

Article

Deeper Exploiting Graph Structure Information by Discrete Ricci Curvature in a Graph Transformer

Xin Lai ^{1,2}, Yang Liu ², Rui Qian ³, Yong Lin ⁴ and Qiwei Ye ^{2,*} 
¹ School of Mathematics, Renmin University of China, Beijing 100872, China; laixin@ruc.edu.cn

² Beijing Academy of Artificial Intelligence, Beijing 100084, China

³ School of Information, Renmin University of China, Beijing 100872, China

⁴ Yau Mathematics Science Center, Tsinghua University, Beijing 100084, China

* Correspondence: qwye@baai.ac.cn

Abstract: Graph-structured data, operating as an abstraction of data containing nodes and interactions between nodes, is pervasive in the real world. There are numerous ways dedicated to extract graph structure information explicitly or implicitly, but whether it has been adequately exploited remains an unanswered question. This work goes deeper by heuristically incorporating a geometric descriptor, the discrete Ricci curvature (DRC), in order to uncover more graph structure information. We present a curvature-based topology-aware graph transformer, termed Curvphormer. This work expands the expressiveness by using a more illuminating geometric descriptor to quantify the connections within graphs in modern models and to extract the desired structure information, such as the inherent community structure in graphs with homogeneous information. We conduct extensive experiments on a variety of scaled datasets, including PCQM4M-LSC, ZINC, and MolHIV, and obtain a remarkable performance gain on various graph-level tasks and fine-tuned tasks.

Keywords: transformers; discrete Ricci curvature; structure information



Citation: Lai, X.; Liu, Y.; Qian, R.; Lin, Y.; Ye, Q. Deeper Exploiting Graph Structure Information by Discrete Ricci Curvature in a Graph Transformer. *Entropy* **2023**, *25*, 885. <https://doi.org/10.3390/e25060885>

Academic Editor: Kinkar Chandra Das

Received: 11 April 2023

Revised: 26 May 2023

Accepted: 27 May 2023

Published: 1 June 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Graph data include considerable structure information; however, existing graph-based algorithms do not fully use the inherent structural information of graphs. Real-world datasets with an inherent node–edge structure, such as citation networks [1], molecules [2], and the Internet [3], can be naturally represented by graphs. Moreover, graphs can be manually established in scattered data such as point clouds [4,5].

The vast majority of GNNs use a message passing (MP) mechanism to explore the graph structure information by aggregating neighborhood information [6–8]; however, they unavoidably run into oversmoothing and oversquashing issues. Due to the MP mechanism, most graph convolution of GNNs may be considered as a special case of Laplacian smoothing [9]. Analogously to random walk on graphs, smoothing operations on graphs result in the mixing of the characteristics of individual nodes. Multiple processes are taken to smooth the characteristics of individual nodes, culminating in the reduction of variability across nodes from diverse groups. This phenomenon of the inability to classify nodes when the network is deeper is the most widely discussed defect of GNNs, i.e., oversmoothing [9,10]. Another newly discussed problem of GNNs is oversquashing [11,12], which indicates that information flows between long-distant nodes encounter an unavoidable distortion. Oversmoothing and oversquashing are inevitable side effects of MP GNNs. Rong et al. [10] alleviated oversmoothing by randomly dropping a percentage of edges in the graph. Alon and Yahav [11] tried to tackle oversquashing by adding a fully adjacent layer. However, these approaches could not totally resolve these issues [13].

Graph-based transformers are another line of recent research. Transformers were originally proposed as powerful solvers for natural language processing (NLP) tasks [14]

and soon became prevailing in many domains, such as computer vision [15], time series [16], and graph representation learning [17,18]. For graph-based transformers, current works mainly focus on how to integrate a graph structure into positional encoding (PE) in transformers [18,19]. Since graph data do not have a canonical position as in images and sequences, the most widely used PE is the graph Laplacian eigenvectors, which preserve the global structure with a permutation invariance [20]. Different from PE methods, Graphormer [21] added structural encodings to the self-attention module as a structure-aware bias of attention weights. It has been experimentally proved that Graphormer is exempt from the problem of oversmoothing. Moreover, because of the self-attention mechanism in the transformer architecture, each node in the network attends to the others as if they were entirely nearby nodes. Consequently, transformer-based graph learners can efficiently avoid the issue of oversquashing. Thus, it is natural to take graph transformers as the backbone architecture for graph-based models.

However, current graph structure descriptors, such as node degrees and shortest path distances (SPD), have limited expressiveness. Rich information in the topology of the graphs still remains unexplored. Graph-based tasks rely heavily on structure information. The basic distinction between graph data and other data types, such as pictures or sequences, is the non-Euclidean node–edge structure. Graphs can be treated as a discretized manifold [22] from the topological view. Based on the homophily assumption of most graphs, the mainstream graph-based tasks, such as node classification, link prediction and graph classification/regression, tend in essence to strengthen the connection between nodes with the same property and discriminate against nodes with different properties. To describe the geometric relationships of nodes from intra-/intercommunities, we draw inspiration from recent research focusing on developing community detection algorithms [22–24] with the help of a geometric notion, i.e., the discrete Ricci curvature (DRC) [25].

The DRC quantifies the intensity of connections between nodes and their neighborhoods with regard to the local graph topology. Node pairs being densely connected are associated with positive DRC values, while sparsely connected pairs give rise to negative DRC values. As illustrated in Figure 1, the nodes connected by a yellow edge are in the same community and have densely connected/overlapped neighborhoods, while the nodes connected by a green edge are from distinct communities with few connections/overlaps between their neighborhoods. Therefore, the DRC value of the yellow edges is 1.33, which is obviously larger than the value -0.6 of the green edges. Purple edges correspond to a scenario between the two extremes; thus, they have a DRC between -0.6 and 1.33. Intuitively, the DRC has the ability to measure the connectedness of nodes and their neighborhoods, thus it can be integrated to graph transformers to explore deeper structure information.

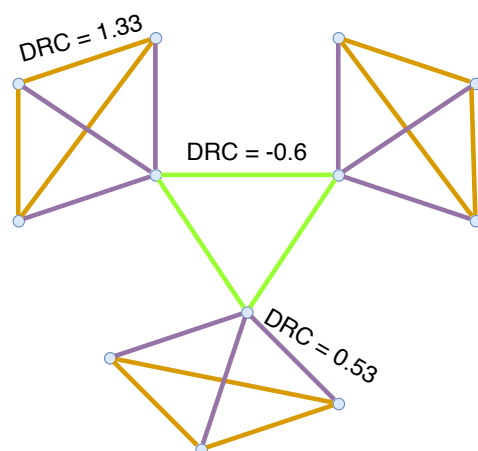


Figure 1. Illustration of DRC on a small graph. Edges with the same color have the same DRC value because of symmetry. Dense connections (yellow edges) correspond to a positive DRC, while sparse connections (green edges) have a negative DRC.

In this paper, we propose a novel curvature-based topology-aware graph transformer architecture, namely, Curvphormer, to exploit advanced structural information from a topological view. We evaluated the performance of our proposed algorithms on widely used testbeds such as MolHIV, PCQM4M-LSC, and ZINC. Curvphormer exceeded previous benchmarks by a significant margin.

2. Related Work

In this section, we highlight the most recent approaches on NN-based models working on demystifying the structure information of graph data. Then, we give prominence to some related applications of the DRC in finding the underlying structure of graphs.

2.1. Structural Encodings

2.1.1. On MP-GNNs

GNN methods processing graph data have natural merits from a theoretical basis. Most GNNs follow the MP mechanism and leverage random walk algorithms to explore the underlying structure of graphs with the aid of stochastic theories [9,26]. Some other GNN methods try to incorporate local structure information by utilizing a local k -hop subgraph as the structure fingerprint of its central node [27,28]. Moreover, some methods propose to explicitly or implicitly introduce some additional structure information encoded by geometric notions such as DRC to GNNs [29,30]. However, due to the inevitable oversmoothing and oversquashing problems and the limited expressiveness of GNNs, the increment of structure information does not yield much improvement in performance.

2.1.2. On Graph-Based Transformers

The challenge of building a powerful transformer architecture in graph representation is how to properly encode structure information into a positional encoding (PE) module [18] or the self-attention module [21]. Dwivedi and Bresson[18] exploited graph structure by pre-computing the Laplacian eigenvectors of the adjacency matrix acting as the PE in the vanilla transformer architecture to provide distance-aware information. Graph-BERT [19] operates on sampled linkless subgraphs for the local structure information and enhances its capability on extremely large graphs. Furthermore, Graph-BERT introduces three PE embeddings to take in the positional information on local subgraphs. Specifically, a Weisfeiler–Lehman (WL) absolute PE is leveraged to capture the global information, and an intimacy-based PE and a hop-based relative PE are introduced to extract the local information in subgraphs. It is notable that TokenGT [17] puts forward that pure transformers can attain impressive performance on graphs by an orthonormal node identifier and a type identifier. It suggests that the transformer architecture itself has the potential to fit in the graph structure. The key to developing transformers for graphs is to extract proper graph structure information in the model. Thus, most graph transformers incorporate graph structure information by some strong graph-specific modifications. Following this guideline, further involving advanced geometric descriptors into the transformer architecture is a promising direction.

2.2. DRC in Finding Graph Structure

In light of the property of the Ricci curvature in Riemannian geometry, the discrete version of the Ricci curvature is a natural choice as a topological descriptor. Ni et al. [3] leveraged the DRC to analyze Internet topologies. Sia et al. [23] constructed a community detection algorithm by removing negative curved edges step by step. Lai et al. [24] leveraged a DRC-based Ricci flow to deform a graph, then intracommunity nodes became closer and intercommunity nodes dispersed. The DRC is capable of finding the underlying relationship between nodes, characterizing them to clusters with identical or distinct properties.

3. Method

In this section, we elaborate the formulation of the discrete Ricci curvature (DRC) and how to incorporate it in Curvphormer. Firstly, the basic settings are stated in Section 3.1. Then, we carefully define the Ricci curvature on graphs in Section 3.2. In Section 3.3, we propose the curvature-based topology-aware Curvphormer.

3.1. Preliminaries

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple connected graph where $\mathcal{V} = \{v_1, \dots, v_n\}$ is the set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges. $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ are the number of nodes and edges, respectively. There are two kinds of information from $\mathcal{G}$, i.e.,

- **Attribute information:** It represents the attribute features carried by the datasets. For example, the signal intensity of a signal tower (which can be abstracted as a node in the network), is a kind of attribute information. Actually, not only nodes but also edges in graphs can contain attribute information. For example, the bonds between molecule pairs can have different types, which can be included in the edge features. We denote the node features by $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times d}$ and edge features by $\mathbf{E} = (\mathbf{x}_{e_1}, \dots, \mathbf{x}_{e_m})^T \in \mathbb{R}^{m \times q}$, where d and q are the dimension of node and edge features, respectively.
- **Structure information:** It represents the positions and interactions of nodes. Because of the absence of canonical node ordering, without loss of generality, position information can be viewed as a simple kind of interactions between nodes, i.e., a node is adjacent or nonadjacent to others. More complex interactions are simply represented by the node–edge form. Thus, in graphs, structure information is usually encoded by the adjacency matrix of the entire graph or subgraphs. Let $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{n \times n}$ denote the adjacency matrix, where $a_{ij} = 1$ when $(v_i, v_j) \in \mathcal{E}$, and $a_{ij} = 0$ otherwise.

3.2. Discrete Ricci Curvature

The Ricci curvature is originally a geometric notion, which plays a very important role in Riemannian manifold analysis. It quantifies the degree of space bending. For its discrete counterpart, the discretized Ricci curvature measures the connectedness of the neighborhood of two nodes. For the discretization of the Ricci curvature, there are two mainstream forms, i.e., the Ollivier Ricci curvature [25,31] and the Forman Ricci curvature [32]. Since the Ollivier Ricci curvature has more theoretical foundations and depicts inherent structures more intrinsically [33], we applied a limit-free Ollivier Ricci curvature [24,34] as the definition of the DRC.

The Ollivier Ricci curvature is defined on the base of the transportation distance. Firstly, we define the probability distribution of nodes on the graph, which indicates the connections or information flow between one node and others, especially its adjacent neighbors.

Definition 1. Probability distribution: For $\forall \alpha \in [0, 1]$ and $\forall x \in \mathcal{V}$, the information flow from node x to other nodes $y \in \mathcal{V}$ can be defined as a probability distribution on $\mathcal{V}$ by

$$m_x^\alpha(y) := \begin{cases} \alpha, & y = x, \\ (1 - \alpha) \frac{\gamma(w_{xy})}{\sum_{z \sim x} \gamma(w_{xz})}, & y \sim x, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where w_{xy} denotes the edge weight on edge $(x, y) \in \mathcal{E}$, $y \sim x$ means y is connected with x by an edge, and $\gamma(\cdot)$ is an arbitrary non-negative real-valued one-to-one function. In our experiments, we set $\gamma(w) = w$.

By the virtue of this definition, m_x^α extracts the local topology of node x on the basis of the graph. The relationship between any two nodes x and y is proportional to the distance

between their neighborhoods, which is defined as the transportation distance between two distributions m_x^α and m_y^α .

Definition 2. Transportation distance: Let $A(x, y) : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$ be a coupling satisfying

$$\sum_{y \in \mathcal{V}} A(x, y) = m_x^\alpha \quad \text{and} \quad \sum_{x \in \mathcal{V}} A(x, y) = m_y^\alpha. \quad (2)$$

Then, the transportation distance between two probability distributions m_x^α and m_y^α is defined as

$$W(m_x^\alpha, m_y^\alpha) := \inf_A \sum_{x, y \in \mathcal{V}} A(x, y) d(x, y), \quad (3)$$

where $d(\cdot, \cdot)$ is a distance function.

Here, we leveraged Dijkstra's shortest path distance as $d(\cdot, \cdot)$ in this work. In order to differentiate topology structures on the basis of graph geometry, the DRC is defined as follows:

Definition 3. α -Ricci curvature:

$$\kappa_\alpha(x, y) = 1 - \frac{W(m_x^\alpha, m_y^\alpha)}{d(x, y)}, \quad \forall \alpha \in [0, 1]. \quad (4)$$

Ollivier Ricci curvature [25]:

$$\kappa(x, y) = \lim_{\alpha \rightarrow 1} \frac{\kappa_\alpha(x, y)}{1 - \alpha}. \quad (5)$$

Note that, in the computation of Ollivier's Ricci curvature, when the node pair x and y connect densely, $\kappa(x, y)$ is larger than the sparsely connected pairs. When computing Ollivier's Ricci curvature, in order to avoid the limit operation, former works set α to 0.5 [3,22] and utilized κ_α as an approximation of κ . In this work, we leveraged another limit-free version of Ollivier's Ricci curvature for computation convenience [34].

Definition 4. Let $B : V \times V \rightarrow \mathbb{R}$ be a coupling function. We simply denote μ_x^0 as μ_x . For any $x, y \in V$, if B satisfies

- $B(x, y) > 0$, while $B(u, v) \leq 0$ for $u \neq x$ or $v \neq y$;
- $\sum_{u, v \in V} B(u, v) = 0$;
- $\sum_{v \in V} B(u, v) = -\mu_x(u)$ for all $u \neq x$;
- $\sum_{u \in V} B(u, v) = -\mu_y(v)$ for all $v \neq y$,

then we call B as a $*$ -coupling between μ_x and μ_y .

Theorem 1. The $*$ -coupling-based Ricci curvature is formulated as:

$$\kappa^*(x, y) = \frac{1}{d(x, y)} \sup_B \sum_{u, v \in V} B(u, v) d(u, v). \quad (6)$$

and for any $x, y \in V$, $x \neq y$, the following equation holds:

$$\kappa^*(x, y) = \kappa(x, y). \quad (7)$$

(Refer to [34] for proof.)

Thus, κ^* illustrates the topological characteristic of a graph as an Ollivier Ricci curvature and omits the limit calculation. In our implementation, we leveraged this κ^* curvature when computing the DRC and denoted the DRC by κ for simplicity. The proof of Theorem 1 can be found in [34]. Algorithm 1 formulates the computation of the DRC.

Algorithm 1: Computation of Discrete Ricci Curvature (DRC)**Input:** A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.**Output:** A weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w, \kappa)$, where w and κ are the weights and discrete Ricci curvature on edges, respectively.

- 1 Initialization. If $\mathcal{G}$ is unweighted, set edge weights $w_e = 1, \forall e \in \mathcal{E}$;
- 2 Compute the shortest path distance (SPD) of each pair of nodes, i.e., $d(u, v) \forall u, v \in \mathcal{V}$;
- 3 **for** $e = (x, y) \in \mathcal{E}$ **do**
- 4 Compute the discrete Ricci curvature. $\kappa_e = \frac{1}{d(x, y)} \sup_B \sum_{u, v \in \mathcal{V}} B(u, v) d(u, v)$;
- 5 **end**

3.3. Curvphormer

Curvphormer incorporates the advanced geometric information represented by the DRC into a graph-based transformer architecture. The overall architecture of Curvphormer is demonstrated in Figure 2.

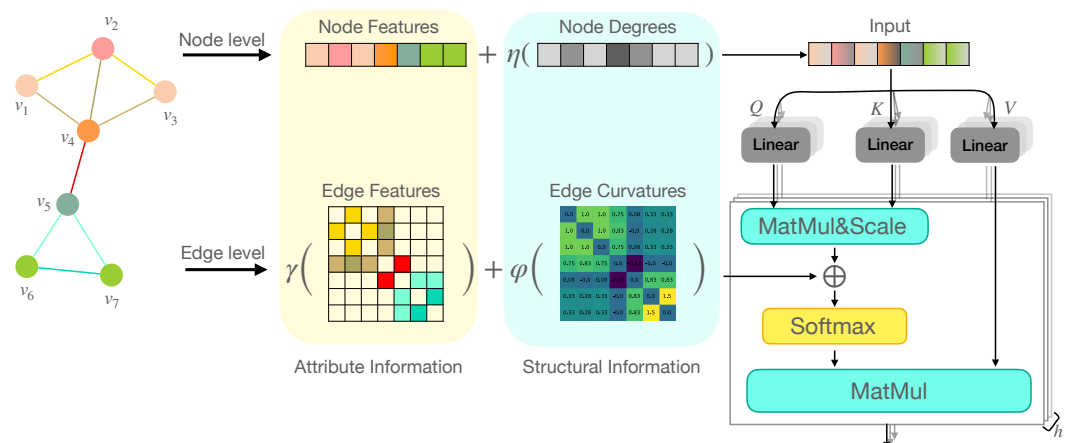


Figure 2. Illustration of Curvphormer with attribute/structure encodings. The input is a combination of two types of node-level information, i.e., node features and node degree encoding. Edge-level information, i.e., encodings of edge features and curvatures, describes the interactions between node pairs; therefore, these two encodings are added to the multihead self-attention module as a bias of the attention weights.

3.3.1. Attribute Encoding

As mentioned before, in graph data, the attribute information is the features carried by nodes and edges, describing some specific information in the dataset. Node features are the most important information characterizing a dataset. In Curvphormer, we leveraged the node features without any affine transformation. In many graphs, edges also have attribute features, which are essential for understanding the underlying graph structure. Although edge features are provided by the dataset, they usually indicate the type or intensity of the interactions between nodes. Thus, for any node pair (v_i, v_j) in a graph, the correlation between v_i and v_j has to account for the edges connecting them. Let v_i and v_j be connected by a shortest path denoted by $v_i \stackrel{e_1}{\sim} \dots \stackrel{e_N}{\sim} v_j$. The correlation between v_i and v_j can be formulated by the mean of the embedded edge features along the path.

$$\gamma(v_i, v_j) = \frac{1}{N} \sum_{k=1}^N \text{EdgeEmbedding}_k(\mathbf{x}_{e_k}), \quad (8)$$

where $\mathbf{x}_{e_k} \in \mathbb{R}^q$ is the edge feature of e_k . $\text{EdgeEmbedding}_k(\mathbf{x}_{e_k}) = \mathbf{x}_{e_k}^T \cdot \mathbf{w}_k$, $\mathbf{w}_k \in \mathbb{R}^q$ is a learnable vector.

3.3.2. Structural Encoding

Structure information here refers to the knowledge of the graph that is induced by the connectedness. As demonstrated in Figure 2, we considered two dimensions of structure information. One is the node-level information to quantify the importance of nodes in the graph. Taking the citation network as an example, the more influential a paper is, the more citations it has, and vice versa. Thus, in an abstract graph, an important node must connect to more neighbors. The node degree is an intuitive choice to describe this node property as in [21]. Let $d_i = \sum_{j \in \mathcal{V}} a_{ij}$ be the degree of node v_i . Then, we embed d_i into a vector:

$$\eta(v_i) = d_i \cdot w_i, \quad (9)$$

where $w_i \in \mathbb{R}^d$ is a learnable vector. Then, we incorporate the node's degree embedding matrix $D = (\eta(v_1), \dots, \eta(v_n))^T \in \mathbb{R}^{n \times d}$ with the node features as the input of the subsequent module, i.e., $H^{(0)} = X + D$.

The other is the edge-level information, which can be interpreted by the positional relationship between any node pairs via the edges connecting them. Former works encoded the position information on graphs by a simple shortest path distance (SPD) [21,35]. However, the SPD can only provide a relative distance on graphs. Graphs can be viewed as a discretized manifold in Riemannian spaces. Thus, the topology structure of the manifold determines the foundation of graphs. A pure SPD neglects the topology structure of the spaces where graphs are embedded in. As we stated in Section 3.2, the DRC depicts the connectedness on the basis of the node's neighborhoods. Nodes with a positive DRC connect densely, while a negative DRC is related to sparsely connected nodes. By virtue of the expressive power of DRC, we encode the relations of the nodes on the graph topology with

$$\varphi(v_i, v_j) = \kappa(v_i, v_j) \cdot w_{ij}, \quad (10)$$

where w_{ij} is a learnable scalar.

3.3.3. Self-Attention Mechanism

The self-attention module is the main part of the transformer architecture, which captures the global information by connecting all positions [14,21]. It computes the weighted sum of values, where the weights of values is obtained by a query-key function. Let $H = (h_1, \dots, h_n)^T \in \mathbb{R}^{n \times d}$ be the input of the module. In Curvphormer, when a node attends other nodes in the graph, the edge attribute information $\Gamma = \{\gamma(v_i, v_j)\}$ as well as the DRC-based structural information $\Phi = \{\varphi(v_i, v_j)\}$ are added to the attention weights to provide more topology-aware ability. Therefore, the self-attention can be formulated as

$$\text{Attention}(H) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}} + \Gamma + \Phi\right)V, \quad (11)$$

where $Q = HW_Q$, $K = HW_K$, $V = HW_V$, and $W_Q, W_K \in \mathbb{R}^{d \times d_K}$, $W_V \in \mathbb{R}^{d \times d_V}$. Thus, the correlation between nodes v_i and v_j is

$$A_{ij} = \text{softmax}\left(\frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_K}} + \gamma(v_i, v_j) + \varphi(v_i, v_j)\right)V. \quad (12)$$

The multihead self-attention is obtained by

$$\text{MHA}(H) = \text{Concat}(\text{Attention}_1(H), \dots, \text{Attention}_h(H))W_O, \quad (13)$$

where $W_O \in \mathbb{R}^{hd \times d_{\text{model}}}$.

3.3.4. Curvphormer Structure

Curvphormer follows the basic architecture of Graphormer [21], which is a variant of the vanilla transformer encoder [14]. Each layer of Curvphormer consists of a multi-head attention module (MHA) and a feed-forward network (FFN) module. The detailed implementation of a Curvphormer layer is formulated as

$$\hat{H}^{(l+1)} = \text{MHA}(\text{LayerNorm}(\mathbf{H}^{(l)})) + \mathbf{H}^{(l)} \quad (14)$$

$$\mathbf{H}^{(l+1)} = \text{FFN}(\text{LayerNorm}(\hat{H}^{(l+1)})) + \hat{H}^{(l+1)} \quad (15)$$

Moreover, in order to enhance the ability of Curvphormer to capture the representation of the entire graph, as in [21], a virtual node is applied, which is connected to all nodes in the graph by virtual edges, and the corresponding structural encodings are set to distinct learnable variables.

The training procedure of Curvphormer is mainly based on a transformer encoding module. The self-attention mechanism has a complexity of $\mathcal{O}(n^2 \cdot d)$ per layer, where n is the number of nodes, and d is the dimension of node features. Before training, Curvphormer computes the DRC as the input of the structural encoding. The computing complexity of DRC is $\mathcal{O}(m \cdot \bar{d}^3)$, where m is the number of edges, and $\bar{d}$ is the average degree of nodes. It is time-consuming to compute the DRC on very large graphs, thus we compute this valuable structure information of graphs before training.

4. Experiments

In this section, we conduct three experiments to intuitively clarify the motivation as well as effectiveness of Curvphormer. Firstly, we illustrate the importance of the topology information in Section 4.1 on a small dataset, i.e., Zachary's Karate Club Network [36], indicating the importance of our inclusion of the curvature as a factor. Then, we intuitively show the expressiveness of the DRC on graph structures comparing it with the widely used graph structure descriptor SPD in Section 4.2. Finally, we perform experiments on three different scaled real-world datasets to test the performance of Curvphormer in Section 4.3.

4.1. Structure Information Is Crucial in Graph-Based Tasks

To illustrate the importance of graph structure information, we devised a binary node classification experiment on the small Karate Club Network (Karate). Karate is composed of two communities with 34 nodes (members of the club). The edges between nodes indicate the interactions between club members. We applied a simple two-layer GCN model [6] to learn the underlying graph structure. Moreover, the node feature was designed based on three cases, i.e., random numbers, the SPD, and the DRC, for testing the influence of different kinds of information in a simple NN-based model.

The accuracy of these three scenarios is shown in Table 1 (best performance in 10 runs). For random features, even though they could not provide any useful information, the classification accuracy was still better than random guess because of the utilization of the adjacency matrix in the model. Notice that when more structure information was provided, the performance of the model improved remarkably. Moreover, the DRC outperformed the SPD in this experiment setting. It indicated that advanced topology information could extract more effective structure information than simple distance information.

Table 1. Test on different types of structure information on the Karate dataset with a 2-layer GCN. Structure information yields better results, and advanced topological DRC outperforms SPD.

Feature Type	Feature Description	Accuracy (%)
Random numbers	No useful information	78
SPD	Provides distance information for nodes	95
DRC	Provides advanced topology information	97

4.2. Why Does DRC Depict Structure Information Better than SPD?

Now, we intuitively show the expressiveness of the DRC compared to that of the SPD by a small graph composed of two small communities bridged by an edge, as shown in Figure 3. Though both the SPD and DRC had the ability to know there were two communities, the DRC depicted more in-depth structure information than the SPD. Note the interactions between nodes 1,3 and nodes 1,5. Nodes 1 and 3 were from the same community, while nodes 1 and 5 were from different communities. The relationships of these two pairs were different, while $SPD_{13} = SPD_{15} = 2$ (highlighted by orange circles in Figure 3c). Moreover, edge e_{45} was the only bridge edge connecting the two communities. However, $SPD_{45} = 1$ (red dotted circle in Figure 3c) could not differentiate e_{45} from other one-hop pairs. The SPD was incapable of describing these differences in structure. Fortunately, the DRC could amend these defects because it considered the nodes' neighborhoods. The tightly interacting pairs tended to have a larger DRC than sparsely interacting pairs. $DRC_{13} = 1$ was apparently larger than $DRC_{15} = 0.08$ for the first case. Meanwhile, $DRC_{45} = -0.83$ highlighted the difference of this edge from others.

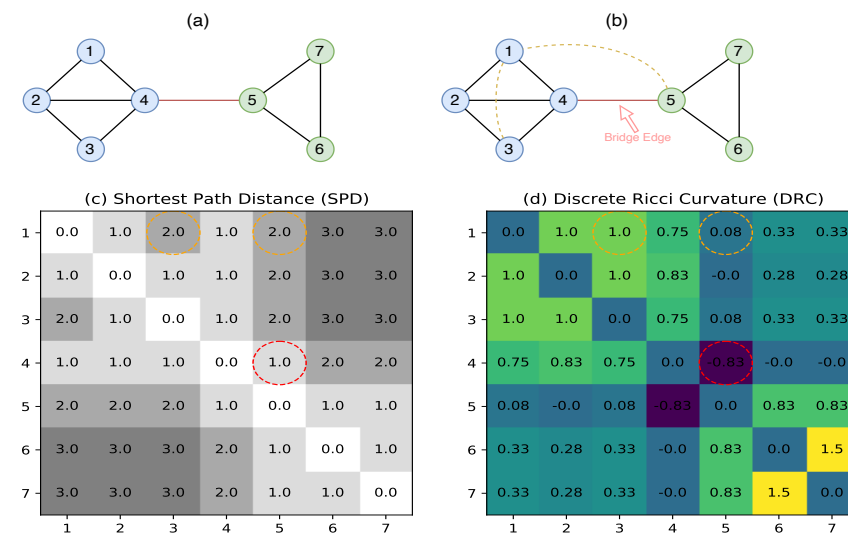


Figure 3. A small graph demonstrates the structural expressiveness of the SPD vs. the DRC. (a,b) are the graph and highlights. (c,d) are the SPD and DRC values of node pairs. The difference between (1) inter-/intra-community relations, i.e., 1 and 3 and 1 and 5, (2) the bridge edge e_{45} and other 1-hop pairs, cannot be captured by the SPD but are well described by the DRC.

4.3. Experiments on Real-Word Datasets

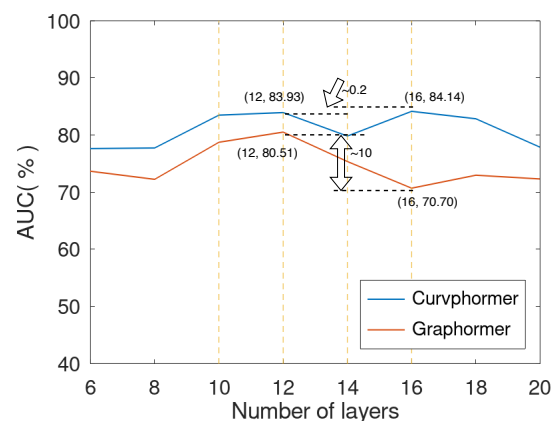
In this part, we devised our experiments on three different scaled datasets, i.e., Mol-HIV (small), ZINC (medium), and PCQM4M-LSC (large). Statistics of the datasets are summarized in Table 2. We summarize the statistics of datasets used in this work in Tables 1 and 3, and Figure 4.

Table 2. Statistics of the datasets.

DATASETS	Scale	#Graphs	#Nodes	#Edges	Task Type
ZINC (sub-set)	Small	12,000	277,920	597,960	Regression
MolHIV	Medium	41,127	1,048,738	1,130,993	Binary classification
PCQM4M-LSC	Large	3,803,453	53,814,542	55,399,880	Regression

Table 3. Results on the PCQM4M-LSC, ZINC, and MolHIV datasets. The performance metric for the regression task on PCQM4M-LSC and ZINC is the MAE, and the AUC for the classification task on MolHIV. validMAE and testMAE refer to the MAE on the validation set and test set, respectively. The test set of PCQM4M-LSC is not publicly available. Curvphormer outperforms the benchmarks on all these datasets.

Datasets	Scale	Task	Model	#Layers	#Param	validMAE
PCQM4M-LSC	Large	Regression	GCN [37]	12	2.0M	0.1691
			GIN [38]	12	3.8M	0.1537
			DeeperGCN [39]	12	25.5M	0.1398
			GT [18]	12	0.6M	0.1400
			Graphormer _{SMALL} [21]	12	12.5M	0.1264
			Graphormer [21]	12	47.1M	0.1234
			Curvphormer	8	34.1M	0.1024
ZINC	Medium	Regression	Model	#Layers	#Param	testMAE
			GIN [38]	2	510K	0.526
			GraphSage [8]	2	505K	0.398
			GAT [7]	2	531K	0.384
			GCN [37]	2	505K	0.367
			GatedGCN-PE [40]	2	505K	0.367
			PNA [41]	16	387K	0.214
			Graphormer _{SLIM} [21]	12	489K	0.122
			Curvphormer	8	34.1M	0.080
			Model	#Layers	#Param	AUC (%)
MolHIV	Small	Classification	GCN-GraphNorm [37]	12	526K	78.83
			PNA [41]	12	326K	79.05
			PHC-GNN [42]	12	111K	79.34
			DeeperGCN-FLAG [39]	12	532K	79.42
			DGN [43]	12	114K	79.70
			Graphormer-FLAG [21]	12	47.0M	80.51
			Curvphormer	12	47.1M	83.93

**Figure 4.** Testing the performance of Curvphormer on MolHIV for different number of layers. Curvphormer surpasses the baseline Graphormer by a significant margin and attains stable satisfactory performance for a varying number of layers.

4.3.1. Experimental Set-Up

We benchmarked Curvphormer with the non-topology-aware Graphormer baseline [21]. The basic setting of Curvphormer followed [21] but we modified some parameters for the model fine-tuning. The number of attention heads and the dimension of node/edge features were set to 16. We used AdamW as the optimizer and set the hyperparameter Adam- ϵ to 1×10^{-8} and Adam- (β_1, β_2) to (0.99, 0.999). The learning rate was set to 2×10^{-4} with a lower bound of 1×10^{-9} . The batch size was set to 512. All models and tasks were trained on eight NVIDIA 3080ti GPUs for about three days. Other settings were the same as those of the baseline. We trained Curvphormer on PCQM4M-LSC and ZINC from scratch. We fine-tuned the pretrained model on ZINC with the small dataset MolHIV to test the transferable ability of Curvphormer. In addition, in order to test if Curvphormer could effectively resist the performance drop caused by oversmoothing, we tested Curvphormer on the MolHIV dataset with a varying number of layers up to 20.

4.3.2. Results

Table 3 summarizes the performance of Curvphormer and other baselines on PCQM4M-LSC, ZINC, and MolHIV. The metrics were the mean absolute error (MAE) for the regression task and the AUC for the classification task. We report the MAE on the validation set (ValidMAE) for PCQM4M-LSC because its test set was not publicly available. Curvphormer achieved the best results and noticeably surpassed the previous state-of-the-art GNNs as well as the recent graph-transformer model GT [18] and Graphormer [21].

Next, we tested Curvphormer's performance further on the MolHIV dataset by comparing it with the baseline Graphormer. Figure 4 shows that both models were capable of resisting oversmoothing. Meanwhile, Curvphormer surpassed Graphormer by a noticeable margin for all layer configurations. It is noteworthy that when the model layer changed from 12 to 16, the performance of Graphormer dropped from 80.51 to 70.70. In contrast, Curvphormer achieved a comparable result after a slight drop.

5. Conclusions and Discussion

This work introduced Curvphormer, a topology-aware graph transformer that incorporates advanced structure information into an expressive Graphormer architecture. The DRC effectively differentiated the topology structure of graphs with the homophily property and helped our model achieve remarkable performance improvements on different scaled datasets in graph classification/regression tasks. It showed that applying more geometric descriptors to expressive graph models is rewarding. Meanwhile, the exploration of graph structure information is still challenging. For example, discovering the topology information of heterogeneous graphs still needs future endeavors. Moreover, the computation complexity of the DRC restricts its application in large dynamic systems. In a nutshell, Curvphormer inspires a better understanding of graph structure and encourages future work.

Author Contributions: Conceptualization, X.L. and Q.Y.; Methodology, X.L., Y.L. (Yong Lin) and Q.Y.; Validation, R.Q.; Investigation, Y.L. (Yang Liu); Writing—original draft, X.L., Y.L. (Yang Liu) and Q.Y.; Writing—review & editing, X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Program of China (2020AAA0105200) and Science and Technology Project of Hebei Education Department (Grant No. ZD2022168).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective Classification in Network Data. *AI Mag.* **2008**, *29*, 93. [\[CrossRef\]](#)
2. Irwin, J.J.; Sterling, T.; Mysinger, M.M.; Bolstad, E.S.; Coleman, R.G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.* **2012**, *52*, 1757–1768. [\[CrossRef\]](#)
3. Ni, C.C.; Lin, Y.Y.; Gao, J.; David Gu, X.; Saucan, E. Ricci curvature of the Internet topology. In Proceedings of the 2015 IEEE Conference on Computer Communications, Kowloon, Hong Kong, China, 26 April–May 2015; pp. 2758–2766.
4. Shi, W.; Rajkumar, R.R. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020.
5. Qian, R.; Lai, X.; Li, X. BADet: Boundary-Aware 3D Object Detection from Point Clouds. *Pattern Recognit.* **2022**, *125*, 108524. [\[CrossRef\]](#)
6. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
7. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
8. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; NIPS'17; pp. 1025–1035.
9. Li, Q.; Han, Z.; Wu, X. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. *Proc. AAAI Conf. Artif. Intell.* **2018**, *32*, 3538–3545. [\[CrossRef\]](#)
10. Rong, Y.; Huang, W.; Xu, T.; Huang, J. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
11. Alon, U.; Yahav, E. On the Bottleneck of Graph Neural Networks and its Practical Implications. In Proceedings of the International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021.
12. Topping, J.; Giovanni, F.D.; Chamberlain, B.P.; Dong, X.; Bronstein, M.M. Understanding over-squashing and bottlenecks on graphs via curvature. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
13. Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; Li, Y. Simple and Deep Graph Convolutional Networks. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; Volume 119, pp. 1725–1735.
14. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 6000–6010.
15. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A Survey on Vision Transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 87–110. [\[CrossRef\]](#) [\[PubMed\]](#)
16. Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; Sun, L. Transformers in Time Series: A Survey. *arXiv* **2022**, arXiv:2202.07125.
17. Kim, J.; Nguyen, D.T.; Min, S.; Cho, S.; Lee, M.; Lee, H.; Hong, S. Pure Transformers are Powerful Graph Learners. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022.
18. Dwivedi, V.P.; Bresson, X. A Generalization of Transformer Networks to Graphs. In Proceedings of the AAAI Workshop on Deep Learning on Graphs: Methods and Applications, Virtual, 8 February 2021.
19. Zhang, J.; Zhang, H.; Xia, C.; Sun, L. Graph-Bert: Only Attention is Needed for Learning Graph Representations. *arXiv* **2020**, arXiv:2001.05140.
20. Dwivedi, V.P.; Luu, A.T.; Laurent, T.; Bengio, Y.; Bresson, X. Graph Neural Networks with Learnable Structural and Positional Representations. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
21. Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; Liu, T.Y. Do transformers really perform badly for graph representation? *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 28877–28888.
22. Ni, C.C.; Lin, Y.Y.; Luo, F.; Gao, J. Community Detection on Networks with Ricci Flow. *Sci. Rep.* **2019**, *9*, 9984. [\[CrossRef\]](#)
23. Sia, J.; Jonckheere, E.; Bogdan, P. Ollivier-Ricci Curvature-Based Method to Community Detection in Complex Networks. *Sci. Rep.* **2019**, *9*, 9800. [\[CrossRef\]](#)
24. Lai, X.; Bai, S.; Lin, Y. Normalized discrete Ricci flow used in community detection. *Phys. A Stat. Mech. Appl.* **2022**, *597*, 127251. [\[CrossRef\]](#)
25. Lin, Y.; Lu, L.; Yau, S.T. Ricci curvature of graphs. *Tohoku Math. J.* **2011**, *63*, 605–627. [\[CrossRef\]](#)
26. Gasteiger, J.; Bojchevski, A.; Günnemann, S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
27. Zhang, K.; Zhu, Y.; Wang, J.; Zhang, J. Adaptive Structural Fingerprints for Graph Attention Networks. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
28. Wang, G.; Ying, R.; Huang, J.; Leskovec, J. Multi-hop Attention Graph Neural Networks. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, QC, Canada, 19–27 August 2021; pp. 3089–3096.
29. Ye, Z.; Liu, K.S.; Ma, T.; Gao, J.; Chen, C. Curvature Graph Network. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
30. Li, H.; Cao, J.; Zhu, J.; Liu, Y.; Zhu, Q.; Wu, G. Curvature graph neural network. *Inf. Sci.* **2022**, *592*, 50–66. [\[CrossRef\]](#)

31. Lin, Y.; Yau, S.T. Ricci Curvature and Eigenvalue Estimate on Locally Finite Graphs. *Math. Res. Lett.* **2010**, *17*, 343–356. [\[CrossRef\]](#)
32. Sreejith, R.P.; Mohanraj, K.; Jost, J.; Saucan, E.; Samal, A. Forman curvature for complex networks. *J. Stat. Mech. Theory Exp.* **2016**, *2016*, 063206. [\[CrossRef\]](#)
33. Samal, A.; Sreejith, R.P.; Gu, J.; Liu, S.; Saucan, E.; Jost, J. Comparative analysis of two discretizations of Ricci curvature for complex networks. *Sci. Rep.* **2018**, *8*, 8650. [\[CrossRef\]](#)
34. Bai, S.; Huang, A.; Lu, L.; Yau, S.T. On the Sum of Ricci-Curvatures for Weighted Graphs. *Pure Appl. Math. Q.* **2021**, *17*, 1599–1617. [\[CrossRef\]](#)
35. Cai, D.; Lam, W. Graph Transformer for Graph-to-Sequence Learning. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 7464–7471. [\[CrossRef\]](#)
36. Zachary, W. An Information Flow Model for Conflict and Fission in Small Groups. *J. Anthropol. Res.* **1977**, *33*, 452–473. [\[CrossRef\]](#)
37. Brossard, R.; Frigo, O.; Dehaene, D. Graph convolutions that can finally model local structure. *arXiv* **2020**, arXiv:2011.15069.
38. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
39. Li, G.; Xiong, C.; Thabet, A.; Ghanem, B. DeeperGCN: All you need to train deeper GCNs. *arXiv* **2020**, arXiv:2006.07739.
40. Bresson, X.; Laurent, T. An experimental study of neural networks for variable graphs. In Proceedings of the International Conference on Learning Representations Workshop, Vancouver, BC, Canada, 30 April–3 May 2018.
41. Corso, G.; Cavalleri, L.; Beaini, D.; Liò, P.; Veličković, P. Principal neighbourhood aggregation for graph nets. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 13260–13271.
42. Le, T.; Bertolini, M.; Noé, F.; Clevert, D.A. Parameterized hypercomplex graph neural networks for graph classification. In Proceedings of the International Conference on Artificial Neural Networks, Bratislava, Slovakia, 14–17 September 2021; pp. 204–216.
43. Beaini, D.; Passaro, S.; Létourneau, V.; Hamilton, W.; Corso, G.; Liò, P. Directional graph networks. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; PMLR: London, UK, 2021; pp. 748–758.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.