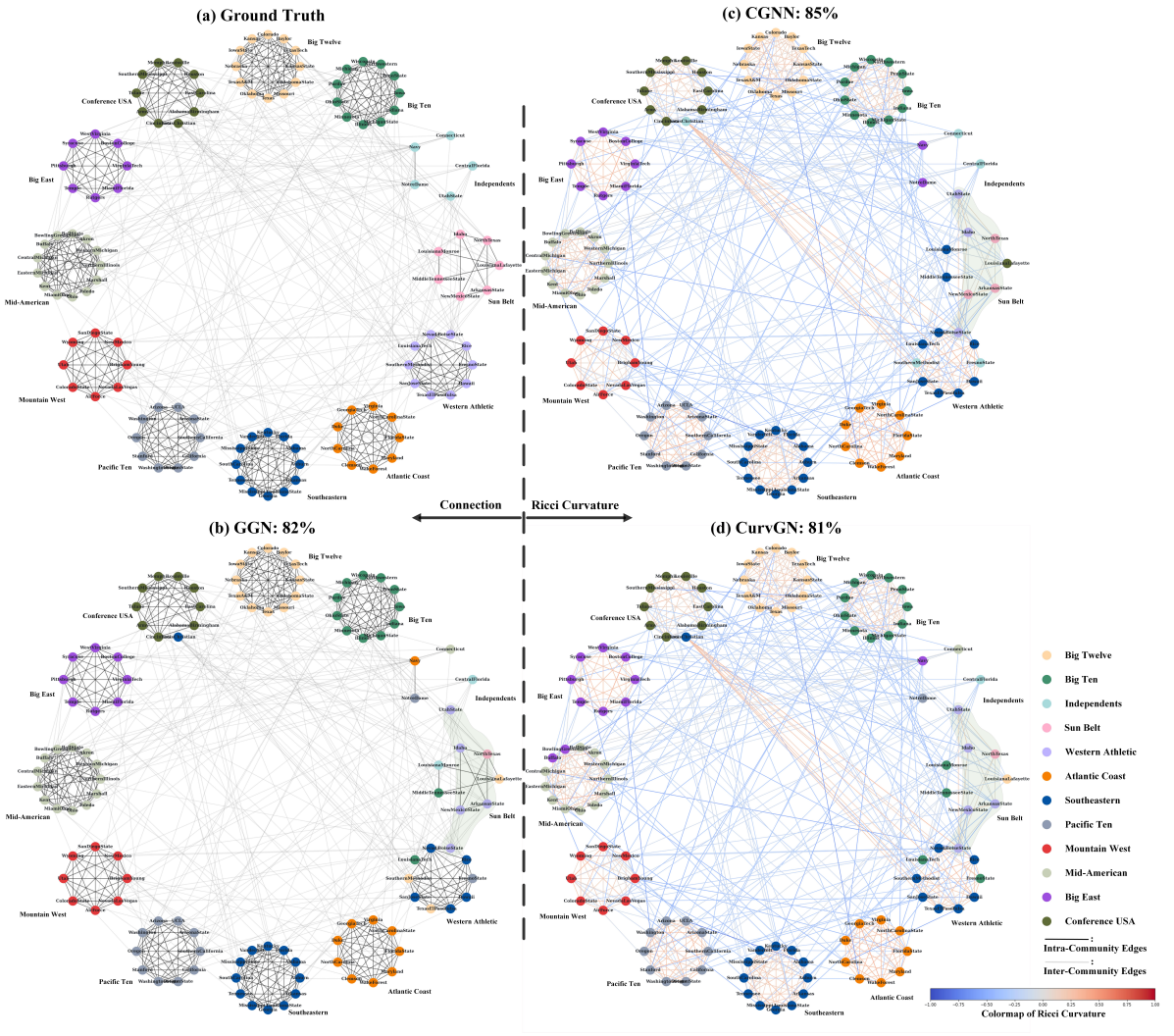
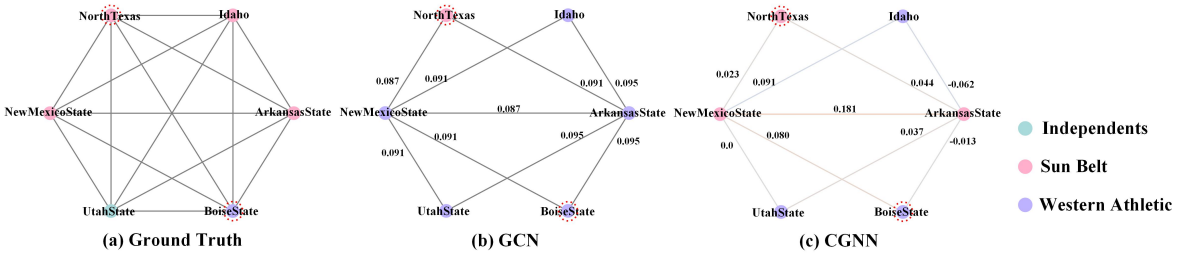


Figure 6: Visualization of the node classification prediction results of three typical models on the Football dataset. The colours of nodes indicate the predicted classes. The black edges of (a) and (c) represent the intra-community edges, while the grey edges represent the inter-community edges. The colour of edges in (b) and (d) indicates the Ricci curvatures of the edges.

Western Athletic, while CGNN classified them correctly. The intersections of neighbor nodes of ArkansasState and NewMexicoState are BoiseState, NorthTexas, Idaho, and UtahState. Among them, BoiseState and NorthTexas are the labelled nodes of the Western Athletic and Sun Belt, respectively. For the Idaho and UtahState nodes, all three models misclassify them into Western Athletic classes.

The key to correctly predicting the ArkansasState and NewMexicoState classes is to weaken the impact of BoiseState, Idaho, and UtahState on these two nodes and strengthen that of NorthTexas. The six nodes coincidentally constitute a fully connected subgraph, as shown in Figure 7(a). We further simplify the subgraph to keep only the edges connecting ArkansasState and NewMexicoState. Since the node degrees of ArkansasState, NewMexicoState, NorthTexas, BoiseState, Idaho, and UtahState are 11, 12, 11, 10, 10 and 10, respectively, GCN considers that BoiseState, Idaho, and UtahState play a more important role for ArkansasState and NewMexicoState than NorthTexas as shown in Figure 7(b). However, the curvature of NorthTexas to ArkansasState and NewMexicoState is significantly greater than that of BoiseState, Idaho, and UtahState. CGNN properly utilizes this property to weaken the effect of BoiseState, Idaho, and UtahState and to strengthen that of NorthTexas, as shown in Figure 7(c). CurvGN does not utilize the Ricci curvature, and it misclassifies BowlingGreenState and Buffalo in Mid-American as Big East classes.



(a) Ground Truth (b) GCN (c) CGNN

Figure 7: Visualization of the node classification prediction results of three typical models on the Football dataset. The colours of nodes indicate the predicted classes. The black edges of (a) and (c) represent the intra-community edges, while the grey edges represent the inter-community edges. The colour of edges in (b) and (d) means the Ricci curvatures of the edges.

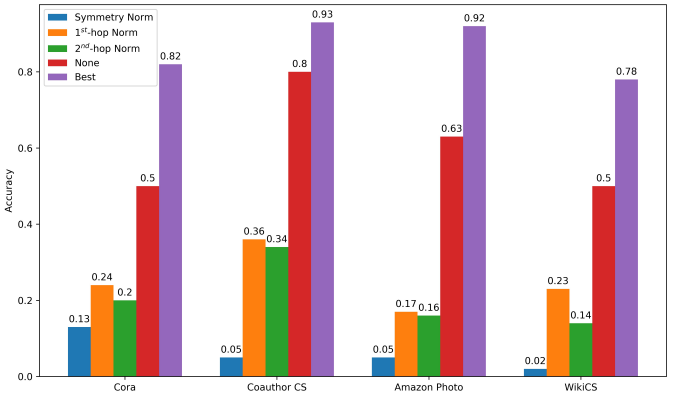


Figure 8: Comparison of predicted accuracies under different normalization methods with/without NCTM. Best represents the best accuracies of CGNN. None indicates CGNN without NCTM and CNM. The rest indicates CGNN without NCTM.

The result suggests that CGNN effectively exploits the property of Ricci curvature to enhance the ability to evaluate the strength of connections between nodes on local structures.

4.7. Ablation Experiment

We design ablation experiments to illustrate the necessity of NCTM. First, we compare the accuracy of CGNN without NCTM to the best accuracy achievable by CGNN on four datasets, and the results are shown in Figure 8. We find that once NCTM is removed, the performance of CGNN inevitably degrades significantly. Under this condition, CGNN with CNM performs even worse than the model without CNM. Due to the negative curvature, symmetry normalization may generate imaginary weights of node features, resulting in worse accuracy than random normalization. For 1^{st}-hop normalization or 2^{nd}-hop normalization, the performance of the model is also only slightly better than that of the random model. This experiment shows that the NCTM has a crucial impact on the convergence of CGNN.

Furthermore, we explore the effect of CNM on the adaptation of CGNN to various datasets. We compare the accuracy of CGNN without CNM to its best performance, and the results are shown in Figure 9. The accuracy of CGNN without CNM decreases only by 1 to 3 percentage points on Cora and Coauthor CS, but it decreases by more than 10 percentage points on the other two datasets. In particular, on Amazon Photo, the accuracy drops by 50 percentage points. It is worth noting that Cora and Coauthor CS are both citation-based datasets, while Amazon Photo is an e-commerce dataset and WikiCS is an Internet dataset. These results show that the normalization module can effectively extend the range of datasets to which CGNN is adapted.

4.8. Sensitivity Analysis of Hyperparameters

Effect of the number of layers. We evaluate the classification accuracies of CGNN with different numbers of layers on different datasets, as shown in Figure 10. The experimental results show that when the number of layers is

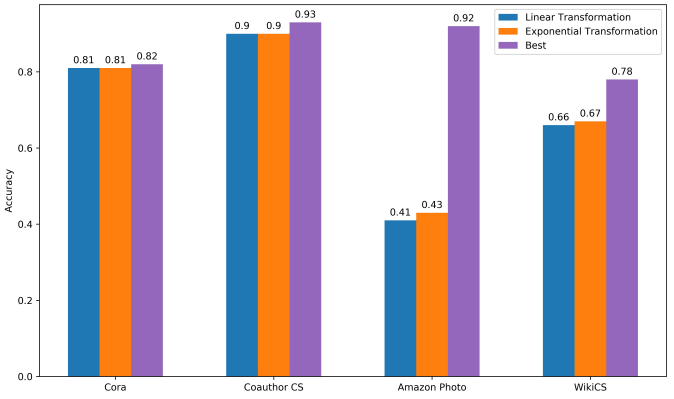


Figure 9: Comparison of predicted accuracies under different negative curvature transformation methods with/without CNM. Best represents the best accuracies of CGNN, and the rest indicates CGNN without CNM.

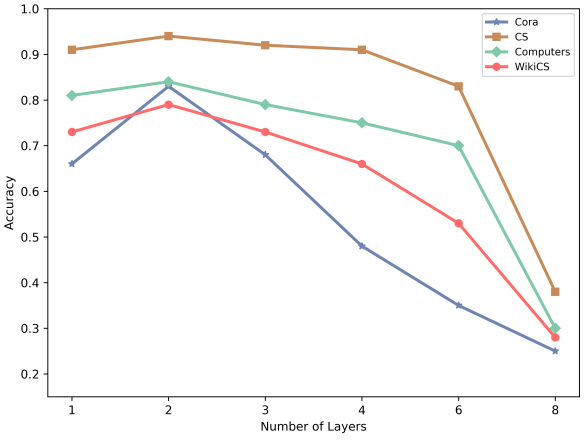


Figure 10: Classification accuracies of CGNN with different numbers of layers.

2, CGNN is more robust. When the number of layers is 1, CGNN struggles to learn high-quality node representations due to insufficient model capacity. Nevertheless, CGNN suffers from the problem of oversmoothing, resulting in indistinguishable node representations between different classes [8], when the number of layers is excessive, e.g., 6. In summary, oversmoothing is caused by the aggregation mechanism. DropEdge [48] tries to alleviate oversmoothing by randomly dropping a portion of the edges. We conjecture that it may be more effective to purposefully drop edges depending on the topology structure. Since the Ricci curvature can characterize the local connectivity, it is worth exploring how to biasedly drop edges based on the Ricci curvature.

The effect of ϵ. ϵ determines the weight of the edge with the smallest curvature after linear transformation. We measure the classification accuracies of CGNN_Linear_2^{st} with different ϵ values, as shown in Table 6. We find that the performance of CGNN significantly varies when ϵ is large enough. Note that the range of the Ricci curvature is from -1 to 1. In this case, CRM noticeably weakens the impact of curvature on the weights of neighboring nodes in aggregation because of the normalization operation. If ϵ is infinite, the weights of neighboring nodes are dependent only on the node degrees but not on the curvature. When ϵ is small, the structural properties of curvature are well captured in the weights of neighboring nodes. Even if ϵ is 0, it is equivalent to deleting the edge with the smallest curvature and does not break the distribution of curvature. Therefore, we propose to set ϵ to 0.

Table 6
Classification accuracies of CGNN_Linear_2^{st} with different ϵ values (in percent). **Bold** numbers indicates the best results.

	0	0.01	0.1	0.5	1	10	100
Cora	81.5±0.5	81.6±0.7	82.0±0.9	82.3±0.6	82.3±0.7	82.7±0.7	**82.7±0.6**
Coauthor CS	**93.5±0.4**	93.4±0.3	93.0±0.3	93.1±0.3	92.8±0.3	92.6±0.4	92.7±0.4
Amazon Computers	84.0±0.7	**84.1±0.7**	83.2±0.7	82.7±0.9	82.6±0.8	82.5±0.8	82.3±0.8
WikiCS	**76.3±0.4**	76.1±0.4	75.9±0.3	75.6±0.4	75.0±0.4	74.0±0.4	74.2±0.4

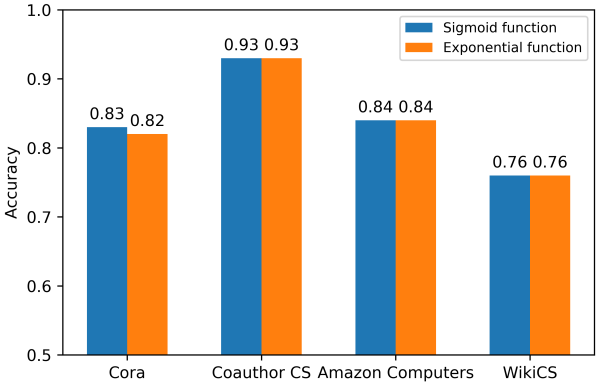


Figure 11: Classification accuracies of CGNN_Exp_2^{st} with different exponential transformations.

The effect of the form of exponential transformation. We investigate the effect of different exponential transformations on CGNN. We compare the classification accuracies of CGNN with the sigmoid function and the exponential function $e^{r_{ij}}$ on different datasets, as shown in Figure 11. The exponential function does not significantly affect the performance of CGNN. The reason is that the exponential function ensures that the transformed curvature is positive and does not destroy the relative magnitude of the curvature. The experimental results suggest that any monotonically increasing function can be a possible form of exponential transformation, and deciding which form is better is left for future work.

4.9. Computational Complexity Analysis

Training CGNN is fast and efficient. We compare the time spent training CGNN and baselines on different datasets, as shown in Table 7. We observe that CGNN spends less time on training than the baselines whether the datasets are small and sparse or large and dense. This is attributed to the simple but effective NCTM and CNM which transform the Ricci curvature into the weights of the neighboring nodes at a negligible computational cost.

Table 7
The running time(s) of training baselines and CGNN with the linear transformation and the symmetric normalization on different datasets.

	GCN	GAT	Appnp	CurvGN	PEGN	CGNN
Cora	4	5	5	4	5	2
Coauthor CS	6	7	7	8	9	4
Amazon Computers	5	10	5	16	18	3
WikiCS	5	8	6	10	11	3

Exactly calculating the Ricci curvature is somewhat time-consuming because it requires solving a linear programming problem. For an edge (i, j), the linear programming problem has $d_i \times d_j$ variables and $d_i + d_j$ linear constraints,

in which d_i and d_j are the degrees of the nodes i and j. The computational complexity is $O((d_i \times d_j)w)$, in which w is the exponent of matrix multiplication (the best known is 2.373). The running time of computing the Ricci curvature is shown in 8. The running time spent computing the curvature is significantly higher than the time spent training CGNN, especially on large and dense datasets. Nevertheless, we compute just once the Ricci curvature by caching it. In addition, we can use the approximation method [49] to accelerate the computation of the Ricci curvature. Both of these measures can significantly reduce the computational resources required to compute the curvature.

Table 8
The running time(s) of computing the Ricci curvature.

Datasets	Cora	Citeseer	PubMed	Coauthor CS	Coauthor Physics
Times	3.7	3.3	30.2	51.3	223.6

5. Conclusions

In this paper, we propose a novel graph neural network called CGNN, improving the discriminative power on the structural importance of neighboring nodes by exploiting Ollivier's Ricci curvature. CGNN utilizes the structural signatures of the Ricci curvature, which measures the interaction or overlap between the neighborhoods of pairwise nodes, to selectively aggregate the features of neighboring nodes. To stabilize the training and improve the performance of CGNN, we transform the negative curvature into positive by the NCTM and normalize the Ricci curvature by the CNM according to the structural information of hops of different orders Experimental results on synthetic datasets show that CGNN significantly outperforms baselines when community structures exist in datasets. For real-world datasets with heterogeneous topology, CGNN achieves comparable or even better results than baselines. Furthermore, we illustrate that CGNN can enhance the connections between intra-community nodes and weaken the connections between inter-community nodes by analyzing the classification results of CGNN, GCN and CurvGN in detail on the Football dataset. Ablation experiments show that the NCTM and the CNM are crucial for the superior performance of GCNN.

In the future, we will explore how the Ricci curvature relieves the oversmoothing of GNNs and improves the generalization ability of GNNs. Randomly dropping fractional edges during the training of GNNs has been demonstrated to be an effective way to alleviate the oversmoothing triggered by stacking many layers [48]. For the node classification task, only oversmoothing node representations of different classes is harmful, while smoothing node representations of the same class is beneficial. Inspired by the Ricci curvature used for community detection, we can drop edges based on the Ricci curvature, e.g., edges with negative curvature have a higher probability of being deleted than those with positive curvature. We also try to extend the Ricci curvature to graph classification tasks, because the Ricci curvature represents the connectivity of the local structure rather than that of a specific graph. We consider that the local structures are similar for the same class of graphs, and the Ricci curvature distributions are also similar. The Ricci curvature combined with the learning ability of GNNs may empower models to generate more representative graph representations, and improve the generalization of GNNs to unknown datasets.

6. Acknowledgments

We appropriate anonymous reviewers for their valuable suggestions to make this paper better. This work was supported in part by the National Natural Science Foundation of China under Grant 41871364, Grant 41861048. We appropriate the High Performance Computing Platform of Central South University to provide HPC resources.

References

[1] Xiaoyan Gao, Fuli Feng, Heyan Huang, Xian-Ling Mao, Tian Lan, and Zewen Chi. Food recommendation with graph convolutional network. *Information Sciences*, 584:170–183, 2022.

[2] Hao Peng, Bowen Du, Mingsheng Liu, Mingzhe Liu, Shumei Ji, Senzhang Wang, Xu Zhang, and Lifang He. Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning. *Information Sciences*, 578:401–416, 2021.

[3] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858, 2019.

[4] Wei Chen, Manrui Jiang, Wei-Guo Zhang, and Zhensong Chen. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences*, 556:67–94, 2021.

[5] Jun Zhao, Xudong Liu, Qiben Yan, Bo Li, Minglai Shao, and Hao Peng. Multi-attributed heterogeneous graph convolutional network for bot detection. *Information Sciences*, 537:380–393, 2020.

[6] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[7] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[8] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[9] Zhijie Deng, Yinpeng Dong, and Jun Zhu. Batch virtual adversarial training for graph convolutional networks. *arXiv preprint arXiv:1902.09192*, 2019.

[10] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445, 2020.

[11] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[12] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[16] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[17] Yong Lin, Linyuan Lu, and Shing-Tung Yau. Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series*, 63(4):605–627, 2011.

[18] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009.

[19] Ze Ye, Kin Sum Liu, Tengfei Ma, Jie Gao, and Chao Chen. Curvature graph network. In *International Conference on Learning Representations*, 2019.

[20] Haifeng Li, Jun Cao, Jiawei Zhu, Qing Zhu, and Guohua Wu. Graph Information Vanishing Phenomenon inImplicit Graph Neural Networks. *arXiv e-prints*, page arXiv:2103.01770, March 2021.

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[22] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[23] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[24] Sichao Fu, Weifeng Liu, Dapeng Tao, Yicong Zhou, and Liqiang Nie. Hesgcn: Hessian graph convolutional networks for semi-supervised classification. *Information Sciences*, 514:484–498, 2020.

[25] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2224–2232, 2015.

[26] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.

[27] Ying Wang, Hongji Wang, Hui Jin, Xinrui Huang, and Xin Wang. Exploring graph capsual network for graph classification. *Information Sciences*, 581:932–950, 2021.

[28] Robin Forman. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete and Computational Geometry*, 29(3):323–374, 2003.

[29] Melanie Weber, Emil Saucan, and Jürgen Jost. Characterizing complex networks with forman-ricci curvature and associated geometric flows. *Journal of Complex Networks*, 5(4):527–550, 2017.

[30] Emil Saucan, Gershon Wolansky, Eli Appleboim, and Yehoshua Y Zeevi. Combinatorial ricci curvature and laplacians for image processing. In *2009 2nd International Congress on Image and Signal Processing*, pages 1–6. IEEE, 2009.

[31] Chien-Chun Ni, Yu-Yao Lin, Jie Gao, Xianfeng David Gu, and Emil Saucan. Ricci curvature of the internet topology. In *2015 IEEE conference on computer communications (INFOCOM)*, pages 2758–2766. IEEE, 2015.

[32] Romeil S Sandhu, Tryphon T Georgiou, and Allen R Tannenbaum. Ricci curvature: An economic indicator for market fragility and systemic risk. *Science advances*, 2(5):e1501495, 2016.

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

[34] RP Sreejith, Karthikeyan Mohanraj, Jürgen Jost, Emil Saucan, and Areejit Samal. Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(6):063206, 2016.

[35] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[36] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

[37] László Erdős, Antti Knowles, Horng-Tzer Yau, Jun Yin, et al. Spectral statistics of erdős–rényi graphs i: Local semicircle law. *The Annals of Probability*, 41(3B):2279–2375, 2013.

[38] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[39] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[40] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[41] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019.

[42] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[43] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[44] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 2896–2906. PMLR, 2020.

[45] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[46] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[47] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[48] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

[49] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.