# GEOM-GCN: GEOMETRIC GRAPH CONVOLUTIONAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Message-passing neural networks (MPNNs) have been successfully applied to representation learning on graphs in a variety of real-world applications. However, two fundamental weaknesses of MPNNs' aggregators limit their ability to represent graph-structured data: losing the structural information of nodes in neighborhoods and lacking the ability to capture long-range dependencies in disassortative graphs. Few studies have noticed the weaknesses from different perspectives. From the observations on classical neural network and network geometry, we propose a novel geometric aggregation scheme for graph neural networks to overcome the two weaknesses. The behind basic idea is the aggregation on a graph can benefit from a continuous space underlying the graph. The proposed aggregation scheme is permutation-invariant and consists of three modules, node embedding, structural neighborhood, and bi-level aggregation. We also present an implementation of the scheme in graph convolutional networks, termed Geom-GCN, to perform transductive learning on graphs. Experimental results show the proposed Geom-GCN achieved state-of-the-art performance on a wide range of open datasets of graphs.

## 1 INTRODUCTION

Message-passing neural networks (MPNNs), such as GNN (Scarselli et al., 2008), ChebNet (Defferrard et al., 2016), GG-NN (Li et al., 2016), GCN (Kipf & Welling, 2017), are powerful for learning on graphs with various applications ranging from brain networks to online social network (Gilmer et al., 2017; Battaglia et al., 2018). In a layer of MPNNs, each node sends its feature representation, a "message", to the nodes in its neighborhood; and then updates its feature representation by aggregating all "messages" received from the neighborhood. The neighborhood is often defined as the set of adjacent nodes in graph. By adopting permutation-invariant aggregation functions (e.g., summation, maximum, and mean), MPNNs are able to learn representations which are invariant to isomorphic graphs, i.e., graphs that are topologically identical.

Although existing MPNNs have been successfully applied in a wide variety of scenarios, two fundamental weaknesses of MPNNs' aggregators limit their ability to represent graph-structured data. Firstly, *the aggregators lose the structural information of nodes in neighborhoods*. Permutation invariance is an essential requirement requirement for any graph learning method. To meet it, existing MPNNs adopt permutation-invariant aggregation functions which treat all "messages" from neighborhood as a set. For instance, GCN simply sums the normalized "messages" from all one-hop neighbors (Kipf & Welling, 2017). Such aggregation loses the structural information of nodes in neighborhood because it does not distinguish the "messages" from different nodes. Therefore, after such aggregation, we cannot know which node contributes what to the final aggregated output.

Without modeling such structural information, as shown in (Kondor et al., 2018) and (Xu et al., 2019), the existing MPNNs cannot discriminate between certain non-isomorphic graphs. In those cases, MPNN may map non-isomorphic graphs to the same feature representations, which is obviously not desirable for graph learning. Unlike MPNNs, classical convolutional neural networks (CNNs) avoid this problem by using aggregators (i.e., convolutional filters) with a structural receiving filed defined on grids, i.e., Euclidean space, and are hence able to distinguish every input. As shown by our experiments, such structural information often contains clues regarding topology

patterns in graph (e.g., hierarchy), and should be extracted and used to learn more discriminating representations for graph-structured data.

Secondly, *the aggregators lack the ability to capture long-range dependencies in disassortative graphs.* In MPNNs, the neighborhood is defined as the set of all neighbors one hop away (e.g., GCN), or all neighbors up to $r$ hops away (e.g., ChebNet). That only messages from nearby nodes are aggregated Through aggregating only messages from nearby nodes, the MPNNs are inclined to learn similar representations for proximal nodes in the graph. This implies that they are probably desirable methods for assortative graphs (e.g., citation networks (Kipf & Welling, 2017) and community networks (Chen et al., 2019)) where node homophily holds (i.e., similar nodes are more likely to be proximal, and vice versa.), but may be inappropriate to the disassortative graphs where node homophily does not hold (Newman, 2002). For example, Ribeiro et al. (2017) shows disassortative graphs where nodes of the same class exhibit high structural similarity but are far apart from each other. In such cases, the representation ability of MPNNs may be limited significantly, since they cannot capture the important features from distant but informative nodes

A straightforward strategy to address this limitation is to use a multi-layered architecture so as to receive "messages" from distant nodes. For instance, due to the localized nature of convolutional filters in classical CNNs, a single convolutional layer is similarly limited in its representational ability. CNNs typically use multiple layers connected in a hierarchical manner to learn complex and global representations. However, unlike CNNs, it is difficult for multi-layer MPNNs to learn good representations for disassortative graphs because of two reasons. On one hand, relevant messages from distant nodes are mixed indistinguishably with a large number of irrelevant messages from proximal nodes, which implies that the relevant information will be "washed out" and cannot be extracted effectively. On the other hand, the representations of different nodes would become very close, and every node's representation actually carries the information about the entire graph (Xu et al., 2018).

In this paper, we overcome the aforementioned weaknesses of graph neural networks starting from two basic observations: i) Classical neural networks effectively address the similar limitations thanks to the stationarity, locality, and compositionality in a continuous space (Bronstein et al., 2017); ii) The notion of network geometry bridges the gap between continuous space and graph (Hoff et al., 2002; Muscoloni et al., 2017). Network geometry aims to understand networks by revealing the latent continuous space underlying them, which assumes that nodes are sampled discretely from a latent continuous space and edges are established according to their distance. In the latent space, complicated topology patterns in graphs can be preserved and presented as intuitive geometry, such as subgraph (Narayanan et al., 2016), community (Ni et al., 2019), and hierarchy (Nickel & Kiela, 2017; 2018). Inspired by those two observations, we raise an enlightening question about the aggregation scheme in graph neural network.

- Can the aggregation on a graph benefit from a latent space, such as using geometry in the space to build structural neighborhoods and capture long-range dependencies on the graph?

To answer the above question, we propose a novel aggregation scheme for graph neural networks, termed the *geometric aggregation* scheme. In the scheme, we map a graph to a latent space via node embedding, and then use the geometric relationships defined in the latent space to build structural neighborhoods for aggregation. Also, we design a bi-level aggregator on the structural neighborhoods to update the feature representations of nodes in graph neural networks, which are able to guarantee permutation invariance for graph-structured data. Compared with existing MPNNs, the scheme extracts more structural information of the graph and can aggregate feature representations from distant nodes via mapping them to the neighborhood in the latent space.

We then present an implementation of the geometric aggregation scheme in graph convolutional networks, which we call *Geom-GCN*, to perform transductive learning, node classification, on graphs. We design structural neighborhood with particular geometric relationships in Euclidean and hyperbolic embedded space respectively. We choose different embedding methods to map the graph to a suitable latent space for different applications, where suitable topology patterns of graph are preserved. Finally, we validate and analyze Geom-GCN on a wide range of open datasets of graphs, and Geom-GCN achieved the state-of-the-art results.

In summary, the contribution of this paper is three-fold: i) We propose a novel geometric aggregation scheme for graph neural network, which operates in both graph and latent space, to overcome the

aforementioned two weaknesses; ii) We present an implementation of the scheme, Geom-GCN, for transductive learning tasks; iii) We validate and analyze Geom-GCN via extensive comparisons with state-of-the-art methods on challenging benchmarks.

## 2 GEOMETRIC AGGREGATION SCHEME

In this section, we start by presenting the geometric aggregation scheme, and then outline its advantages and limitations compared to existing works. As shown in Fig. 1, the aggregation scheme consist of three modules, node embedding (panel A1-A3), structural neighborhood (panel B), and bi-level aggregation (panel C). We will elaborate on them in the following.

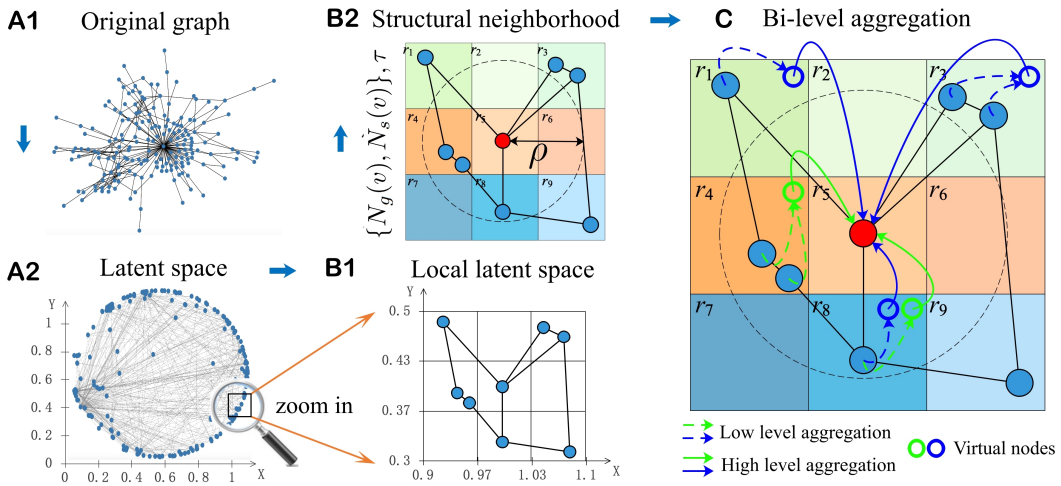



Figure 1: An illustration of the geometric aggregation scheme. **A1-A2** The original graph is mapped to a latent continuous space. **B1-B2** The structural neighborhood. All adjacent nodes lie in a small region around a center node in B1 for visualization. The neighborhood in the graph contains all adjacent nodes in graph; the neighborhood in the latent space contains the nodes within the dashed circle in B2. The relational operator $\tau$ is illustrated by a colorful $3 \times 3$ grid in B2 where each unit is corresponding to a geometric relationship to the red node. **C** Bi-level aggregation on the structural neighborhood. Dashed and solid arrows denote the low-level and high-level aggregation, respectively. Blue and green denote the aggregation on neighborhood of the graph and the latent space, respectively.

**A. Node embedding**. This is a fundamental module which maps the nodes in a graph to a latent continuous space. Let $\mathcal{G} = (V, E)$ be a graph, where each node $v \in V$ has a feature vector $\boldsymbol{x}_v$ and each edge $e \in E$ connects two nodes. Let $f : v \to \boldsymbol{z}_v$ be a mapping function from a node in graph to a representation vector. Here, $\boldsymbol{z}_v \in \mathbb{R}^d$ can also be considered as the position of node $v$ in a latent continuous space, and $d$ is the number of dimensions of the space. During the mapping, the structure and properties of graph are preserved and presented as the geometry in the latent space. For instance, hierarchical pattern in graph is presented as the distance to the original in hyperbolic space. One can employ various embedding methods to infer the latent space (Cai et al., 2018).

**B. Structural neighborhood**. Based on the graph and the latent space, we then build a structural neighborhood, $\mathcal{N}(v) = (\{N_g(v), N_s(v)\}, \tau)$, for the next aggregation. The structural neighborhood consists of a set of neighborhoods $\{N_g(v), N_s(v)\}$, and a global relational operator on nodes $\tau$.

The neighborhood in the graph, $N_g(v) = \{u | u \in V, (u, v) \in E\}$, is the set of adjacent nodes of $v$. The neighborhood in the latent space, $N_s(v) = \{u | u \in V, d(\boldsymbol{z}_u, \boldsymbol{z}_v) < \rho\}$, is the set of nodes from which the distance to $v$ is less than a pre-given parameter $\rho$. The distance function $d(\cdot, \cdot)$ depends on the particular metric in the space. Compared with $N_g(v)$, $N_s(v)$ may contain nodes which are far from $v$ in the graph, but have a certain similarity with $v$, and hence are mapped together with $v$ in the latent space though preserving the similarity. By aggregating in such neighborhood $N_s(v)$, the long-range dependencies in disassortative graphs can be captured.

The relational operator $\tau$ is a global function defined in the latent space. It inputs an ordered position pair $(\boldsymbol{z}_v, \boldsymbol{z}_u)$ of nodes $v$ and $u$, and outputs a discrete variable $r$ which indicates the geometric relationship from $v$ to $u$ in the space. For $u, v \in V$,

$$\tau : (\boldsymbol{z}_v, \boldsymbol{z}_u) \rightarrow r \in R,$$

where $R$ is the set of the geometric relationships. When one constructs the operator $\tau$, $r$ can be specified as an arbitrary geometric relationship of interest in the latent space. A requirement on $\tau$ is that it should guarantee that each ordered position pair has only one geometric relationship. For example, $\tau$ is illustrated in Fig. 1B by a colorful $3 \times 3$ grid in a 2-dimensional Euclidean space, in which each unit is corresponding to a geometric relationship to $v$.

**C. Bi-level aggregation**. With the structural neighborhood $\mathcal{N}(v)$, we propose a novel bi-level aggregation for graph neural network to update the hidden features of nodes. The bi-level aggregation consists of two aggregation functions and operates in a neural network layer. It can extract effectively structural information of nodes in neighborhoods as well as guarantee permutation invariance for graph. Let $\boldsymbol{h}_v^l$ be the hidden features of node $v$ at the $l$-th layer, and $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$ be the node features. The $l$-th layer updates $\boldsymbol{h}_v^l$ for every $v \in V$ by the following.

$$\begin{aligned}
&\boldsymbol{e}_{(i,r)}^{v,l+1} = p(\{\boldsymbol{h}_u^l | u \in N_i(v), \tau(\boldsymbol{z}_v, \boldsymbol{z}_u) = r\}), \forall i \in \{g, s\}, \forall r \in R && \text{(Low-level aggregation)} \\
&\boldsymbol{m}_v^{l+1} = \underset{i \in \{g,s\}, r \in R}{q}((\boldsymbol{e}_{(i,r)}^{v,l+1}, (i,r))) && \text{(High-level aggregation)} \\
&\boldsymbol{h}_v^{l+1} = \sigma(\boldsymbol{W}_l \cdot \boldsymbol{m}_v^{l+1}) && \text{(Non-linear transform)}
\end{aligned} \tag{1}$$

In the low-level, the hidden features of nodes that are in the same neighborhood and have the same geometric relationship are aggregated to a virtual node via the aggregation function $p$. The features of the virtual node are $\boldsymbol{e}_{(i,r)}^{v,l+1}$, and the virtual node is indexed by $(i, r)$ which is corresponding to the combination of a neighborhood $i$ and a relationship $r$. It is required to adopt a permutation-invariant function for $p$, such as an $L_p$-norm (the choice of $p = 1, 2,$ or $\infty$ results in average, energy, or max pooling). The low level aggregation is illustrated by dashed arrows in Fig. 1C.

In the high-level, the features of virtual nodes are further aggregated by function $q$. The inputs of function $q$ contain both the features of virtual nodes $\boldsymbol{e}_{(i,r)}^{v,l+1}$ and the identity of virtual nodes $(i, r)$. That is, we can adopt functions that take an ordered object as input for $q$, e.g., concatenation, to distinguish the features of different virtual nodes, thereby extracting the structural information of nodes in neighborhoods explicitly. The output of high-level aggregation is a vector $\boldsymbol{m}_v^{l+1}$. Then new hidden features of $v$, $\boldsymbol{h}_v^{(l+1)}$, are given by a non-linear transform, wherein $\boldsymbol{W}_l$ is a learnable weight matrix on the $l$-th layer shared by all nodes, and $\sigma(\cdot)$ is a non-linear activation function, e.g., a ReLU.

Permutation invariance is an essential requirement for aggregators in graph neural network. Thus, we then prove that our bi-level aggregation, Eq. 1, is able to guarantee invariance for any permutation of nodes. We give a definition for permutation-invariant mapping of graph at first.

**Definition 1.** *Let a bijective function $\psi : V \rightarrow V$ be a permutation for nodes, which renames $v \in V$ as $\psi(v) \in V$. Let $V^{'}$ and $E^{'}$ be the node and edge set after a permutation $\psi$, respectively. A mapping of graph, $\phi(\mathcal{G})$, is permutation-invariant if, given any permutation $\psi$, we have $\phi(\mathcal{G}) = \phi(\mathcal{G}^{'}), \mathcal{G}^{'} = (V^{'}, E^{'})$.*

**Lemma 1.** *For a composite function $\phi_1 \circ \phi_2(\mathcal{G})$, if $\phi_2(\mathcal{G})$ is permutation-invariant, the entire composite function $\phi_1 \circ \phi_2(\mathcal{G})$ is permutation-invariant.*

*Proof.* Let $\mathcal{G}^{'}$ be an isomorphic graph of $\mathcal{G}^{'}$ after a permutation $\psi$, as defined in Definition 1. If $\phi_2(\mathcal{G})$ is permutation-invariant, we have $\phi_2(\mathcal{G}) = \phi_2(\mathcal{G}^{'})$. Therefore, the entire composite function $\phi_1 \circ \phi_2(\mathcal{G})$ is permutation-invariant because $\phi_1 \circ \phi_2(\mathcal{G}) = \phi_1 \circ \phi_2(\mathcal{G}^{'})$. □

**Theorem 1.** *Given a graph $\mathcal{G} = (V, E)$ and its structural neighborhood $\mathcal{N}(v), \forall v \in V$, the bi-level aggregation, Eq. 1, is a permutation-invariant mapping of graph.*

*Proof.* The bi-level aggregation, Eq. 1, is a composite function, where the low-level aggregation is the input of the high-level aggregation. Thus, Eq. 1 is permutation-invariant if the low-level aggregation is permutation-invariant according to Lemma 1.

We then prove that the low-level aggregation is permutation-invariant. The low-level aggregation consists of $2 \times |R|$ sub-aggregations, each of which is corresponding to the nodes in a neighborhood $i$ and with a relationship $r$ to $v$. Firstly, the input of the sub-aggregations is permutation-invariant because both $i \in \{g, s\}$ and $r \in R$ are determined by the given structural neighborhood $\mathcal{N}(v), \forall v \in V$, which is constant for any permutation. Then, the low-level aggregation is clearly permutation-invariant because we adopt a permutation-invariant aggregation function $p$ for the sub-aggregations in Eq. 1. □

### 2.1 Comparisons to related work

We now discuss how the proposed geometric aggregation scheme overcomes the two aforementioned weaknesses, i.e., how it effectively models the structural information and long-range dependencies, in comparison to some closely related works.

To overcome the first weakness of MPNNs, i.e., losing the structural information of nodes in neighborhoods, the proposed scheme explicitly models the structural information of nodes in neighborhoods by exploiting their geometric relationships in the latent space and then extracting the information effectively by using the bi-level aggregations. In contrast, most existing works attempt to learn some implicit structure-like information to distinguish different neighbors when aggregating features. For example, GAT (Velickovic et al., 2017), LGCL (Gao et al., 2018) and GG-NN (Li et al., 2016) learn some weights on different neighbors by using attention mechanisms and node and edge attributes. CCN (Kondor et al., 2018) utilizes a covariance architecture to learn structure-aware representations. The major difference between these works and ours is that we offer an explicit and interpretable way to model the structural information of neighborhoods, with the assistance of the geometry in a latent space. We note that our work is orthogonal with existing methods and thus can be readily incorporated to further improve their performance. In particular, we exploit geometric relationships from the aspect of *graph topology*, while other methods focus on that of *feature representation*– the two aspects are complementary.

For the second weakness of MPNNs, i.e., lacking the ability to capture long-range dependencies, the proposed scheme models the long-range dependencies in disassortative graphs in two different ways. First of all, the distant (but similar) nodes in the graph can be mapped into a latent-space-based neighborhood of the target node, and then their useful feature representations can be used for aggregations. This way depends on an appropriate embedding method, which is able to preserve the similarities between the distant nodes and the target node. On the other hand, the structural information enables the method to distinguish different nodes in a graph-based neighborhood (as mentioned above). The informative nodes may have some special geometric relationships to the target node (e.g., a particular angle or distance), whose relevant features will be passed to the target node with much higher weights (and kept passed to its own neighbors), compared to the uninformative nodes. As a result, the long-range dependencies are captured indirectly through the whole message propagation process in all graph-based neighborhoods. In the literature, a recent method JK-Nets (Xu et al., 2018) captures the long-range dependencies by skipping some connections during feature aggregations.

#### 2.1.1 Case study to distinguish non-isomorphic graphs

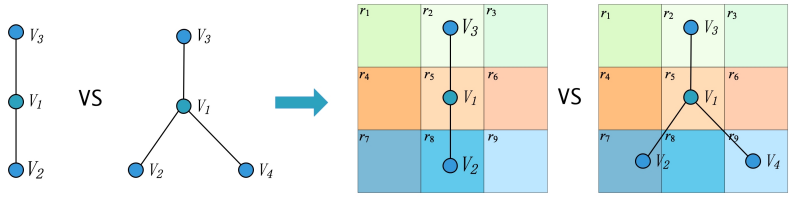



Figure 2: An illustration to distinguish non-isomorphic graphs by structural neighborhood.

In literature, there are several non-isomorphic example graphs that cannot be distinguished by the aggregators in existing MPNNs (Kondor et al., 2018; Xu et al., 2019). In this case study, we illustrate how to distinguish the non-isomorphic graphs via aggregators in structural neighborhood. We take two non-isomorphic graphs in (Xu et al., 2019) as shown in Fig. 2 (left), where every node has the same feature $a$ and after any mapping $f(a)$ is the same across all nodes. The aggregator, mean or maximum, over $f(a)$ remains $f(a)$, and hence the representations of every node are the same. That is, mean and maximum aggregators fail to distinguish the graphs.

In contrast, the two graphs become distinguishable once we apply a structural neighborhood to the graphs. Although the nodes have the same feature, they have different geometric relationships to the center node $V_1$ in the structural neighborhood, as shown in Fig. 2 (right). Taking aggregation for $V_1$ as an example, we can easily apply different mapping function $f_i, i \in R$ to the neighbors with different relationship $i$ to $V_1$. Then, the aggregator, mean or maximum, will have different output over different mappings $f_i(a)$ in two graph (i.e., $\{f_2(a), f_8(a)\}$ in the left graph and $\{f_2(a), f_7(a), f_9(a)\}$ in the right graph), thereby distinguishing the different structure of the two graphs.

## 3 GEOM-GCN: AN IMPLEMENTATION OF THE SCHEME

In this section, we present Geom-GCN, a specific implementation of the geometric aggregation scheme in graph convolutional networks, to perform transductive learning on a graph. To implement the general aggregation scheme, one needs to specify its three modules: node embedding, structural neighborhood, and aggregation function.

Node embedding is the fundamental. As shown in our experiments, a common embedding method which only preserves the connection patterns of a graph can already benefit the aggregation. For particular applications, one can specify embedding methods to create suitable latent spaces where particular topological patterns (e.g., hierarchy) are preserved. We employ three embedding methods, Isomap (Tenenbaum et al., 2000), Poincare embedding (Nickel & Kiela, 2017), and struc2vec (Ribeiro et al., 2017), which result in three Geom-GCN variants: Geom-GCN-I, Geom-GCN-P, and Geom-GCN-S. Isomap is a widely used low-dimensional embedding method, by which only distance patterns (lengths of shortest paths) are preserved explicitly in the latent space. Poincare embedding and struc2vec can create particular latent spaces that preserve hierarchies and local structures in a graph, respectively. We use an embedding space of dimension 2 for ease of explanation.

The structural neighborhood $\mathcal{N}(v) = (\{N_g(v), N_s(v)\}, \tau)$ of node $v$ includes its neighborhoods in both the graph and latent space. The neighborhood-in-graph $N_g(v)$ consists of the set of $v$'s adjacent nodes in the graph, and the neighborhood-in-latent-space $N_s(v)$ those nodes whose distances to $v$ are less than a parameter $\rho$ in the latent space. We determine $\rho$ by increasing $\rho$ from zero until the average size of $N_s(v)$ equals to that of $N_g(v)$, $\forall v \in V$– i.e., when the average neighborhood sizes in the graph and latent spaces are the same. We use Euclidean distance in the Euclidean space. In the hyperbolic space, we approximate the geodesic distance between two nodes via their Euclidean distance in the local tangent plane.

Here we simply implement the geometric operator $\tau$ as four relationships of the relative positions between two nodes in the 2-d Euclidean and hyperbolic spaces. Particularly, the relationship set $R$= {left upper, right upper, left lower, right lower}, and a $\tau(\boldsymbol{z}_v, \boldsymbol{z}_u)$ is given by Table 1. Notice that, we adopt the rectangular coordinate system in the Euclidean space and angular coordinate in the hyperbolic space. By this way, the relationship "upper" indicates the node nearer to the origin and thus lie in a higher level in a hierarchical graph. One can design a more sophisticated operator $\tau$, such as borrowing the structure of descriptors in manifold geometry (Kokkinos et al., 2012; Monti et al., 2017), thereby preserving more and richer neighborhood structures.

Table 1: The relationship operator

| $\tau(\boldsymbol{z}_v, \boldsymbol{z}_u)$ | $\boldsymbol{z}_v[0] > \boldsymbol{z}_u[0]$ | $\boldsymbol{z}_v[0] < \boldsymbol{z}_u[0]$ |
|---|---|---|
| $\boldsymbol{z}_v[1] < \boldsymbol{z}_u[1]$ | left upper | right upper |
| $\boldsymbol{z}_v[1] > \boldsymbol{z}_u[1]$ | left lower | right lower |

Finally, to implement bi-level aggregation, we adopt the same summation of normalized hidden features as GCN (Kipf & Welling, 2017) as the aggregation function $p$ in the low-level aggregation,

$$\boldsymbol{e}_{(i,r)}^{v,l+1} = \sum_{u \in N_i(v)} \delta(\tau(\boldsymbol{z}_v, \boldsymbol{z}_u), r)(\deg(v)\deg(u))^{\frac{1}{2}} \boldsymbol{h}_u^l, \forall i \in \{g, s\}, \forall r \in R,$$

where $\deg(v)$ is the degree of node $v$ in graph, and $\delta(\cdot,\cdot)$ is a Kronecker delta function that only allows the nodes with relationship $r$ to $v$ to be included. The features of all virtual nodes $\boldsymbol{e}_{(i,r)}^{v,l+1}$ are further aggregated in the high-level aggregation. The aggregation function $q$ is concatenation $||$ for all layers except the final layer, which uses mean for its aggregation function. Then, the overall bi-level aggregation of Geom-GCN is given by

$$\boldsymbol{h}_v^{l+1} = \sigma(W_l \cdot \underset{i \in \{g,p\}}{||} \underset{r \in R}{||} \boldsymbol{e}_{(i,r)}^{v,l+1})$$

where we use ReLU as the non-linear activation function $\sigma(\cdot)$.

# 4 EXPERIMENTS

We validate Geom-GCN's performance by comparing Geom-GCN's performance with the performance of Graph Convolutional Networks (GCN) (Kipf & Welling (2017)) and Graph Attention Networks (GAT) (Velickovic et al. (2017)). Two state-of-the-art graph neural networks, on transductive node label prediction tasks on a wide variety of open graph datasets.

## 4.1 DATASETS

We utilize nine open graph datasets to validate the proposed methods. An overview summary of characteristics of the datasets is given in Table 2.

Table 2: Datasets statistics

| **Dataset** | Cora | Cite. | Pubm. | Cham. | Squi. | Actor | Corn. | Texa. | Wisc. |
|---|---|---|---|---|---|---|---|---|---|
| # Nodes | 2708 | 3327 | 19717 | 2277 | 5201 | 7600 | 183 | 183 | 251 |
| # Edges | 5429 | 4732 | 44338 | 36101 | 217073 | 33544 | 295 | 309 | 499 |
| # Features | 1433 | 3703 | 500 | 2325 | 2089 | 931 | 1703 | 1703 | 1703 |
| # Classes | 7 | 6 | 3 | 4 | 4 | 4 | 5 | 5 | 5 |

*Citation networks*. Cora, Citeseer, and Pubmed are standard citation network benchmark datasets (Sen et al., 2008; Namata et al., 2012). In these networks, nodes represent papers, and edges denote citations of one paper by another. Node features are the bag-of-words representation of papers, and node labels are academic topics of a paper.

*WebKB*. WebKB[1] is a webpage dataset collected from computer science departments of various universities. We use the three subdatasets of it, Cornell, Texas, and Wisconsin, where nodes represent web pages, and edges are hyperlinks between them. Node features are the bag-of-words representation of web pages. The web pages were manually classified into the five categories, student, project, course, staff, and faculty.

*Actor co-occurrence network*. This dataset is the actor-only induced subgraph of the film-director-actor-writer network (Tang et al., 2009). Each nodes correspond to an actor, and the edge between two nodes denotes co-occurrence on the same Wikipedia page. Node features correspond to some keywords in the Wikipedia pages. We classify the nodes into four categories in term of words of actor's Wikipedia.

*Wikipedia network*. Chameleon and squirrel are two page-page networks on specific topics in Wikipedia (Rozemberczki et al., 2019). Nodes represent web pages and edges are mutual links between them. Node features correspond to some informative nouns in the Wikipedia pages. We classify the nodes into five categories in term of the number of the average monthly traffic of the page.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/index.html

### 4.2 EXPERIMENTAL SETUP

As mentioned in Section 3, we construct three Geom-GCN variants by employing three embedding methods, Isomap (Geom-GCN-I), Poincare embedding (Geom-GCN-P), and struc2vec (Geom-GCN-S). We specify the dimension of embedding space as two. We use the simple relationship operator $\tau$ defined in Table 1, and apply mean and concatenation as the low- and high- level aggregation function, respectively.

With the structural neighborhood, we perform a hyper-parameter search for all models on validation set. For fairness, the size of search space for each method is the same. The searching hyper-parameters include number of hidden unit, initial learning rate, weight decay, and dropout. We fix the number of layer to 2 and use Adam optimizer (Kingma & Ba, 2014) for all models. We use ReLU as activation functions for Geom-GCN and GCN, and ELU for GAT. The final hyper-parameter setting is dropout of $p = 0.5$, initial learning rate of $0.05$, patience of 100 epochs, weight decay of $5E$-6 (WebKB datasets) or $5E$-5 (the all other datasets). In GCN, the number of hidden unit is 16 (Cora), 16 (Citeseer), 64 (Pubmed), 32 (WebKB), 48 (Wikipedia), and 32 (Actor). In Geom-GCN, the number of hidden unit is 8 times as many as the number in GCN since Geom-GCN has 8 kinds of virtual nodes. For each attention head of GAT, the number of hidden unit is 8 (Citation networks), 32 (WebKB), 48 (Wikipedia), and 32 (Actor). GAT has 8 attention heads in layer one and 8 (Pubmed) or 1 (the all other datasets) attention heads in layer two.

For all graph datasets, we randomly split nodes of each class into $60\%$, $20\%$, and $20\%$ for training, validation and testing. With the hyper-parameters, we measure the performance of all models on the test sets over 10 random splits.

### 4.3 RESULTS AND ANALYSIS

Results are summarized in Table 3. The reported numbers denote the mean classification accuracy in percent. In general, Geom-GCN achieves state-of-the-art performance. The best performing method is highlighted. From the results, Isomap embedding (Geom-GCN-I) which only preserves the connection pattern of graph can already benefit the aggregation. And we can also specify an embedding method to create a suitable latent space (e.g., disassortative graph or hierarchical graph) for a particular application, by doing which a significant performance improvement is achieved (e.g., Geom-GCN-P).

Table 3: Mean Classification Accuracy (Percent)

| **Dataset** | Cora | Cite. | Pubm. | Cham. | Squi. | Actor | Corn. | Texa. | Wisc. |
|---|---|---|---|---|---|---|---|---|---|
| GCN | 85.77 | 73.68 | 88.13 | 28.18 | 23.96 | 27.92 | 52.70 | 52.16 | 45.88 |
| GAT | **86.37** | 74.32 | 87.62 | 42.93 | 30.03 | 28.15 | 54.32 | 58.38 | 49.41 |
| Geom-GCN-I | 85.19 | **77.99** | **90.05** | 60.31 | 33.32 | 28.65 | 56.76 | 57.58 | 58.24 |
| Geom-GCN-P | 84.93 | 75.14 | 88.09 | **60.90** | **38.14** | **31.28** | **60.81** | **67.57** | **64.12** |
| Geom-GCN-S | 85.27 | 74.71 | 84.75 | 59.96 | 36.24 | 29.83 | 55.68 | 59.73 | 56.67 |

#### 4.3.1 ANALYSIS OF CONTRIBUTIONS FROM TWO NEIGHBORHOODS

In this section, we further evaluate Geom-GCN by analyzing the contributions from graph neighborhood and latent space neighborhood, respectively. We construct new Geom-GCN variants by taking the structural neighborhood in only graph or latent space. For the variants with only graph neighborhood, we use g as a suffix of their name (e.g., Geom-GCN-I-g), and use suffix s to denote the variants with only latent space neighborhood (e.g., Geom-GCN-I-s). In this analysis, we use GCN as a baseline and the contributions are measured via the performance difference between GCN and the new variants. The reasults are summarized in Table 4, where the positive improvement is denoted by a $\uparrow$ and negative improvement by a $\downarrow$. The best performing method is also highlighted.

We also design an index denoted by $\beta$ to measure the node homophily of a graph,

$$\beta = \frac{1}{N} \sum_{v \in V} \frac{\text{Number of } v\text{'s neighbors who have the same label as } v}{\text{Number of } v\text{'s neighbors}}.$$

A larger $\beta$ value implies that node homophily, in term of node labels, is stronger for a given graph. From Table 4, assortative graphs (e.g., citation networks) have a larger $\beta$ than disassortative graphs (e.g., WebKB networks).

Table 4 exhibits three interesting patterns: i) Both graph and latent space neighborhoods benefit the aggregation in most cases; ii) The latent space neighborhoods have larger contributions in disassortative graphs (with a small $\beta$) than assortative ones, which implies relevant information from disconnected nodes is aggregated effectively in the neighborhood of the latent space; iii) To our surprise, several variants with only one neighborhood (Table 4) achieve better performances than the variants with both graph and latent space neighborhood (Tabel 3). We think the reason is that Geom-GCN with two neighborhoods aggregate more irrelevant "messages" than Geom-GCN with only one neighborhood, and the irrelevant "messages" adversely affect the performance. Thus, we believe an attention mechanism can alleviate this issue– which we will study as future work.

Table 4: Mean Classification Accuracy (Percent)

| **Dataset**<br>$\beta$ | Cora<br>0.83 | Cite.<br>0.71 | Cham.<br>0.25 | Corn.<br>0.11 | Texa.<br>0.06 | Wisc.<br>0.16 |
|---|---|---|---|---|---|---|
| Geom-GCN-I-g | 86.26<br>↑ 0.48 | **80.64**<br>↑ 6.96 | **68.00**<br>↑ 39.82 | 65.40<br>↑ 12.70 | 73.51<br>↑ 21.35 | 68.23<br>↑ 22.35 |
| Geom-GCN-I-s | 77.34<br>↓ 8.34 | 72.22<br>↓ 1.46 | 61.64<br>↑ 33.46 | 62.16<br>↑ 9.46 | 60.54<br>↑ 8.38 | 64.90<br>↑ 19.01 |
| Geom-GCN-P-g | 86.30<br>↑ 0.52 | 75.45<br>↑ 1.76 | 63.07<br>↑ 34.89 | 64.05<br>↑ 11.35 | **74.05**<br>↑ 21.89 | 69.41<br>↑ 23.53 |
| Geom-GCN-P-s | 73.14<br>↓ 12.63 | 71.65<br>↓ 2.04 | 43.20<br>↑ 15.02 | **75.40**<br>↑ 22.70 | 73.51<br>↑ 21.35 | **80.39**<br>↑ 34.51 |
| Geom-GCN-S-g | **87.00**<br>↑ 1.23 | 75.73<br>↑ 2.04 | 67.04<br>↑ 38.86 | 67.02<br>↑ 14.32 | 71.62<br>↑ 19.46 | 69.41<br>↑ 23.52 |
| Geom-GCN-S-s | 66.92<br>↓ 18.85 | 66.03<br>↓ 7.65 | 49.21<br>↑ 21.03 | 62.43<br>↑ 9.73 | 63.24<br>↑ 11.08 | 64.51<br>↑ 18.63 |

#### 4.3.2 ANALYSIS OF STRUCTURAL NEIGHBORHOOD COMBINATION

To evaluate which combination of structural neighborhoods is better, we construct new Geom-GCN variants once again by combining the structural neighborhoods from different embedded spaces. For the variants with graph neighborhood defined by Isomap and latent space neighborhood defined by poincare embedding, we use Geom-GCN-IP to denote it. The naming rule is the same for other combinations. The performances of all variants are summarized in Table 5. We can observe that several combinations achieve better performance than the Geom-GCN with structural neighborhoods defined by only one embedded space (Table 3); and there are also many combinations that have bad performances. To design an end-to-end framework that can automatically determine the right embedded space for Geom-GCN will be our future work.

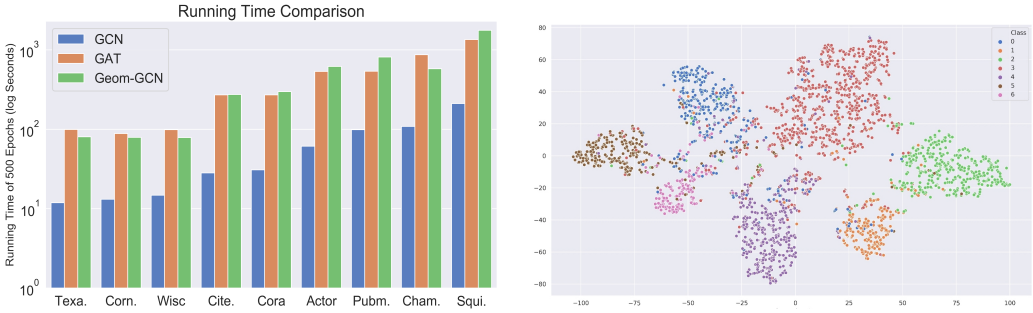
Table 5: Mean Classification Accuracy (Percent)

| **Dataset** | Cora | Cite. | Cham. | Corn. | Texa. | Wisc. |
|---|---|---|---|---|---|---|
| Geom-GCN-IP | 85.13 | **79.41** | 65.77 | **60.00** | 66.49 | 62.75 |
| Geom-GCN-PI | 85.09 | 75.08 | 59.19 | 58.11 | 58.11 | 58.63 |
| Geom-GCN-IS | 84.51 | 77.83 | 58.40 | 54.32 | 57.57 | 57.65 |
| Geom-GCN-SI | 85.31 | 75.50 | 62.13 | 57.30 | 60.00 | 55.10 |
| Geom-GCN-PS | **85.65** | 74.84 | 56.34 | 58.11 | 62.43 | 60.59 |
| Geom-GCN-SP | 85.43 | 75.71 | **65.81** | 58.38 | **67.84** | **65.10** |

#### 4.3.3 ANALYSIS OF TIME COMPLEXITY

Time complexity is very important for graph neural networks because real-world graphs are always very large. In this subsection, we present the theoretical time complexity of Geom-GCN and then compare the real running time of GCN, GAT, and Geom-GCN..

To update the representations of one node, the time complexity of Geom-GCN is $O(n \times m \times |R|)$ where $n$ is the size of input representations, $m$ is the number of hidden unit to transform each virtual node (i.e., $(i, r)$), and $|R|$ is the number of virtual nodes. Geom-GCN has $|R|$ times complexity than GCN whose time complexity is $O(n \times m)$. We compare the real running time of GCN, GAT, and Geom-GCN with the hyper-parameters described in Section 4.2. Results are shown in Fig. 3 (a). One can see that GAT and Geom-GCN are in the same level. An important future work is to solve the scalability of Geom-GCN by employing accelerating technologies, such as FastGCN (Chen et al., 2018) and SGC (Wu et al., 2019).




(a) (b)

Figure 3: (a) Running time comparison. GCN, GAT, and Geom-GCN both run 500 epochs, and y axis is the log seconds. (b) A visualization for the feature representations of Cora dataset obtained from Geom-GCN-P. Node colors denote labels. There are two obvious patterns, nodes with the same label exhibit a spatial clustering and all nodes distribute radially. The radial pattern indicates the hierarchy in graph.

#### 4.3.4 VISUALIZATION

To study what patterns are learned in the feature representations of node by Geom-GCN, we visualize the feature representations extracted by the last layer of Geom-GCN-P on Cora by mapping it into 2d space though t-SNE (Maaten & Hinton, 2008), as shown in Fig. 3 (b). In the figure, the nodes with the same label exhibit spatial clustering, which shows the discriminative power of Geom-GCN. That all nodes distribute radially in the figure indicates the proposed model learned hierarchical pattern in graph.

### 4.4 CONCLUSION AND FUTURE WORK

We tackle the two major weaknesses of existing message-passing neural networks over graphs– losses of discriminative structures and long-range dependencies. As our key insight, we bridge a discrete graph to a continuous geometric space via graph embedding. That is, we exploit the principle of convolution: *spatial aggregation over a meaningful space*– and our approach thus extracts or "recovers" the lost information (discriminative structures and long-range dependencies) in an embedded space from a graph. We proposed a general geometric aggregation scheme and instantiated it with several specific Geom-GCN implementations, and our experiments validated clear advantages over the state of the art. As future work, we will explore techniques for choosing a right embedding method– depending not only on input graphs but also on target applications.

## REFERENCES

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems (NeurIPS)*, pp. 3844–3852, 2016.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1416–1424. ACM, 2018.

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 (ICML)*, pp. 1263–1272, 2017.

Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Iasonas Kokkinos, Michael M Bronstein, Roee Litman, and Alex M Bronstein. Intrinsic shape context descriptors for deformable shapes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 159–166. IEEE, 2012.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon M. Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. In *International Conference on Learning Representations (ICLR)*, 2018.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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*, pp. 5115–5124, 2017.

Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature communications*, 8(1):1615, 2017.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, pp. 8, 2012.

Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *CoRR*, abs/1606.08928, 2016.

Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

Chien-Chun Ni, Yu-Yao Lin, Feng Luo, and Jie Gao. Community detection on networks with ricci flow. *Scientific reports*, 9(1):9984, 2019.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6338–6347, 2017.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning (ICML)*, pp. 3776–3785, 2018.

Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. *struc2vec*: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394, 2017.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *arXiv preprint arXiv:1909.13021*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 807–816. ACM, 2009.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning (ICML)*, pp. 5449–5458, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.