Neural Networks 161 (2023) 505–514

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

SP-GNN: Learning structure and position information from graphs

Yangrui Chen [a,*], Jiaxuan You [b], Jun He [c], Yuan Lin [c], Yanghua Peng [c], Chuan Wu [a], Yibo Zhu [c]

[a] Department of Computer Science, University of Hong Kong, 999077, Hong Kong, China
[b] Department of Computer Science, Stanford University, Stanford, 94305, USA
[c] ByteDance Inc., Beijing, 100086, China

ARTICLE INFO

Article history:
Received 4 July 2022
Received in revised form 30 November 2022
Accepted 31 January 2023
Available online 4 February 2023

Dataset link: Ogbn (Reference data)

Keywords:
Graph neural networks
Positional embedding
Structural embedding
Node classification
Graph classification

ABSTRACT

Graph neural network (GNN) is a powerful model for learning from graph data. However, existing GNNs may have limited expressive power, especially in terms of capturing adequate structural and positional information of input graphs. Structure properties and node position information are unique to graph-structured data, but few GNNs are capable of capturing them. This paper proposes *Structure-* and *Position-aware Graph Neural Networks (SP-GNN)*, a new class of GNNs offering generic and expressive power of graph data. SP-GNN enhances the expressive power of GNN architectures by incorporating a near-isometric proximity-aware position encoder and a scalable structure encoder. Further, given a GNN learning task, SP-GNN can be used to analyze positional and structural awareness of GNN tasks using the corresponding embeddings computed by the encoders. The awareness scores can guide fusion strategies of the extracted positional and structural information with raw features for better performance of GNNs on downstream tasks. We conduct extensive experiments using SP-GNN on various graph datasets and observe significant improvement in classification over existing GNN models.

1. Introduction

Graphs are everywhere in modern Internet applications, *e.g.*, social networks (Hamilton, Ying, & Leskovec, 2017; Kipf & Welling, 2017; Kwak, Lee, Park, & Moon, 2010), knowledge graphs (Hamaguchi, Oiwa, Shimbo, & Matsumoto, 2017), and citation networks (Sen et al., 2008; Wang et al., 2020). Graphs are natural representations to model objects and their relationships. Recent years have seen a surge of interest in applying deep learning methods to graph-structured data. A number of Graph Neural Networks (GNNs) (Hamilton et al., 2017; Kipf & Welling, 2017; Velickovic et al., 2018; Xu, Hu, Leskovec, & Jegelka, 2019) (*e.g.*, GCN Kipf & Welling, 2017, GraphSAGE Hamilton et al., 2017 and GAT Velickovic et al., 2018) have been proposed, which outperform traditional graph analysis methods in various web data analytics domains, such as recommendation (Ying et al., 2018), web-link prediction (Zhang & Chen, 2018), and product classification (He & McAuley, 2016; Hu et al., 2020).

Although GNN is considered a powerful model for encoding relational data, it has been proved that existing GNNs usually have limited expressive power in terms of graph topology discrimination, upper-bounded by the 1-Weisfeiler–Lehman (1-WL) graph isomorphism test (Xu et al., 2019). Concrete limitations of existing GNNs are two-fold: (1) the same computational ego-network[1] of two nodes may correspond to different local neighborhood structures, thus indistinguishable by GNNs; (2) two nodes that reside in different parts of a graph but with the same local neighborhood structures cannot be differentiated by GNNs. Fig. 2 gives an example. The computational graphs of existing GNNs on d-regular graphs in Fig. 2(a) are identical, thereby generating the same node embeddings. Similarly, GNNs cannot differentiate computational graphs of the task that is strongly correlated to the position either in Fig. 2(b). These limitations imply that existing GNNs fall short in capturing adequate structural and positional information in input graphs.

A few recent studies were devoted to developing heuristics to acquire extra structure and position information, and to increase the expressiveness and representation power of GNNs. ID-GNN (You et al., 2021) is designed to extend the structural information captured by existing GNN architectures by inductively encoding node identities[2] during heterogeneous message passing. P-GNN (You, Ying, & Leskovec, 2019) overcomes the position-expressiveness limitation of GNNs using anchor nodes

* Corresponding author.
E-mail address: yrchen@cs.hku.hk (Y. Chen).

https://doi.org/10.1016/j.neunet.2023.01.051
0893-6080/

[1] A computational graph specifies the procedure to produce a node's embedding (You, Gomes-Selman, Ying, & Leskovec, 2021).

[2] Identity denotes the number of the l-cycle starting from the root node.

sampled randomly from the graph for encoding node position information. These models are task-specific, *i.e.*, usually of benefits for structure- or position-aware tasks, and having poor scalability due to high complexity in preprocessing. The calculation of shortest path distance in P-GNN or long-hop cycles of identities in ID-GNN involves considerable time and memory consumption, thus inapplicable for analyzing large graphs with millions of nodes.

In addition, graph tasks differ in their needs for structural and positional information, making existing designs not generic enough to different graph prediction tasks. For example, graph classification tasks often require graph structure information, while positional information usually plays a more important role in node classification tasks. It is desirable to involve structural and positional information jointly but with different weights to enhance the expressive power of GNN architectures, according to their corresponding awareness for different tasks. This is especially important for tasks that are not pure structure- or position-aware, where the weighted awareness can benefit downstream tasks in both aspects.

This paper proposes **Structure- and Position-aware Graph Neural Network** (SP-GNN), a new class of GNNs offering generic, expressive GNN solutions to various graph-learning tasks. SP-GNN empowers GNN architectures to capture adequate structural and positional information, extending their expressive power beyond the 1-WL test. SP-GNN analyzes the position and structure awareness of different GNN tasks, according to which graph structure properties and node position information can be properly combined for augmenting downstream training, significantly improving the performance. We make the following contributions with SP-GNN.

▷ SP-GNN provides universal, efficient structure encoder and position encoder to augment a GNN architecture. The position encoder generates position embeddings that minimize the distortion over graph distance space. It avoids the expensive computation of shortest-path-distance (SPD) used by most position-related solutions, while effectively capturing positional relations among nodes in the graph. Meanwhile, the structure encoder computes structure embeddings of a given node by considering both its ego-network structure and its identity.

▷ Given a GNN task, SP-GNN calculates positional and structural awareness scores using its positional and structural embeddings. These awareness scores are useful to determine whether the task is position or structure sensitive. The score can also guide the fusion strategies of the extracted positional and structural information with raw features, as input to the GNN. Our theoretical results show that SP-GNN has greater expressive power beyond the 1-WL test, and is superior or comparable to state-of-the-art GNNs with augmented structure and position embeddings.

▷ We compare SP-GNN against GNNs with extensive experiments on various real-world graph datasets. We demonstrate that SP-GNN outperforms existing GNNs by 0.7%–5.4% and 2.0%–3.5% in node classification and graph classification tasks, respectively. Additionally, as compared to other expressive graph networks specifically designed for structure- or position-aware tasks, SP-GNN shows improved or comparable performance, further revealing the efficacy and generality of SP-GNN.

2. Background

2.1. Related work

Structure-aware node embeddings. Embedding network structures has been substantially studied over the past decades from different perspectives, which is instrumental for various machine learning tasks over graph-structured data, such as node classification and link prediction. Struc2Vec (Ribeiro, Saverese, & Figueiredo, 2017) is a representative embedding method that represents the structural information with multi-hop degree sequences, which builds structural similarity graphs using dynamic time warping as the similarity function, and then performs Skip-gram model with random walks over the similarity graphs to obtain the node embeddings. Such a method incurs extensive time and memory complexity, which limits its application to small graphs only. More efficient methods to capture the structure information have been proposed. GNNs (Hamilton et al., 2017; Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2019) with variants of aggregation and message passing functions have achieved significant success in encoding local neighborhood structures of nodes in a given graph. ID-GNN (You et al., 2021) extends the expressive power of GNNs beyond the 1-WL test by encoding the identity information during message passing. However, ID-GNN and other GNNs cannot model the relative position information of nodes in the graph, and thus may perform poorly in position-aware graph prediction tasks.

Position-aware node embeddings. The relative graph distance between vertices in a graph has also been investigated for improving GNN expressiveness. Position information plays a crucial role in distinguishing topologically identical substructures in a graph. Graph kernel methods (Vishwanathan, Schraudolph, Kondor, & Borgwardt, 2010; Yanardag & Vishwanathan, 2015), including random walk kernels (Sugiyama & Borgwardt, 2015) and Weisfeiler–Lehman kernels (Shervashidze, Schweitzer, Van Leeuwen, Mehlhorn, & Borgwardt, 2011), are used to encode positional information for graph representation learning. Traditional embedding learning methods also generate position-aware node embeddings, forcing nodes within short graph distances to be close in the embedding space. For example, DeepWalk (Perozzi, Al-Rfou, & Skiena, 2014) and Node2Vec (Grover & Leskovec, 2016) perform random walks over a graph and use sampled sequences to learn embeddings with Skip-gram objectives, while LINE (Tang et al., 2015) preserves first and second orders of network proximity within the graph and the embedding space. However, these embedding learning methods are transductive and involve a large number of trainable parameters (*i.e.*, node embeddings). Recent studies explore strategies that incorporate positional information with the GNN model. P-GNN (You et al., 2019) computes the shortest-path-distance between each node and the selected anchor nodes as positional embeddings, combined with the GNN representations before feeding them into the classification layer. GraphReach (Nishad, Agarwal, Bhattacharya, & Ranu, 2021) extends P-GNN with different anchor selection methods, *i.e.*, selecting anchor nodes by maximizing reachability with a greedy hill-climbing algorithm. It also suffers from high computation complexity, limiting its usage on large graphs. Benchmark-GNN (Dwivedi, Joshi, Laurent, Bengio, & Bresson, 2020) calculates Laplacian eigenvectors of the graph for features augmentation, and GraphBert (Zhang, Zhang, Xia, & Sun, 2020) uses multiple positional embeddings (*e.g.*, WL role embedding and hop-based distance embedding) as the input of the graph transformer model. These methods using position-aware node embedding are not applicable for structure-aware graph prediction tasks, and also suffer from high preprocessing complexity (*e.g.*, calculating the shortest-path-distance).

Unifying positional and structural embeddings. Srinivasan and Ribeiro (2020) propose a unified theoretical framework for positional embedding and structural representations according to the invariant theory. However, it is unclear how this theoretical framework maps onto real-world graph mining methods (Rossi et al., 2020; Zhu, Lu, Heimann, & Koutra, 2021). PhUSION (Zhu et al., 2021) is a proximity-based unified framework for computing structural and positional node embeddings, but it does not

analyze the awareness difference for GNN tasks when combining two types of node embeddings. Furthermore, the above methods can only be applied to small graph datasets due to high time and memory complexity.

2.2. Preliminaries

GNN definition. Graph neural networks are neural networks that learn meaningful node embeddings over graphs. We focus on popular GNN tasks (*e.g.*, node classification and graph classification tasks) which train a GNN over node features in an input graph, as described by $G = (V, E, F)$: V and E denote the node set and edge set of the graph, respectively, and $F = \{\mathbf{f}_v, v \in V\}$ is the set of feature vectors of the nodes. GNN embedding computing collectively aggregates information following the graph structure and performs various feature transformations. Following the definition of GNN in Xu et al. (2019), the computation in the kth layer of a GNN can be expressed as:

$$\begin{aligned}\mathbf{m}_u^{(k)} &= \mathrm{MSG}^{(k)}(h_u^k),\\ \mathbf{h}_v^{(k)} &= \mathrm{AGG}^{(k)}(\{\mathbf{m}_u^{(k)}, u \in N(v)\}, \mathbf{h}_v^{(k-1)}),\\ \mathbf{h}_v^{(0)} &= \mathbf{f}_v,\end{aligned} \tag{1}$$

where $N(v)$ is the neighbor set of vertex v, $\mathbf{m}_v^{(k)}$ denotes message embedding and $\mathbf{h}_v^{(k)}$ represents the node embedding. GNNs vary in the message function $\mathrm{MSG}(\cdot)$ and aggregation functions $\mathrm{AGG}(\cdot)$. For instance, the message passing function of GraphSAGE (Hamilton et al., 2017) can be represented as:

$$\begin{aligned}\mathbf{m}_u^{(k)} &= \mathrm{RELU}(W^{(k)}\mathbf{h}_u^{k-1}),\\ \mathbf{h}_v^{(k)} &= \mathrm{U}^{(k)}\mathrm{CONCAT}(\mathrm{MAX}(\{\mathbf{m}_u^{(k)}, u \in N(v)\}), \mathbf{h}_v^{(k-1)}),\\ \mathbf{h}_v^{(0)} &= \mathbf{f}_v\end{aligned} \tag{2}$$

where W^k and $\mathrm{U}^{(k)}$ are trainable parameters. The output node representations of the GNN are then fed into a classification network for downstream prediction tasks.

Structure- and position-awareness. P-GNN (You et al., 2019) gives definitions of structure-aware and position-aware node embeddings:

Definition 2.1. A node embedding $\mathbf{z}_i = f_p(v_i), \forall v_i \in V$, is position-aware if there exists a function $g_p(\cdot,\cdot)$ such that $d_{sp}(v_i, v_j) = g_p(\mathbf{z}_i, \mathbf{z}_j)$, where $d_{sp}(\cdot,\cdot)$ is the shortest path distance in G.

Definition 2.2. A node embedding $\mathbf{z}_i = f_{s_q}(v_i), \forall v_i \in V$, is structure-aware if it is a function involving up to q-hop network neighborhood of node v_i. Specifically, $\mathbf{z}_i = g_s(N_1(v_i), \ldots, N_q(v_i))$, where $N_k(v_i)$ is the set of nodes k hops away from node v_i, and g_s can be any function.

Following the definitions of structure-aware and position-aware node embeddings, we refer to a graph prediction task as being position- or structure-aware, if the label distance of two nodes in the graph is strongly correlated to the distance of their position- or structure-aware node embeddings. There are multiple optional metrics for quantifying the awareness, which can be roughly categorized into two types: (1) **non-parametric metrics** (*e.g.*, Kendall's Tau Kendall, 1948, Pearson correlation Benesty, Chen, Huang, & Cohen, 2009 and KNN Fix & Hodges, 1989), which directly measure the statistical correlation between embeddings and labels; and (2) **parametric metrics** (e.g., MLP Gardner & Dorling, 1998), which construct a neural network with embeddings as input to predict the pairwise label equivalence. Though non-parametric metrics do not make strong assumptions on the data distribution, they are slow and require large quantities of data to estimate the unknown function without over-fitting (Kanodia, Wolfgang, Stefansson, Ning, & Mahadevan, 2019). Thus, we adopt parametric metrics (*i.e.*, neural networks) to analyze the awareness of given graph tasks (see Section 4.2).

GNN tasks differ in their structure and position awareness. Though substantial efforts have been devoted to enhancing the expressive power of GNNs, there is no generic framework that analyzes both structure and position awareness of tasks, to our best knowledge. Most existing studies apply either structure- or position-aware methods for all tasks. Our goal is to analyze awareness of the GNN tasks on local network structures and global positions of nodes, in order to further boost the expressive power of GNNs as well as the downstream prediction performance.

Algorithm 1 The SP-GNN training framework

Input: Graph $G = (V, E, F)$; trainable functions $\mathrm{MSG}(\cdot)$ and AGG for message passing and aggregation; $\textsc{Sampler}(\cdot)$ extracts the L-hop computational graphs, Layer L; $\textsc{PosEncoder}(\cdot)$ and $\textsc{StructEncoder}(\cdot)$ are functions that compute position and structure embeddings, respectively; $\textsc{AnalyzeAware}(\cdot)$ generates the awareness scores
Output: Node embeddings $\mathbf{h_v}$ for all $v \in V$.

$\mathbf{f}_{pos}(v) \leftarrow \textsc{PosEncoder}(v)$
$\mathbf{f}_{struct}(v) \leftarrow \textsc{StructEncoder}(v)$
$w_{pos} \leftarrow \textsc{AnalyzeAware}(\mathbf{f}_{pos}(v))$
$w_{struct} \leftarrow \textsc{AnalyzeAware}(\mathbf{f}_{struct}(v))$
$w_{raw} \leftarrow \textsc{AnalyzeAware}(\mathbf{f}_v)$
$\mathbf{h_v^0} \leftarrow \mathrm{CONCAT}(w_{raw} \cdot \mathbf{f_v}, w_{pos} \cdot \mathbf{f}_{pos}(v), w_{struct} \cdot \mathbf{f}_{struct}(v)), \forall v \in V$
for $l \leftarrow 1, \ldots, L$ **do**
  for $v \in V$ **do**
    $\mathbf{m}_u^{(k)} = \mathrm{RELU}(W^{(k)}\mathbf{h}_u^{k-1})$
    $\mathbf{h}_v^{(k)} = \mathrm{U}^{(k)}\mathrm{CONCAT}(\mathrm{MAX}(\{\mathbf{m}_u^{(k)}, u \in N(v)\}), \mathbf{h}_v^{(k-1)})$
$\mathbf{h}_v \leftarrow \mathbf{h}_v^k$

3. SP-GNN Design

3.1. Overview of sp-gnn

SP-GNN incorporates position and structure awareness into existing GNNs, to enhance their expressive power. Fig. 1 illustrates the overall architecture of SP-GNN. There are two stages in SP-GNN: (1) **awareness scoring stage**, where SP-GNN computes the structure and position embeddings using respective encoders and calculates the awareness score for each type of features; (2) **feature fusion stage**, where SP-GNN fuses the position and structure embeddings with raw features, weighted with calculated awareness scores. Algorithm 1 describes the workflow.

Given a graph prediction task, SP-GNN first pre-processes the graph to extract the structural and positional information. The position encoder learns the positional node embeddings with the position loss, minimizing the distance between embedding similarity and graph distance similarity. The structure encoder computes the structural node embeddings by extracting the multi-hop neighborhood sequence and identity information. SP-GNN then uses a neural network (*e.g.*, MLP) to analyze the structure awareness and position awareness by predicting the pairwise label equivalence. Such awareness reflects the correlation between position or structure embeddings and labels, which are

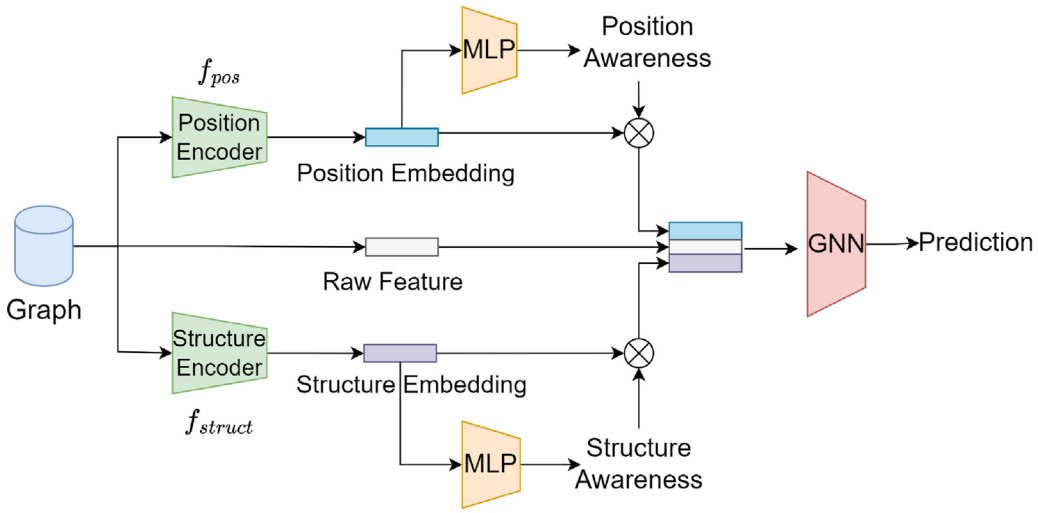


Fig. 1. An overview of SP-GNN.

task-specific and model-agnostic. Then we exploit *early fusion* for all types of features to improve the performance of downstream GNN learning. Specifically, we employ **CONCAT** operation to fuse positional and structural embeddings, together with raw features, if any, using pre-calculated awareness scores as the regularization weights.

The computation of two encoders is independent, which captures the positional and structural information, respectively. During training, the output embeddings are connected with the awareness scores, indicating their respective importance to the downstream tasks. The universal framework of SP-GNN benefits different types of tasks, while existing work can usually only enhance either structure-aware or position-aware GNN tasks.

3.2. Position encoders

The positional information, which reflects the distance between nodes in the graph, can benefit numerous GNN applications (Dwivedi et al., 2020; Nishad et al., 2021; You et al., 2019). The shortest-path-distance (SPD) matrix of the graph is widely used to encode distance information of nodes (*e.g.*, P-GNN You et al., 2019). Although this encoder may work well with small graphs, its time and memory complexity makes it unsuitable for large graphs. Another natural idea is to embed the graph space into a lower-dimensional Euclidean space isometrically that maintains the distance of two nodes. Unfortunately, it has been shown that isometric embeddings often do not exist (Charikar, 2018; Walker, Yan, Xiao, Wang, & Acharya, 2020). Our goal is to generate feasible embeddings with low complexity and *distortion* so that they can fully maintain the positional information. The distortion here is defined as follows Charikar (2018):

Definition 3.1. A mapping $f : X \to Y$ from a metric space (X, d_X) to a metric space (Y, d_Y) is an embedding with distortion α if there exists a constant $r \geq 0$ such that for every $x_1, x_2 \in X$,

$$r \cdot d_X(x_1, x_2) \leq d_Y(f(x_1), f(x_2)) \leq \alpha r \cdot d_X(x_1, x_2)$$

The Bourgain Theorem (Bourgain, 1985) provides a theoretical guarantee that there exist low-dimension embeddings with low distortion.

Theorem 3.1 (*Bourgain Theorem*)**.** *Given any finite metric space (V, d) with $|V| = n$, there exists an embedding of (V, d) into $\mathcal{R}^k$ under any l_p metric,*[3] *where $k = O(log^2 n)$, and the distortion of the embedding is $O(logn)$.*

Inspired by IGNN (Walker et al., 2020), we design a position encoder that learns the position embedding respecting the distances in the original graph. The position loss function is as follows:

$$L_{pos} = \sum_{v_i \neq v_j} [(1 - d_{cos}(f_{pos}(v_i), f_{pos}(v_j)))/2 - (1 - 1/d_{spd}(v_i, v_j))]^2$$

Here $d_{cos}(\cdot, \cdot)$ measures the cosine similarity of two position embeddings, $f_{pos}(v_i)$ and $f_{pos}(v_j)$; $d_{spd}(\cdot, \cdot)$ denotes the shortest-path-distance of two nodes. The position loss function describes the gap between the similarity of two nodes in the embedding space and that in the graph space. When the loss value approaches zero during training, the position embedding is a near-isometric embedding of the original graph space. IGNN incorporates the position loss function with downstream task loss function, instead of building a position encoder to compute position embedding of given graphs. SP-GNN is task-agnostic, hence more generic than IGNN.

The complexity to sample all pairs of nodes and calculate their shortest-path-distance for the loss function is high. We apply a sampling strategy for faster embedding learning. For each node, instead of forming training pairs with all other nodes, we sample its k-hop neighbors to avoid directly computing the graph distance. Further, we apply random sampling for nodes residing in very different parts of the graph, where the distance is set to be a large number (e.g., the diameter of the graph). The choice of sampling depth k reflects the trade-off between the complexity and the quality of the learned position embedding. We show that the distortion is bounded with the k-hop sampling method.

Proposition 1. *Suppose the positional embedding with 1-hop sampling satisfies $L_{pos}^1 \leq r$, where $0 \leq r \leq 1$ is a constant, i.e., $(d_{cos}(f_{pos}(v_i), f_{pos}(v_j)) - 1)^2 \leq r, \forall (v_i, v_j) \in E$. Then the distortion of the embedding over graph space is $\frac{1+\sqrt{r}}{1-\sqrt{r}}$.*

Proof. Suppose for any two nodes v_i and v_p with $(k-1)$-hop shortest path distance in the graph space, *i.e.*, $d_{spd}(v_i, v_p) = k - 1$, the corresponding positional embeddings satisfy

$$d_{cos}(f_{pos}(v_i), f_{pos}(v_p)) \leq (k-1)(1 + \sqrt{r}) \tag{3}$$

[3] l_p metric on $\mathcal{R}^k$ is defined as $d_p(x - y) = (\sum_{i=1}^k |x_i - y_i|^p)^{1/p}$, $x, y \in \mathcal{R}^k$.

Then, for any two vertices v_i and v_j with k-hop shortest path distance in the graph space, there must exist a k-length path $\{v_i, \ldots, v_p, v_j\}$ and a node v_p whose shortest path distance to v_i and v_j is $(k-1)$ and 1, respectively. According to the triangle rule, the cosine distance between $f_{pos}(v_i)$ and $f_{pos}(v_j)$ can be bounded by

$$\begin{aligned} d_{cos}(f_{pos}(v_i), f_{pos}(v_j)) \leq & d_{cos}(f_{pos}(v_i), f_{pos}(v_p)) \\ & + d_{cos}(f_{pos}(v_p), f_{pos}(v_j)) \end{aligned} \tag{4}$$

Plugging assumption $L^1_{pos} \leq r$ and Eq. (3) into Eq. (4), we have

$$\begin{aligned} d_{cos}(f_{pos}(v_i), f_{pos}(v_j)) \leq & (k-1)(1+\sqrt{r}) + (1+\sqrt{r}) \\ = & k(1+\sqrt{r}) \end{aligned} \tag{5}$$

By induction, we have $d_{cos}(f_{pos}(v_i), f_{pos}(v_j)) \leq k(1+\sqrt{r})$ for any node pair v_i and v_j with k-hop shortest path length. Similarly, we can prove that $d_{cos}(f_{pos}(v_i), f_{pos}(v_j)) \geq k(1-\sqrt{r})$. Thus, the distortion of $f_{pos}(\cdot)$ is $\frac{1+\sqrt{r}}{1-\sqrt{r}}$. □

The above proposition builds the relationship between the training loss of the position encoder and the corresponding distortion. It indicates that when the training loss of position encoder approaches 0, the distortion of the position encoder is close to 1. In addition, the proposition helps us to set the stopping criteria of the embedding learning. And with more sampling hops, the position encoder can capture more positional information, and may provide more performance gains (see Section 4.3).

3.3. Structure encoders

Structural information is important in many GNN applications. For example, collaborative filtering signals (Wang, He, Wang, Feng, & Chua, 2019) in long-hop neighbors have been demonstrated to be effective for the recommendation (Wu, Sun, Zhang, & Cui, 2020). Our goal of designing the structure encoder is to **capture the missing structural information in existing GNNs**. The structure embeddings can be used to differentiate isomorphic graphs that are indistinguishable by traditional GNNs. In structure-aware tasks, labels are strongly correlated to local neighborhood structures (Yanardag & Vishwanathan, 2015). Compared to traditional GNNs, our structure encoder seeks to capture longer-hop distance information. Training longer-hop subgraphs may lead to the over-smoothing problem with traditional GNNs (Li, Han, & Wu, 2018; Yan, Hashemi, Swersky, Yang, & Koutra, 2021), while encoding them as structure embeddings does not have such a problem.

The structure encoder of SP-GNN focuses on two types of structural information that are missing in traditional GNNs: **multi-hop neighbor sequence** and **identities**. The former embeds the computational ego-network of each node into a fixed vector, while the latter counts the cycle number starting from the root node of the computational ego-network, representing the node clustering property. The two types of structure information are complementary and important to improve the discrimination capability of nodes with similar structures. The computation of SP-GNN's structure encoder is simple and efficient as follows:

$$\begin{aligned} f_{neigh} &= [A \cdot \mathbf{1}, A \cdot (A \cdot \mathbf{1}), \ldots]^T, \\ f_{identity} &= [diag(A), diag(A^2), \ldots], \\ f_{struct} &= CONCAT(f_{neigh}, f_{identity}), \end{aligned} \tag{6}$$

where $A \in \mathrm{R}^{n \times n}$ denotes the adjacency matrix of the graph and $diag(\cdot)$ is a vector containing the diagonal elements of the matrix. SP-GNN's structure encoder scales well with the depth of the structure, and can easily encode tens of hops without neighbor explosion issues that other methods may encounter. Though the computation of identities may be time-consuming in large graphs, we calculate the identity information in the sampled subgraphs during training to minimize the overhead.

3.4. Awareness-based embedding fusion

Awareness scoring. With positional and structural embeddings, SP-GNN uses an MLP model to analyze the structure awareness and position awareness by predicting the pairwise label equivalence. We randomly select a certain number of positive and negative samples for each node to construct pairwise dataset, of which the 80% and 20% splits are used for training and testing. We then train MLP model with a cross-entropy loss function to predict the pairwise label equivalence using positional or structural embeddings. The testing results of the MLP model represents the corresponding awareness scores.

Having elaborated position and structure encoders and their awareness scoring method, we continue to propose two fusion strategies for combining both embeddings for GNN training: **early fusion** and **late fusion**.

Early fusion. The early fusion fuses information in feature space. It is natural to use classic feature fusion methods such as CONCAT or SUM to fuse position embedding and structure embedding with the raw feature vector (Barnum, Talukder, & Yue, 2020; Dwivedi et al., 2020). The message passing function can be rewritten as follows:

$$\begin{aligned} m_u^{(k)} &= MSG^{(k)}(h_u^k), \\ h_v^{(k)} &= AGG^{(k)}(\{m_u^{(k)}, u \in N(v)\}, h_v^{(k-1)}), \\ h_v^{(0)} &= Fuse(w_{pos} \cdot f_{pos}(v), w_{struct} \cdot f_{struct}(v), w_{raw} \cdot f_v), \end{aligned}$$

where the *Fuse* function can be CONCAT or SUM. w_{pos}, w_{struct} and w_{raw} denote the weights of each sort of embedding. In particular, w_{pos} and w_{struct} can also be regarded as structure and position awareness scores, respectively. The values of w_{pos} and w_{struct} depend on how sensitive the task is to positional and structural embeddings. There are various methods to calculate awareness score, and we will defer introducing our methods to Section 4.2.

The advantage of early fusion is that the neighborhood structural and positional embeddings can be aggregated during message passing, providing more information for GNNs to learn.

Late fusion. The other scheme is late fusion. It aims to combine the outputs of GNN, position encoder and structure encoder, and then construct them as input of the downstream task, e.g., a classifier. Under such a setting, we have two training strategies. The first is a 2-stage training: (1) separately training GNN, position encoder and structure encoder; (2) training the downstream task. The other training method is to perform joint training using a composite loss function that combines the task loss with position encoder and structure encoder losses, weighted by their awareness scores.

Our experiments show that no training strategy can always perform better than the other on different tasks. However, the early fusion works slightly better than the late fusion. Therefore, we use early fusion in the experiments.

3.5. SP-GNN analysis

3.5.1. Expressive power

SP-GNN has more expressive power than classical GNNs. It has been proved that the expressive power of classical GNNs is bounded by the 1-WL test (Xu et al., 2019), while GIN is as powerful as the 1-WL test and thus the most expressive GNN. Following the proof of ID-GNN (You et al., 2021), we show that SP-GNN is more expressive than classical GNNs.

Proposition 2. *Using GIN as the base model, SP-GNN can differentiate any graphs that GIN can differentiate, while distinguishing certain structures that GIN fails to distinguish.*

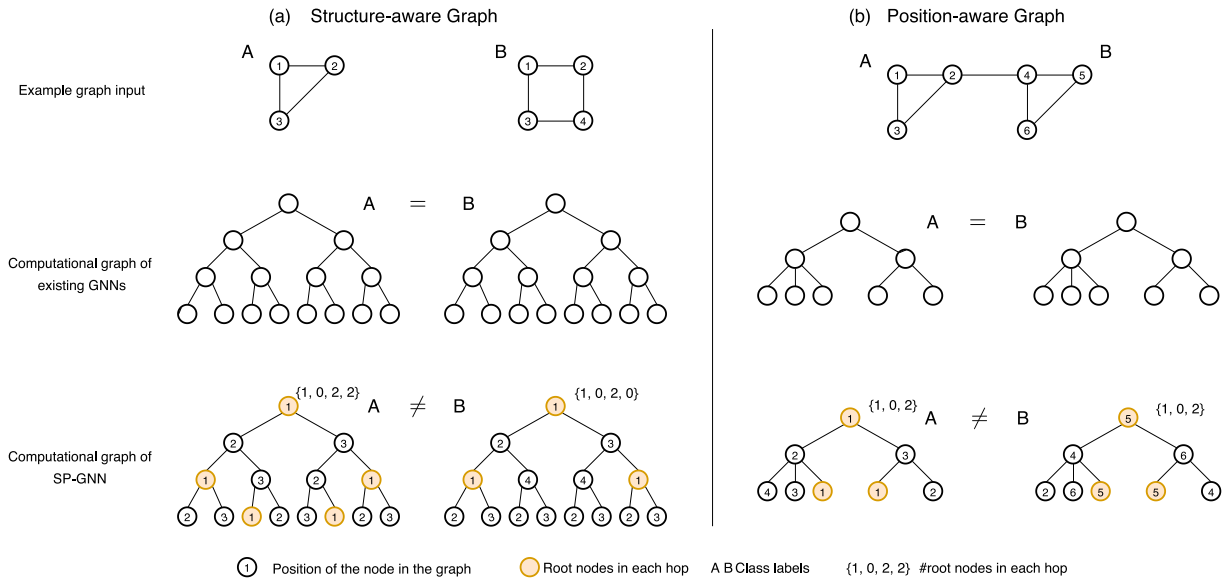


Fig. 2. The concrete graph examples of expressiveness of existing GNNs and SP-GNN.
Source: Adapted from You et al. (2021, 2019).

Proof. By setting the same message passing functions and aggregation functions as GIN, SP-GNN can be identical to GIN without augmented position and structure embeddings. This proves the first part of the proposition. For different d-regular graphs, both GIN and the 1-WL test fail to distinguish them, while SP-GNN can generate different outputs with the help of structural information. Fig. 2(a) shows that though the topology of two computational graphs is identical, SP-GNN captures the identity information and generates different structure embeddings ($\{1, 0, 2, 2\}$ for 3-regular graph and $\{1, 0, 2, 0\}$ for 4-regular graph). Thus, different d-regular graphs can be differentiated by SP-GNN. This proves the latter part. □

SP-GNN has more expressive power than pure structure- or position-aware GNNs. We next show that SP-GNN is also more expressive than pure structure-aware or position-aware GNNs. Fig. 2(b) gives a failure example of pure structure-aware GNNs, where two nodes in very different parts of the graph differ in the label. Extra structural properties (*e.g.*, identities in ID-GNN You et al., 2021) offer no help in this case due to identical structures (see Fig. 2(b)), such that the pure structure-aware GNNs fail to differentiate the computational graphs. Similarly, pure position-aware GNNs (*e.g.*, P-GNN You et al., 2019) cannot discriminate the example graphs in Fig. 2(a), as the node position information cannot reflect the dissimilarity of nodes between graphs. By unifying structural and positional information, SP-GNN can differentiate both cases, thus more expressive than pure structure-aware and position-aware GNNs.

3.5.2. Complexity analysis

Besides forward and backward computation of downstream GNN training, SP-GNN's complexity mainly lies in the position and structure encoders. To learn the position embeddings, the initial position encoder visits all pairs of nodes in the graph, incurring a cost of $\mathcal{O}(|V|^2)$ in each training epoch and $\mathcal{O}(|E||V| + |V|^2 log|V|)$ for calculating the shortest path with Johnson's algorithm (Johnson, 1977). To scale to large graphs, we optimize the position encoder by sampling k-hop neighbors for each node to construct training pairs, which reduces the complexity to $O(k|V|)$. The structure encoder encodes the m-hop structural information using adjacency matrix multiplication as in Eq. (6), of which the complexity is $\mathcal{O}(m|V|^3)$. This can be accelerated with parallel computation using GPUs.

4. Experiments

4.1. Methodology

Datasets. We evaluate SP-GNN on seven datasets with different graph sizes, graph types and graph tasks. For node classification tasks, we use citation network datasets (1) Cora (Sen et al., 2008), (2) CiteSeer (Giles, Bollacker, & Lawrence, 1998) and (3) Ogbn-arxiv (Wang et al., 2020), where nodes correspond to publications and edges to citations. Each publication in Cora and CiteSeer is described by a one-hot word vector indicating the presence of the corresponding word from the dictionary, while node features of Ogbn-arxiv are obtained by averaging the embeddings of words in its title and abstract. We also use (4) Ogbn-products (Hu et al., 2020) dataset, a large Amazon product co-purchasing network with bag-of-words features from the product descriptions. For graph classification tasks, three social network datasets, (5) IMDB-BINARY (Yanardag & Vishwanathan, 2015), (6) IMDB-MULTI (Yanardag & Vishwanathan, 2015) and (7) COLLAB (Yanardag & Vishwanathan, 2015), are adopted, without raw features. In IMDB-BINARY and IMDB-MULTI, nodes represent actors/actresses, and an edge exists if they appear in the same movie. Each graph in COLLAB corresponds to a researcher's ego network. We summarize the detailed statistics of the datasets in Table 1.

We follow the default dataset splitting from the dataset sources of all node classification tasks. For graph classification tasks, random splits of 80%, 10%, 10% of the dataset are used for training, validation and testing, respectively. We train all the models until loss converges and report the testing performance of the best model on the validation set.

Experimental Setup. We use a three-layer GAT (Velickovic et al., 2018) model with four attention heads as the base model of SP-GNN. The number of hidden units is 256 and 64 for node and graph classification tasks, respectively. Ogbn-products dataset is too large for full-batch training (DGL, 2020), and hence we adopt mini-batch training with neighbor sampling size {5, 10, 15}. We use full batch training on all other datasets. For large graphs, Ogbn-arxiv and Ogbn-products, the structure encoder of SP-GNN encodes 8-hop neighbor sequence and 3-hop identity information. We calculate 8-hop structural information for all

Table 1
Statistics of the real-world datasets.

Dataset	Node classification				Graph classification		
	Cora	CiteSeer	Ogbn-arxiv	Ogbn-products	IMDB-BINARY	IMDB-MULTI	COLLAB
#Nodes per graph	2708	3327	169343	2449029	19.77	13.00	74.49
#Edges per graph	10 556	9228	1 166 243	123 718 280	193.062	131.871	4914.43
#Graphs	1	1	1	1	1000	1500	5000
Avg. degree	3.90	2.77	6.89	50.52	9.77	10.143	65.97
#Raw feats	1433	3703	128	100	0	0	0
#Labels	7	6	40	47	2	3	3

Table 2
Comparing SP-GNN with baselines.

	Node classification				Graph classification		
	Ogbn-arxiv	Ogbn-products	Cora	CiteSeer	IMDB-BINARY	IMDB-MULTI	COLLAB
GCN	0.717 ± 0.003	0.756 ± 0.002	0.867 ± 0.002	0.743 ± 0.010	0.723 ± 0.029	0.471 ± 0.019	0.775 ± 0.038
GraphSAGE	0.715 ± 0.003	0.783 ± 0.002	0.873 ± 0.008	0.743 ± 0.012	0.717 ± 0.005	0.491 ± 0.016	0.777 ± 0.043
GAT	0.732 ± 0.001	0.795 ± 0.006	0.872 ± 0.014	0.746 ± 0.005	0.763 ± 0.025	0.536 ± 0.006	0.818 ± 0.010
GIN	0.707 ± 0.002	0.781 ± 0.004	0.869 ± 0.011	0.731 ± 0.003	0.750 ± 0.041	0.469 ± 0.006	0.755 ± 0.011
SP-GNN-Structure	0.733 ± 0.001	0.803 ± 0.003	0.873 ± 0.011	0.794 ± 0.007	0.780 ± 0.016	0.542 ± 0.011	0.833 ± 0.003
SP-GNN-Position	0.736 ± 0.001	0.816 ± 0.001	0.878 ± 0.009	0.775 ± 0.009	0.777 ± 0.013	0.558 ± 0.008	0.816 ± 0.002
SP-GNN-Both	**0.739 ± 0.002**	**0.816 ± 0.005**	**0.880 ± 0.009**	**0.800 ± 0.005**	**0.783 ± 0.005**	**0.571 ± 0.006**	**0.839 ± 0.004**
Over best GNN	0.7%	2.2%	0.7%	5.4%	2.0%	3.5%	2.2%

other datasets. The position encoder computes 128-dimensional position embedding using a 3-hop sampling method, *i.e.*, constructing pairs for each node from neighbors within three hops. We also randomly sample 5 negatives for each node, setting the distance as the diameter of the graph.

Baseline models. We compare SP-GNN with four widely adopted GNN models: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018) and GIN (Xu et al., 2019). We set the same hyper-parameters for all models for a fair comparison. Other state-of-the-art methods are also used to evaluate the efficacy of encoders of SP-GNN, including two structure encoders (Struc2Vec Ribeiro et al., 2017 and ID-GNN You et al., 2021) and four position encoders (shortest-path-distance with anchor nodes You et al., 2019, Laplacian eigenvectors Dwivedi et al., 2020, Node2Vec Grover & Leskovec, 2016 and LINE Tang et al., 2015).

4.2. Overall performance

Table 2 shows the test accuracy of baselines and SP-GNN when training on various node classification and graph classification tasks. The results demonstrate significant improvement of SP-GNN over existing GNNs.

Node Classification. For node classification, we observe accuracy improved by positional and structural information. Also, the gain from positional information is noticeably larger than structural information. The best performance result is achieved when both positional and structural embeddings are used, with a significant improvement of 0.7% to 5.4% across datasets.

Graph Classification. Graph classification tells the other side of the story: structural information plays a more important role. On all three datasets, the test accuracy of SP-GNN with the structure encoder outperforms the ones with the positional encoder. It is also worth noting that position encoder provides little improvement compared to structure encoder. We believe, for graph-level tasks, the position information inside each graph may not reflect the relative relation between graphs, thus providing less improvement than node-level tasks.

Awareness Analysis. We derive the awareness scores of position and structure for various datasets and tasks, with results in Tables 3, 4 and Fig. 3. In each dataset, we randomly sample 5 positives and 5 negatives for each data sample and adopt a 3-layer MLP (Gardner & Dorling, 1998) as the classifier to predict the label equivalence given pairwise embeddings. Test data samples are masked out during the training process. Since the label equivalence of pairwise data samples is agnostic to the number of classes, the awareness metric is universal to different tasks.

Table 3
Structure and position awareness scores for real-world graph classification tasks.

Dataset	Structure	Position
IMDB-BINARY	0.6424	0.5040
IMDB-MULTI	0.5618	0.4992
COLLAB	0.7589	0.4109

On graph classification datasets, we observe high structure awareness, indicating that structural information outweighs positions. Position embeddings only reflect the distance of nodes within each graph instead of between disjointed graphs. This explains the larger improvement with the structure encoder over the position encoder in Table 2 on graph classification tasks.

On node classification tasks, we see that most graphs are position-aware, and the highest position awareness score, 0.9369, is achieved on Ogbn-products. The high position awareness score aligns with the results in Table 2, where the overall performance is greatly boosted by positional information.

Convergence. We also compare training convergence of SP-GNN that contains extra structural and positional information with baseline model GAT on Ogbn-arxiv and Ogbn-products datasets. In Fig. 4, we observe that SP-GNN not only achieves higher test accuracy over the baseline when the model has converged, but also has a faster convergence rate. This motivates the usage of SP-GNN in training a large graph with a limited time budget.

4.3. Deep dive of the results

Structure Encoders. We compare the structure encoder of SP-GNN with other methods that also capture the structure information of graphs: (1) Struc2Vec (Ribeiro et al., 2017), which represents the structural information with multi-hop degree sequences, (2) GraphSAGE (Hamilton et al., 2017), a classic GNN to aggregate neighborhood information, and (3) ID-GNN (You

Table 4
Structure, position and raw feature awareness scores for real-world node classification tasks.

Dataset	Structure	Position	Raw Feature
Cora	0.5075	0.8708	0.8133
CiteSeer	0.5214	0.7525	0.8373
Pubmed	0.5133	0.8299	0.9075
Ogbn-arxiv	0.6769	0.8508	0.7691
Ogbn-products	0.5287	0.9369	0.8017

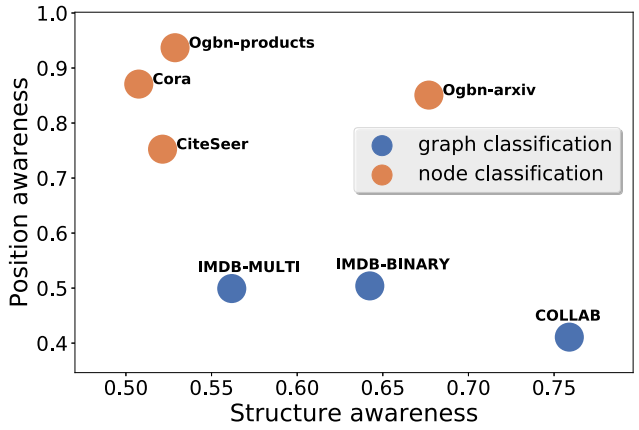


Fig. 3. Task awareness.

et al., 2021), which incorporates GNN architectures with identity information. Two structure-aware datasets, USA air-traffic network (Ribeiro et al., 2017) (with 1190 nodes and 13 599 edges representing commercial flights between airports) and actor co-occurrence network (Ma, Qin, Qiu, Zheng, & Wang, 2019) (with 7779 nodes and 26 752 edges indicating actors co-occurrence in films), are used for comparison.

In Table 5, the numbers are test accuracy. The structure encoder of SP-GNN achieves the best performance on both datasets, outperforming the best baseline model by 0.8% and 0.7%, respectively. We also observe the unsteady performance of the baselines, while the improvement brought by SP-GNN is stable.

Position Encoders. We compare the position encoder of SP-GNN with the following position-aware baselines: (1) two non-parametric methods: shortest-path-distance with 1024 anchor nodes (You et al., 2019) and Laplacian eigenvectors (Dwivedi et al., 2020), and (2) two parametric embedding learning methods: Node2Vec (Grover & Leskovec, 2016) and LINE (Tang et al., 2015). For the shortest-path-distance method, we use PCA (Pearson, 1901) to reduce the embedding dimension to 64. For the Laplacian eigenvectors encoder, we use the same hyper-parameter setting as in the original paper (Dwivedi et al., 2020). Two position-aware datasets, Ogbn-arxiv and Ogbn-products, are used to test the position encoders. Positional embeddings of each position encoder are computed with the given graphs, which are then fused with the raw features and applied to the GAT model for node classification. The test accuracies of the GAT model in different cases are given in Table 6.

The position encoder of SP-GNN performs the best on two position-aware graphs, improving the test accuracy by up to 0.25% and 1.15% over the best baselines, respectively. Compared to embedding learning methods, SP-GNN achieves better performance because it is less distorted with respect to the graph distance metric. SP-GNN also outperforms the shortest-path-distance encoder by 0.33% on Ogbn-arxiv, since the random selection of anchor nodes may affect position information encoding. Calculating the shortest path distances on larger datasets such as Ogbn-products (with over two million nodes) is time-consuming, and Laplacian matrix decomposition becomes impossible due to memory constraints. Hence, the two non-parametric methods fail to run on Ogbn-products. On the other hand, SP-GNN can scale well on large datasets, and outperform the best baseline by 1.15%.

Table 5
Comparison of structure encoders.

Methods	USA airports	Actor
Struc2Vec	0.6438	0.4748
GraphSAGE	0.5994	0.4936
ID-GNN	0.479	0.6350
SP-GNN-Struct	**0.6513**	**0.6414**

Table 6
Comparison of position encoders.

Methods	Ogbn-arxiv	Ogbn-products
Shortest-path-distance	0.7323	>10 h
Laplacian eigenvectors	0.7231	OOM
Node2Vec	0.7331	0.7978
LINE	0.7313	0.8045
SP-GNN-Pos	**0.7356**	**0.8160**

Table 7
Comparison of different methods unifying position and structure information.

Methods	Cora	CiteSeer
PPMI	0.8641	0.7367
CGNN	0.726	0.7210
MC-SVD	0.672	0.7370
SP-GNN	**0.8801**	**0.8003**

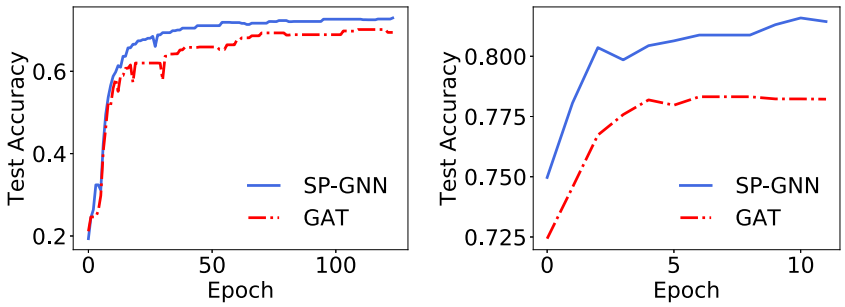
Table 8
SP-GNN with different base GNN models on Cora dataset. "Baseline" denotes original GNN models without position and structural information augmentation.

Methods	GCN	GAT	GraphSAGE	GIN
Baseline	0.8662	0.8721	0.8776	0.8579
SP-GNN	**0.8782**	**0.8801**	**0.8813**	**0.8616**

Comparison with other unifying methods. We compare SP-GNN with the methods that are also unifying positional and structural embeddings: PPMI (Zhu et al., 2021), CGNN (Srinivasan & Ribeiro, 2020) and MC-SVD (Srinivasan & Ribeiro, 2020). We use the same hyper-parameters as in the original papers. Since these methods cannot scale to large graphs (*e.g.*, Ogbn-arxiv and Ogbn-products) due to out-of-memory (OOM) and large preprocessing time (>10 h), we train them on Cora and CiteSeer datasets. Table 7 gives the results. SP-GNN achieves the best performance among all schemes. Compared to other methods that also unify positional and structural embeddings, SP-GNN uses shadow encoders and non-parametric methods to augment state-of-the-art GNNs, which has a different training manner and involves no duplicate information.

Different base models. We evaluate SP-GNN with four base models on Cora dataset (see Table 8). SP-GNN is not model-dependent and can benefit different GNNs with extra positional and structural information. Overall, SP-GNN increases the accuracy by 0.37% to 1.2% with different base models.

Case study: Efficacy of Computing Position and Structure Awareness. Finally, we want to show that the computed positional and structure awareness scores are highly indicative for GNN architecture design. Applying a high-quality position encoder to a position-aware task is likely to be useful, but applying it to a structure-aware task is likely not. To validate this argument, we pick two tasks, Ogbn-arxiv and Ogbn-products, which



(a) Ogbn-arxiv (b) Ogbn-products

Fig. 4. Convergence comparison.

Table 9
Impact of positional information.

#Dim	Ogbn-arxiv	Ogbn-products
32	0.7340	0.7827
64	0.7330	0.7965
128	0.7356	0.8156
#sampled hop	Ogbn-arxiv	Ogbn-products
1	0.7313	0.8045
2	0.7333	0.8056
3	0.7356	0.8156

have high position awareness scores. Since we know these tasks are highly position-aware, we choose to devote more computational resources on the positional encoder, by increasing the hop size and embedding dimension of the positional encoder. As is shown in Table 9, the final accuracy is indeed improved by up to 0.5% and 3.3% on Ogbn-arxiv and Ogbn-products, respectively.

5. Conclusion

In this paper, we propose SP-GNN, a powerful GNN framework that incorporates position and structure information to enhance existing GNN models. SP-GNN employs a near-isometric proximity-aware position encoder and a scalable structure encoder to improve the expressiveness of GNN architectures beyond the 1-WL test. Further, SP-GNN can be used to analyze the positional and structural awareness of given GNN tasks. The awareness scores can guide optimizations in position encoder or structure encoder, depending on the analyzed awareness. Extensive experiments on comprehensive graph datasets demonstrate significant improvement of SP-GNN over existing GNN models. Our proposed model is generic and scalable, best to be used in various Web applications such as recommendation systems or social network analysis. The influence of more base models in SP-GNN on different types of GNN tasks is our future work, together with larger scale graph datasets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to the used public data at the Attach File step.

Ogbn (Reference data) (Open Graph Benchmark)

Acknowledgments

This work was supported in part by grants from Hong Kong RGC under the contracts HKU 17207621 and 17203522, and ITC PRP grant (PRP/082/20FX).

References

Barnum, G., Talukder, S., & Yue, Y. (2020). On the benefits of early fusion in multimodal representation learning. arXiv preprint arXiv:2011.07191.
Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1–4). Springer.
Bourgain, J. (1985). On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, *52*(1–2), 46–52.
Charikar, M. (2018). Metric spaces, embeddings, and distortion. http://web.stanford.edu/class/cs369m/cs369mlecture1.pdf.
DGL (2020). Deep graph library (DGL). https://github.com/dmlc/dgl.
Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., & Bresson, X. (2020). Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982.
Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*(3), 238–247.
Gardner, M. W., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Enviroment*, *32*(14–15), 2627–2636.
Giles, C. L., Bollacker, K. D., & Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on digital libraries* (pp. 89–98).
Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).
Hamaguchi, T., Oiwa, H., Shimbo, M., & Matsumoto, Y. (2017). Knowledge transfer for out-of-knowledge-base entities : A graph neural network approach. In *Proc. of the 26th international joint conference on artificial intelligence (IJCAI)*.
Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proc. of advances in neural information processing systems (NeurIPS)*.
He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web* (pp. 507–517).
Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., et al. (2020). Open graph benchmark: Datasets for machine learning on graphs. CoRR abs/2005.00687.
Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, *24*(1), 1–13.
Kanodia, S., Wolfgang, A., Stefansson, G. K., Ning, B., & Mahadevan, S. (2019). Mass–Radius relationship for m dwarf exoplanets: Comparing nonparametric and parametric methods. *Astrophysical Journal*, *882*(1), 38.
Kendall, M. G. (1948). Rank correlation methods.
Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proc. of the 5th international conference on learning representations ICLR*.
Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a social network or a news media? In *Proceedings of the 19th international conference on world wide web* (pp. 591–600).
Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.
Ma, X., Qin, G., Qiu, Z., Zheng, M., & Wang, Z. (2019). RiWalk: Fast structural node embedding via role identification. In *2019 IEEE international conference on data mining (ICDM)* (pp. 478–487). IEEE.

Nishad, S., Agarwal, S., Bhattacharya, A., & Ranu, S. (2021). GraphReach: Position-aware graph neural network using reachability estimations. In Z.-H. Zhou (Ed.), *Proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI 2021*.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Ribeiro, L. F., Saverese, P. H., & Figueiredo, D. R. (2017). 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).

Rossi, R. A., Jin, D., Kim, S., Ahmed, N. K., Koutra, D., & Lee, J. B. (2020). On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, *12*(9).

Srinivasan, B., & Ribeiro, B. (2020). On the equivalence between positional node embeddings and structural graph representations. In *8th international conference on learning representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Sugiyama, M., & Borgwardt, K. (2015). Halting in random walk kernels. *Advances in Neural Information Processing Systems*, *28*, 1639–1647.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *Proc. of the 6th international conference on learning representations (ICLR)*.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., & Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, *11*, 1201–1242.

Walker, M., Yan, B., Xiao, Y., Wang, Y., & Acharya, A. (2020). Isometric graph neural networks. arXiv preprint arXiv:2006.09554.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174).

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., & Kanakia, A. (2020). Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, *1*(1), 396–413.

Wu, S., Sun, F., Zhang, W., & Cui, B. (2020). Graph neural networks in recommender systems: a survey. arXiv preprint arXiv:2011.02260.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In *7th international conference on learning representations, ICLR 2019, New Orleans, la, USA, May 6-9, 2019*.

Yan, Y., Hashemi, M., Swersky, K., Yang, Y., & Koutra, D. (2021). Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. arXiv preprint arXiv:2102.06462.

Yanardag, P., & Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1365–1374).

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proc. of the 24th ACM international conference on knowledge discovery & data mining (KDD)*.

You, J., Gomes-Selman, J. M., Ying, R., & Leskovec, J. (2021). Identity-aware graph neural networks. In *Thirty-Fifth AAAI conference on artificial intelligence, AAAI 2021*.

You, J., Ying, R., & Leskovec, J. (2019). Position-aware graph neural networks. In *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*.

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. In *Proc. of advances in neural information processing systems (NeurIPS)*.

Zhang, J., Zhang, H., Xia, C., & Sun, L. (2020). Graph-bert: Only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140.

Zhu, J., Lu, X., Heimann, M., & Koutra, D. (2021). Node proximity is all you need: Unified structural and positional node and graph embedding. In *Proceedings of the 2021 SIAM international conference on data mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*.